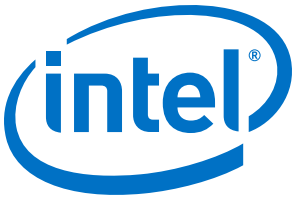
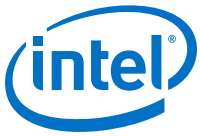


Arria 10 SoC Device Design Guidelines

AN-763
2017.05.08

Subscribe
Send Feedback

Contents



1 Overview of Design Guidelines for Arria 10 SoC FPGAs

The purpose of this document is to provide a set of guidelines and recommendations, as well as a list of factors to consider, for designs that use the Intel® Arria® 10 SoC FPGA devices. This document assists you in the planning and early design phases of the Arria 10 SoC FPGA design, Qsys sub-system design, board design and software application design.

This application note does not include all the Arria 10 Hard Processor System (HPS) device details, features or information on designing the hardware or software system. For more information about the Arria 10 HPS features and individual peripherals, refer to the Arria 10 Hard Processor System Technical Reference Manual.

Note: Intel recommends that you use Quartus Prime Pro and the SoC EDS Pro to develop Arria 10 SoC designs. Although Quartus Prime Standard and the SoC EDS Standard continue to support the Arria 10 SoC family on a maintenance basis, future enhancements will only be supported with the Pro software.

Hardware developed with Quartus Prime Pro only supports software developed with the SoC EDS Pro. Hardware developed with Quartus Prime Standard only supports software developed with the SoC EDS Standard.

Related Links

Arria 10 Hard Processor System Technical Reference Manual

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2008 Registered

1.1 The SoC FPGA Designer's Checklist

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

Table 1. The SoC FPGA Designer's Checklist

Step Title	Links	Check (X)
HPS Designer's Checklist for SoC FPGAs		
Start your SoC FPGA Design here	Recommended Starting Point for HPS-to-FPGA Interface Designs on page 24	
	Determining your SoC FPGA Topology on page 24	
Design Considerations for Connecting Device I/O to HPS Peripherals and Memory	HPS Pin Multiplexing Design Considerations on page 26	
	HPS I/O Settings: Constraints and Drive Strengths on page 27	
HPS Clocking and Reset Design Considerations	HPS Clock Planning on page 28	
	Early Pin Planning and I/O Assignment Analysis on page 28	
	Pin Features and Connections for HPS Clocks, Reset and PoR on page 29	
	Internal Clocks on page 29	
	HPS Reset during FPGA reconfiguration and FPGA configuration failures on page 30	
HPS EMIF Design Considerations	Considerations for Connecting HPS to SDRAM on page 32	
	HPS SDRAM I/O Locations on page 33	
	Integrating the Arria 10 HPS EMIF with the SoC FPGA Device on page 36	
	HPS Memory Debug on page 36	
Design Considerations for FPGA based Accelerators	Choosing a DMA Controller on page 37	
	Optimizing DMA Master Bandwidth through HPS Interconnect on page 37	
Board Designer's Checklist for SoC FPGAs		
HPS Power Design Considerations	Power On Board Bring Up and Boot ROM/Boot Loader Debugging on page 38	
	Early System and Board Planning on page 38	
	Design Considerations for HPS and FPGA Power Supplies for SoC FPGA devices on page 40	
	Pin Connection Considerations for Board Designs on page 41	
	Power Analysis on page 42	
	Power Optimization on page 42	
FPGA Reconfiguration	Flash Update with HPS Reboot on page 44	
	Partial Reconfiguration of the SoC FPGA on page 45	
Boundary Scan for HPS	Boundary Scan for HPS on page 45	
HPS EMAC PHY Interfaces	PHY Interfaces Connected Through Shared I/O on page 47	
	PHY Interfaces Connected Through FPGA I/O on page 51	

continued...

Step Title	Links	Check (X)
	MDIO on page 54	
	Common PHY Interface Design Considerations on page 54	
Interface Design Guidelines	USB Interface Design Guidelines on page 55	
	QSPI Flash Interface Design Guidelines on page 56	
	SD/MMC and eMMC Card Interface Design Guidelines on page 56	
	NAND Flash Interface Design Guidelines on page 57	
	UART Interface Design Guidelines on page 58	
	I2C Interface Design Guidelines on page 58	
Embedded Software Designer's Checklist for SoC FPGAs		
Assemble the components of your Software Development Platform	Assembling the components of your Software Development Platform on page 60	
	Golden Hardware Reference Design (GHRD) on page 61	
Select an Operating System (OS) for your application	Using Linux or RTOS on page 63	
	Developing a Bare-Metal Application on page 63	
	Using Symmetrical vs. Asymmetrical Multiprocessing (SMP vs. AMP) Modes on page 64	
Assemble your Software Development Platform for Linux	Golden System Reference Design (GSRD) for Linux on page 65	
	Linux Device Tree Design Considerations on page 67	
Assemble your Software Development Platform for Bare-metal Application	Assembling your Software Development Platform for a Bare-metal Application on page 68	
Assemble your Software Development Platform for Partner OS/RTOS Application	Assembling your Software Development Platform for Partner OS or RTOS on page 69	
Choose the Boot Loader Software	Choosing Boot Loader Software on page 69	
Selecting Software Tools for Development, Debug and Trace	Selecting Software Build Tools on page 70	
	Selecting Software Debug Tools on page 71	
	Selecting Software Trace Tools on page 71	
Board Bring Up Considerations	Board Bring Up Considerations on page 71	
Boot and Configuration Design Considerations	Boot Design Considerations on page 72	
	Configuration on page 76	
Flash Device Driver Considerations	Flash Device Driver Design Considerations on page 78	
HPS ECC Design Considerations	HPS ECC Design Considerations on page 78	
Security Design Considerations	Security Design Considerations on page 81	
Embedded Software Debugging and Trace	Embedded Software Debugging and Trace on page 82	

1.2 Overview of HPS Design Guidelines for SoC FPGA design

Table 2. HPS: Design Guidelines Overview

Stages of the HPS Design Flow	Guidelines	Links
Hardware and Software Partitioning	Determine your system topology and use it as a starting point for your HPS to FPGA interface design	Guidelines for Interconnecting the Arria 10 HPS and FPGA on page 10
HPS Pin Multiplexing and I/O Configuration Settings	Plan configuration settings for the HPS system including I/O multiplexing options, interface to FPGA and SDRAM, clocks, peripheral settings	Design Considerations for Connecting Device I/O to HPS Peripherals and Memory on page 25
HPS Clocks and Reset Considerations	HPS clocks and cold and warm reset considerations	HPS Clocking and Reset Design Considerations on page 28
HPS EMIF Considerations	Usage of the HPS EMIF controller and related considerations	HPS EMIF Design Considerations on page 31
FPGA Accelerator Design Considerations	Design considerations to manage coherency between FPGA accelerators and the HPS	DMA Considerations on page 37

1.3 Overview of Board Design Guidelines for SoC FPGA Design

Table 3. Board Design: Design Guidelines Overview

Stages of the Board Design Flow	Guidelines	Links
HPS Power design considerations	Power on board bring up, early power estimation, design considerations for HPS and FPGA power supplies, power analysis and power optimization	HPS Power Design Considerations on page 38
FPGA Reconfiguration	Reconfiguring FPGA is it becomes unresponsive using flash update with HPS reboot or partial reconfiguration	FPGA Reconfiguration on page 44
Board design guidelines for HPS interfaces	Includes EMAC, USB, QSPI, SD/MMC, NAND, UART and I^2C	Design Guidelines for HPS Interfaces on page 46

1.4 Overview of Embedded Software Design Guidelines for SoC FPGA Design

Table 4. Embedded Software: Design Guidelines Overview

Stages of the Embedded Software Design Flow	Guidelines	Links
Operating System (OS) Considerations	OS considerations to meet your application needs, including real time, software reuse, support and ease of use considerations	Selecting an Operating System for your application on page 63
Boot Loader Considerations	Boot loader considerations to meet your application needs, including GPL requirements and features.	Choosing Boot Loader Software on page 69
Boot and Configuration Design Considerations	Boot source, boot clock, boot fuses, configuration flows	Boot and Configuration Design Considerations on page 72
HPS ECC Considerations	ECC for external SDRAM interface, L2 cache data memory, flash memory	HPS ECC Design Considerations on page 78
Security Design Considerations	Secured boot, secure design IP, encryption and authentication	Security Design Considerations on page 81
Embedded Software Debugging and Trace	Types of tracing and trace interfaces	Embedded Software Debugging and Trace on page 82
Software Tools for Development, Debug and Trace	Design considerations for selecting software tools	Selecting Software Tools for Development, Debug and Trace on page 70

2 Guidelines for Interconnecting the Arria 10 HPS and FPGA

The memory-mapped connectivity between the HPS and the FPGA fabric is a crucial tool to maximize the performance of your design. Use the guidelines in this chapter for recommended topologies to optimize your system's performance.

Design guidelines for the remainder of the FPGA portion of your design are provided in the *Arria 10 Device Design Guidelines*.

Related Links

Arria 10 Device Design Guidelines

2.1 Overview of HPS Memory-Mapped Interfaces

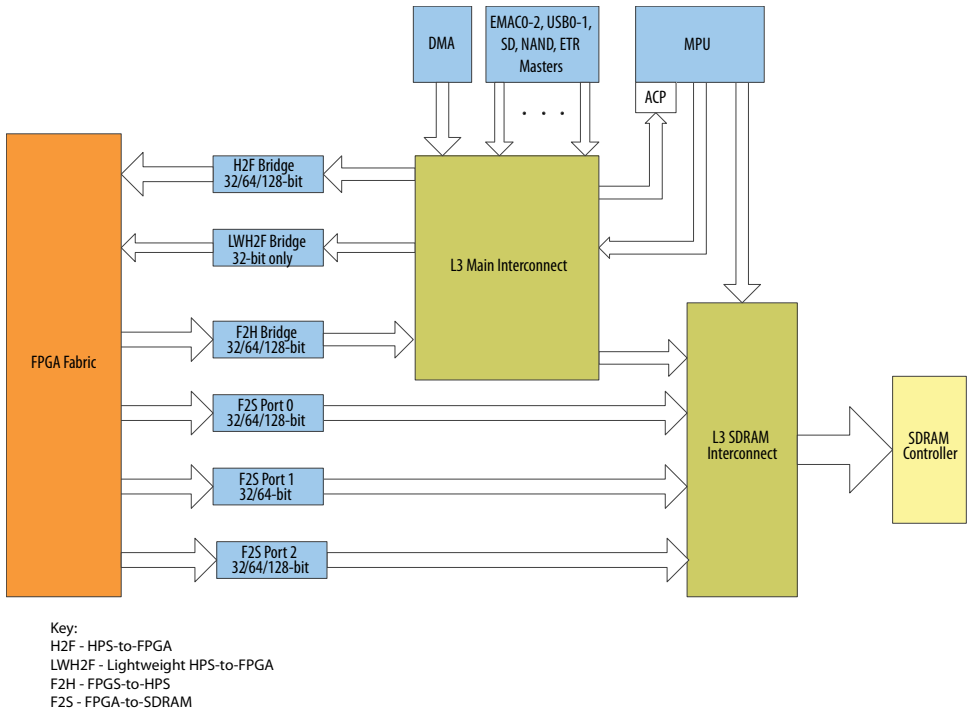
The HPS exposes five ARM Advanced Microcontroller Bus Architecture (AMBA®) 3 Advanced eXtensible Interface (AXI™) memory mapped interfaces for memory transfers between the HPS and the FPGA fabric. Each port has a different purpose and associated direction.

The HPS component in Qsys can be connected to masters and slaves that implement Avalon-MM interfaces or supported AMBA 3 and 4 interfaces such as AMBA 3 AXI and AMBA 4 AXI4. Qsys generates an interconnect to handle interoperability between interfaces that have different capabilities or use different protocols.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2008 Registered

Figure 1. Arria 10 HPS Connectivity



2.1.1 HPS-to-FPGA Bridge

GUIDELINE: Use the FPGA-to-HPS bridge to connect memory hosted by the FPGA to the HPS.

The HPS-to-FPGA bridge allows masters in the HPS such as the microprocessor unit (MPU), DMA, or peripherals with integrated masters to access memory hosted by the FPGA portion of the SoC device. This bridge supports 32, 64, and 128-bit data paths allowing the width to be tuned to the largest slave data width in the FPGA fabric connected to the bridge. This bridge is intended to be used by masters performing bursting transfers and should not be used for accessing peripheral registers in the FPGA fabric. Control and status register accesses should be sent to the lightweight HPS-to-FPGA bridge instead.

GUIDELINE: If memory connected to the FPGA-to-HPS bridge is used for HPS boot, ensure that its slave address is set to 0x0 in Qsys.

When the HPS BSEL pins are set to boot from FPGA (BSEL = 1) the processor executes code hosted by the FPGA residing at offset 0x0 from the HPS-to-FPGA bridge. This is the only bridge that can be used for hosting code at boot time.

2.1.2 Lightweight HPS-to-FPGA Bridge

GUIDELINE: Use the lightweight HPS-to-FPGA bridge to connect IP that needs to be controlled by the HPS.

The lightweight HPS-to-FPGA bridge allows masters in the HPS to access memory-mapped control slave ports in the FPGA portion of the SoC device. Typically, only the MPU inside the HPS accesses this bridge to perform control and status register accesses to peripherals in the FPGA.

GUIDELINE: Do not use the lightweight HPS-to-FPGA bridge for FPGA memory. Instead use the HPS-to-FPGA bridge for memory.

When the MPU accesses control and status registers within peripherals, these transactions are typically strongly ordered (non-posted). By dedicating the lightweight HPS-to-FPGA bridge to only register accesses, the access time is minimized because bursting traffic is routed to the HPS-to-FPGA bridge instead. The lightweight HPS-to-FPGA bridge has a fixed 32-bit width connection to the FPGA fabric because most IP cores implement 32-bit control and status registers; but Qsys can adapt the transactions to widths other than 32 bits within the FPGA generated network interconnect.

2.1.3 FPGA-to-HPS Bridge

GUIDELINE: Use the FPGA-to-HPS bridge for cacheable accesses to the HPS from masters in the FPGA.

The FPGA-to-HPS bridge allows masters implemented in the FPGA fabric to access memory and peripherals inside the HPS. This bridge supports 32, 64, and 128-bit data paths so that you can adjust it to be as wide as the widest master implemented in the FPGA.

GUIDELINE: Use the FPGA-to-HPS bridge to access cache-coherent memory, peripherals, or on-chip RAM in the HPS from masters in the FPGA.

Although this bridge has direct connectivity to the SDRAM subsystem, the main intent of the bridge is to provide access to peripherals and on-chip memory, as well as provide cache coherency with connectivity to the MPU accelerator coherency port (ACP).

To access the HPS SDRAM directly without coherency you should connect masters in the FPGA to the FPGA-to-SDRAM ports instead, because they provide much more bandwidth and lower-latency access.

2.1.4 FPGA-to-SDRAM Ports

GUIDELINE: Use the FPGA-to-SDRAM ports for non-cacheable access to the HPS SDRAM from masters in the FPGA.

The FPGA-to-SDRAM ports allow masters implemented in the FPGA fabric to directly access HPS SDRAM without the transactions flowing through the L3 Main Interconnect. These interfaces connect only to the HPS SDRAM subsystem so it is recommended to use them in your design if the FPGA needs high-throughput, low-latency access to the HPS SDRAM. The exception to this recommendation is if the FPGA requires cache

coherent access to SDRAM, the FPGA-to-SDRAM interfaces cannot access the MPU ACP slave so if you require a master implemented in the FPGA to access cache coherent data, ensure that it is connected to the FPGA-to-HPS bridge instead.

There are three FPGA-to-SDRAM ports with ports FPGA-to-SDRAM0 and FPGA-to-SDRAM2 supporting 32, 64, and 128-bit datapaths and FPGA-to-SDRAM1 supporting 32 and 64-bit datapaths. Four combinations of port configurations are available with the maximum aggregate bandwidth being available when FPGA-to-SDRAM0 and FPGA-to-SDRAM2 are setup for 128-bit datapaths. Each FPGA-to-SDRAM port is serviced independently including the ports connect to the L3 Main Interconnect and the MPU. The four possible FPGA-to-SDRAM port combinations are listed the table below.

Table 5. Available FPGA-to-SDRAM Port Combinations

Port Configuration	FPGA-to-SDRAM0	FPGA-to-SDRAM1	FPGA-to-SDRAM2
1	32 bits	32 bits	32 bits
2	64 bits	64 bits	64 bits
3	128 bits	Unavailable	128 bits
4	128 bits	32 bits	64 bits

GUIDELINE: Ensure that data accessed by the FPGA through the FPGA-to-SDRAM ports is flushed from the level 1 (L1) and level 2 (L2) caches before accessing data via the FPGA-to-SDRAM ports.

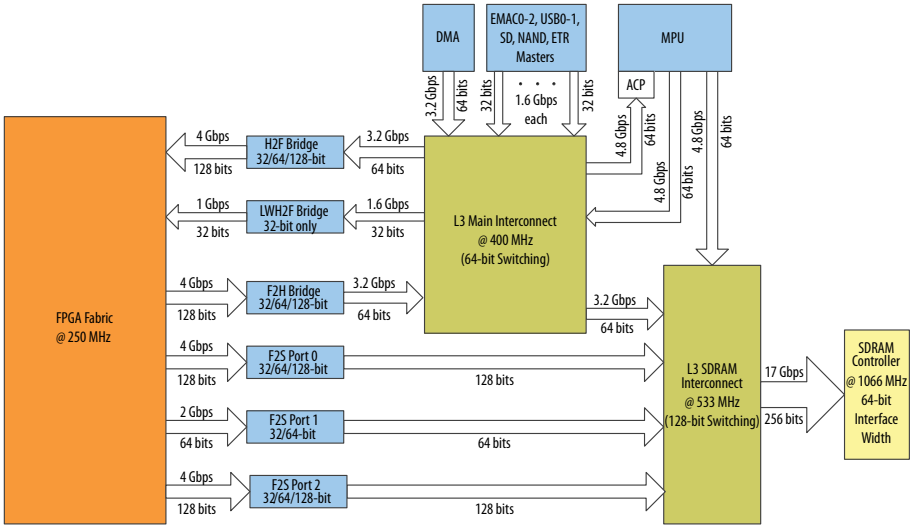
This ensures the latest copy of data is resident in SDRAM before the FPGA attempts to access the data.

2.1.5 Interface Bandwidths

To identify which interface should be used to move data between the HPS and FPGA fabric an understanding of the bandwidth of each interface is necessary. The figure below illustrates the peak throughput available between the HPS and FPGA fabric as well as the internal bandwidths within the HPS. The example shown assumes that the FPGA fabric operates at 250 MHz, the MPU operates at 1200 MHz, and the 64-bit external SDRAM operates at 1067 MHz.

Figure 2. Arria 10 HPS Memory Mapped Bandwidth

For abbreviations, refer to the figure in *Overview of HPS Memory-Mapped Interfaces*.



Relative Latencies and Throughputs for Each HPS Interface

This table shows usages, relative latencies, and throughputs for each interface.

Interface	Transaction Use Case	Latency	Throughput	Configuration recommended?
HPS-to-FPGA	MPU accessing memory in FPGA	Medium	Medium	Yes
HPS-to-FPGA	MPU accessing peripheral in FPGA	Medium	Very Low	No—see GUIDELINE: Avoid using the HPS-to-FPGA bridge to access peripheral registers in the FPGA from the MPU. on page 15
Lightweight HPS-to-FPGA	MPU accessing register in FPGA	Low	Low	Yes
Lightweight HPS-to-FPGA	MPU accessing memory in FPGA	Low	Very Low	No—see GUIDELINE: Avoid using the lightweight HPS-to-FPGA bridge to access memory in the FPGA from the MPU. on page 15
FPGA-to-HPS	FPGA master accessing non-cache coherent SDRAM	High	Medium	No—see GUIDELINE: Avoid using the FPGA-to-HPS bridge to access non-cache coherent SDRAM from soft logic in the FPGA. on page 15
FPGA-to-HPS	FPGA master accessing HPS on-chip RAM	Low	High	Yes
FPGA-to-HPS	FPGA master accessing HPS peripheral	Low	Low	Yes
FPGA-to-HPS	FPGA master accessing coherent memory resulting in cache miss	High	Medium	Yes

continued...

Interface	Transaction Use Case	Latency	Throughput	Configuration recommended?
FPGA-to-HPS	FPGA master accessing coherent memory resulting in cache hit	Low	Medium-High	Yes
FPGA-to-SDRAM	FPGA master accessing SDRAM through single FPGA-to-SDRAM port	Medium	High	Yes
FPGA-to-SDRAM	FPGA masters accessing SDRAM through multiple FPGA-to-SDRAM ports	Medium	Very High	Yes

GUIDELINE: Avoid using the HPS-to-FPGA bridge to access peripheral registers in the FPGA from the MPU.

The HPS-to-FPGA bridge is optimized for bursting traffic and peripheral accesses are typically short word sized accesses of only one beat. As a result if peripherals are accessed through the HPS-to-FPGA bridge the transaction can be stalled by other bursting traffic that is already in flight.

GUIDELINE: Avoid using the lightweight HPS-to-FPGA bridge to access memory in the FPGA from the MPU.

The lightweight HPS-to-FPGA bridge is optimized for non-bursting traffic and typically memory accesses are performed as bursts (often 32 bytes due to cache operations). As a result, if memory is accessed through the lightweight HPS-to-FPGA bridge, the throughput is limited.

GUIDELINE: Avoid using the FPGA-to-HPS bridge to access non-cache coherent SDRAM from soft logic in the FPGA.

The FPGA-to-HPS bridge is optimized for accessing non-SDRAM accesses (peripherals, on-chip RAM, ACP). As a result, accessing SDRAM directly by performing non-coherent accesses increases the latency and limits the throughput compared to accesses to FPGA-to-SDRAM ports.

GUIDELINE: Use soft logic in the FPGA (e.g. a DMA controller) to move shared data between the HPS and FPGA. Avoid using the MPU and the HPS DMA controller for this use case.

When moving shared data between the HPS and FPGA it is recommended to do so from the FPGA instead of moving the data using the MPU or HPS DMA controller. If the FPGA must access cache coherent data then it must access the FPGA-to-HPS bridge with the appropriate signaling to issue a cacheable transaction. If non-cache coherent data must be moved to the FPGA or HPS, a DMA engine implemented in FPGA logic can move the data, achieving the highest throughput possible. Even though the HPS includes a DMA engine internally that could move data between the HPS and FPGA, its purpose is to assist peripherals that do not master memory or provide memory to memory data movements on behalf of the MPU.

GUIDELINE: Use the HPS-to-FPGA Bridges Design Example as a starting point for designs that need to move data through the FPGA-to-HPS bridge or FPGA-to-SDRAM port.

If you own the Arria 10 SoC Development Kit, Intel offers a design example that moves data through the FPGA-to-HPS bridge and FPGA-to-SDRAM ports. The design measures the throughput of each interface under different burst sizes so that you can see how the memory bandwidth varies under different conditions.

Related Links

- HPS-to-FPGA Bridges Design Example
  Design example demonstrating the memory mapped interfaces of the hard processor system (HPS)
- Overview of HPS Memory-Mapped Interfaces on page 10
  For figure abbreviations, refer to the key in the "Arria 10 HPS Connectivity" figure.

2.2 Recommended System Topologies

Selecting the right system topology can help your design achieve the highest throughput possible. For optimum performance, observe Intel's topology guidelines moving data between the HPS and FPGA. These guidelines cover both cache coherent and non-cache coherent data movements.

2.2.1 HPS Accesses to FPGA Fabric

There are two bridges available for masters in the HPS to access the FPGA fabric. Each bridge is optimized for specific traffic patterns and as a result you should determine which is applicable to your system if an HPS master needs to access the FPGA fabric.

GUIDELINE: Connect the HPS to soft logic peripherals in the FPGA through the lightweight HPS-to-FPGA bridge.

If your hardware design has peripherals that are accessible to the HPS then you should connect them to the lightweight HPS-to-FPGA bridge. Peripherals are typically accessed by the HPS MPU one register at a time using strongly ordered (non-posted) accesses. Since the accesses are strongly ordered, the transaction from the MPU does not complete until the response from the slave returns. As a result, strongly ordered accesses are latency sensitive so the lightweight HPS-to-FPGA bridge is included in the HPS to reduce the latency of strongly ordered accesses.

GUIDELINE: Connect the HPS to FPGA memory through the HPS-to-FPGA bridge.

If your hardware design has memory that is accessible to the HPS then you should connect it to the HPS-to-FPGA bridge. Unlike the lightweight HPS-to-FPGA bridge, the HPS-to-FPGA bridge is intended for bursting traffic such as DMA transfers or MPU software execution from FPGA memory.

GUIDELINE: If the HPS must access both memory and peripherals in your FPGA logic, use HPS-to-FPGA and also lightweight HPS-to-FPGA bridges.

It is important to include both HPS-to-FPGA and lightweight HPS-to-FPGA bridges in your design if the FPGA logic contains a mix of memory and peripherals accessible to the HPS. Since peripheral accesses are typically latency-sensitive, using the

lightweight HPS-to-FPGA bridge for those accesses prevents starvation when other bursting accesses to the FPGA fabric are made through the HPS-to-FPGA bridge. Both bridges can be accessed in parallel if there are multiple HPS masters accessing the FPGA fabric at the same time so including both bridges can also improve the performance of the system.

2.2.2 Maintaining Cache Coherency

Cache coherency is a fundamental topic to understand any time data must be shared amongst multiple masters in a system. In the context of a SoC device these masters can be the MPU, DMA, peripherals with master interfaces, and masters in the FPGA connected to the HPS. Since the MPU contains level 1 and level 2 cache controllers it can hold more up-to-date contents than main memory in the system. The HPS supports two mechanisms to make sure masters in the system observe a coherent view of memory: ensuring main memory contains the latest value, or have masters access the ACP slave of the HPS.

The MPU can allocate buffers to be non-cacheable which ensures data is never cached by the L1 and L2 caches. The MPU can also access cacheable data and either flush it to main memory or copy it to a non-cacheable buffer before other masters attempt to access the data. Operating systems typically provide mechanisms for maintaining cache coherency both ways described above.

Masters in the system access coherent data by either relying on the MPU to place data into main memory instead of having it cached, or by having the master in the system perform a cacheable access via the ACP slave. Which mechanism you use typically depends on the size of the buffer of memory the master is accessing.

GUIDELINE: Ensure that data accessed through the ACP slave fits in the 512 KB L2 cache to avoid thrashing overhead.

Since the L2 cache is 512 KB in size, if a master in the system frequently accesses buffers whose total size exceeds 512 KB, thrashing results.

Cache thrashing is a situation where the size of the data exceeds the size of the cache, causing the cache to perform frequent evictions and prefetches to main memory. Thrashing negates the performance benefits of caching the data.

In potential thrashing situation, it makes more sense to have the masters access non-cache coherent data and allow software executing on the MPU maintain the data coherency throughout the system.

GUIDELINE: For small buffers of data shared between the MPU and system masters, consider having the system master perform cacheable accesses to avoid overhead caused by cache flushing operations.

If a master in the system requires access to smaller coherent blocks of data then you should consider having the MPU access the buffer as cacheable memory and the master in the system perform cacheable accesses to the data. Cacheable accesses are automatically routed to the MPU ACP slave ensuring that the master and MPU access the same copy of the data. By having the MPU use cacheable buffers and the system master performing cacheable accesses, software does not have to maintain system wide coherency ensuring both the MPU and system master observe the same copy of data.

2.2.3 MPU Sharing Data with FPGA

You can optimize data throughput by selecting the correct method of sharing data between the HPS and the FPGA. This section assumes that the HPS SDRAM is the data source and the FPGA requires access to it. There are three main ways for the FPGA to access data that originates in HPS SDRAM:

- FPGA accesses data directly through FPGA-to-SDRAM ports
- FPGA accesses data directly through FPGA-to-HPS bridge
- FPGA accesses copy of data moved to the FPGA via the HPS DMA (not recommended)

If the data in the SDRAM is the most recent copy of the data (software managed coherency) then the highest throughput method of accessing the data is to have masters in the FPGA access the data directly through the FPGA-to-SDRAM ports.

If the data in the SDRAM potentially is not the most recent copy of the data and software does not flush the MPU caches to ensure system wide coherency is maintained, then the FPGA master should perform cacheable transactions to the FPGA-to-HPS bridge to ensure the most recent data is accessed.

GUIDELINE: Avoid using the HPS DMA controller to move data between the FPGA and HPS. Use a soft DMA controller in the FPGA fabric instead. Use the HPS DMA controller only for memory copies or peripheral data movements that remain inside the HPS.

It is not recommended to use the HPS DMA to move the data to the FPGA because the DMA bandwidth into the HPS SDRAM is limited. The HPS DMA is intended to be used to move buffers on behalf of the MPU or used for transfers between peripherals and memory. As a result, any time the FPGA needs access to buffers in HPS memory, or if the HPS requires access to data stored in the FPGA, it is always recommended to have masters in the FPGA perform these transfers instead of the HPS initiating them.

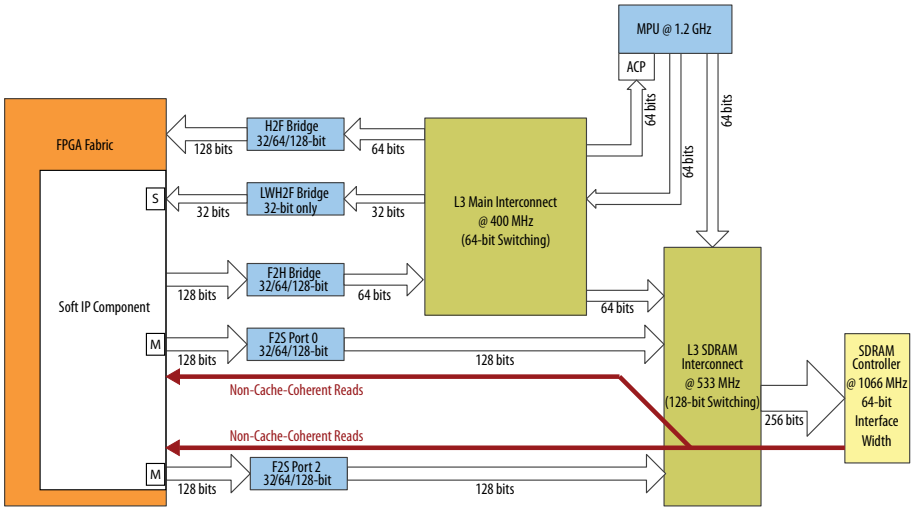
2.2.4 Examples of Cacheable and Non-Cacheable Data Accesses From the FPGA

2.2.4.1 Example 1: FPGA Reading Data from HPS SDRAM Directly

In this example the FPGA requires access to data that is stored in the HPS SDRAM. For the FPGA to access the same copy of the data as the MPU has access to, the L1 data cache and L2 cache need to be flushed if they already have a copy of the data. Once the HPS SDRAM contains the most up-to-date copy of the data, the optimal path for the FPGA to access this data is for FPGA masters to read the data through a FPGA-to-SDRAM port.

Figure 3. FPGA Reading Data from HPS FPGA-to-SDRAM Ports

For abbreviations, refer to the figure in *Overview of HPS Memory-Mapped Interfaces*.



Since the Arria 10 HPS offers two 128-bit ports into the SDRAM you can maximize the read throughput by implementing two masters in the FPGA accessing data in the SDRAM through both ports. If you decide to implement multiple paths into the SDRAM through the FPGA-to-SDRAM ports ensure that you handle synchronization at a system level since each port is serviced independently from the other. If one port should have a higher priority than the other, then you can adjust the QoS settings for each port shaping the traffic patterns as needed by your application. It is recommended to use a burst capable master in the FPGA to read from the FPGA-to-SDRAM ports, capable of posting burst lengths of four beats or larger.

Related Links

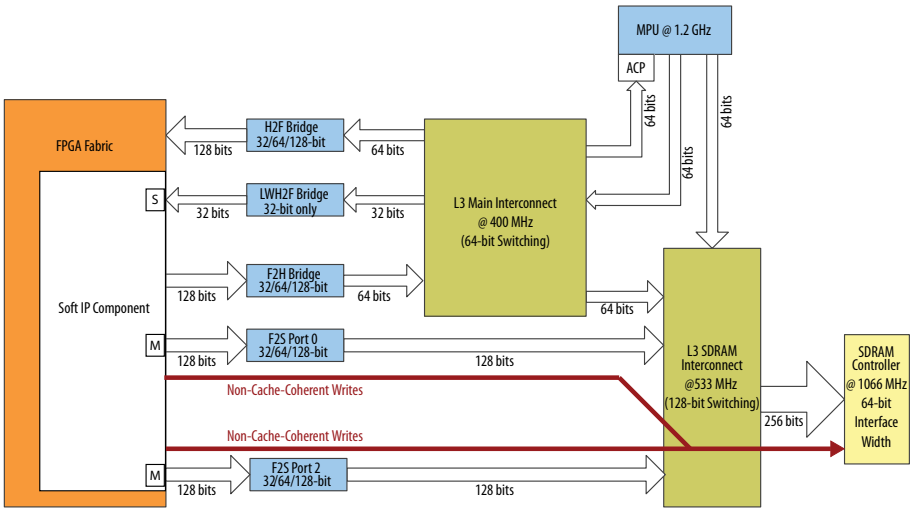
Overview of HPS Memory-Mapped Interfaces on page 10
For figure abbreviations, refer to the key in the "Arria 10 HPS Connectivity" figure.

2.2.4.2 Example 2: FPGA Writing Data into HPS SDRAM Directly

In this example the HPS MPU requires access to data that originates from within the FPGA. For the MPU to be able to access the data coherently after it is written before the transfer starts software may need to flush or invalidate cache lines, to ensure that the SDRAM contains the latest data after it is written. Failing to perform cache operations can cause one or more cache lines to eventually become evicted overwriting the data that was written by the FPGA master.

Figure 4. FPGA Writing Data to HPS FPGA-to-SDRAM Ports

For abbreviations, refer to the figure in *Overview of HPS Memory-Mapped Interfaces*.



Like in Example 1, where the FPGA reads data from the FPGA-to-SDRAM ports, you can maximize write throughput into the HPS SDRAM by using two 128-bit FPGA-to-SDRAM ports with at least one master in the FPGA connected to each port.

Related Links

- Example 1: FPGA Reading Data from HPS SDRAM Directly on page 18
- Overview of HPS Memory-Mapped Interfaces on page 10
  For figure abbreviations, refer to the key in the "Arria 10 HPS Connectivity" figure.

2.2.4.3 Example 3: FPGA Reading Cache Coherent Data from HPS

In this example the FPGA requires access to data originating in the HPS. The MPU in the HPS recently accessed this data so there is a chance that the data is still contained in the cache and therefore it may be optimal for the FPGA to access the cached data. To avoid the overhead of software having to flush dirty cache lines the FPGA can perform cache coherent reads to the FPGA-to-HPS bridge. It is important that the buffers being read be relatively small in size. Otherwise the L2 cache might thrash reading data from SDRAM for the majority of the transfer. For large buffer transfers it is more appropriate to have the FPGA read data from the FPGA-to-SDRAM ports directly as shown in Example 1.

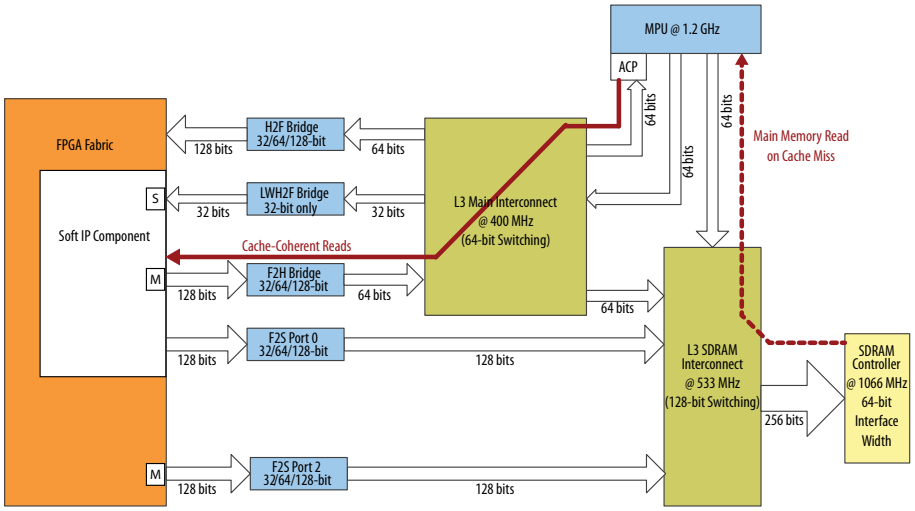
GUIDELINE: Perform full accesses targeting FPGA-to-HPS bridge.

For the transaction to be cacheable, the FPGA master must read from the FPGA-to-HPS bridge and at a minimum set the cacheable and read-allocate bits of the ARCACHE signal. If you use Avalon-MM masters to access cacheable data, you must

provide logic to force the ARCACHE signal to the appropriate values. An example of forcing Avalon-MM transactions to be cacheable can be found in the FPGA-to-HPS Bridge design example.

Figure 5. FPGA Reading Cache Coherent Data

For abbreviations, refer to the figure in *Overview of HPS Memory-Mapped Interfaces*.



GUIDELINE: Perform cacheable accesses aligned to 32 bytes targeting the FPGA-to-HPS bridge.

The ACP slave of the HPS is optimized for transactions that are the same size as the cache line (32 bytes). As a result you should attempt to align the data to 32 byte boundaries and ensure after data width adaptation the burst length into the 64-bit ACP slave is four beats long. For example, if the FPGA-to-HPS bridge is set up for 128-bit transactions you should align the data to be 32 byte aligned and perform full 128-bit accesses with a burst length of 2.

GUIDELINE: Access 32 bytes per cacheable transaction.

Ensure that each burst transaction accesses 32 bytes. Each transaction must start on a 32-byte boundary.

Table 6. Burst Lengths for 32-Byte Alignment

Bridge Width (Bits)	Access Size (Bytes)	Burst Length
32	4	8
64	8	4
128	16	2

Related Links

- HPS-to-FPGA Bridges Design Example
  This design example forces Avalon-MM transactions to be cacheable.
- Example 1: FPGA Reading Data from HPS SDRAM Directly on page 18
- Overview of HPS Memory-Mapped Interfaces on page 10
  For figure abbreviations, refer to the key in the "Arria 10 HPS Connectivity" figure.

2.2.4.4 Example 4: FPGA Writing Cache Coherent Data to HPS

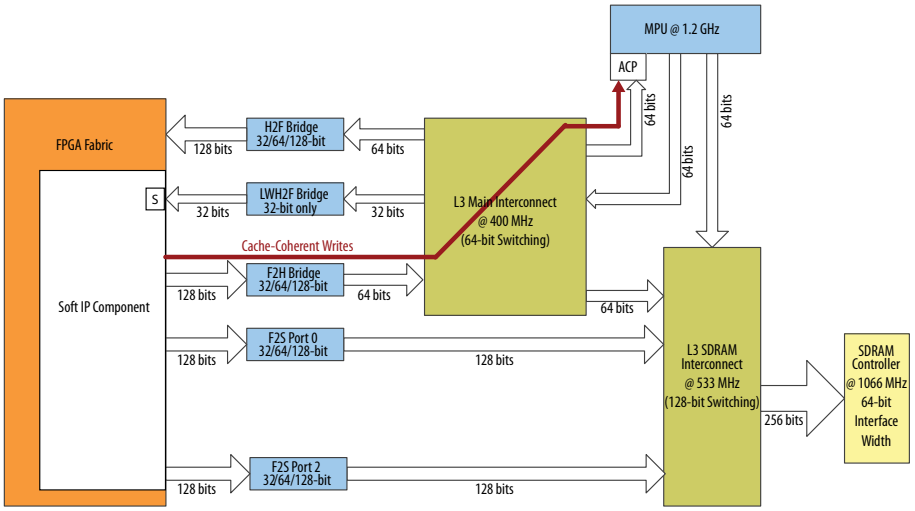
In this example the HPS MPU requires access to data that originates in the FPGA. The most efficient mechanism for sharing small blocks of data with the MPU is to have logic in the FPGA perform cacheable writes to the HPS. It is important that the amount of data to be written to the HPS be in the form of relatively small blocks because large block writes cause the L2 cache to thrash, causing the cache to write to SDRAM for the majority of the transfer. For large buffer transfers it is more appropriate to have the FPGA write data to the FPGA-to-SDRAM ports directly as shown in Example 2.

GUIDELINE: Perform full accesses targeting FPGA-to-HPS bridge.

For the transaction to be cacheable, the FPGA master must write to the FPGA-to-HPS bridge and at a minimum set the bufferable, cacheable and write-allocate bits of the AWCACHE signal. If you use Avalon-MM masters to access cacheable data, you must provide logic to force the AWCACHE signal to the appropriate values. An example of forcing Avalon-MM transactions to be cacheable can be found in the FPGA-to-HPS Bridge design example.

Figure 6. FPGA Writing Cache Coherent Data

For abbreviations, refer to the figure in *Overview of HPS Memory-Mapped Interfaces*.



GUIDELINE: Perform cacheable accesses aligned to 32 bytes targeting the FPGA-to-HPS bridge.

The ACP slave of the HPS is optimized for transactions that are the same size as the cache line (32 bytes). As a result you should attempt to align the data to 32 byte boundaries and ensure after data width adaptation the burst length into the 64-bit ACP slave is four beats long. For example, if the FPGA-to-HPS bridge is set up for 128-bit transactions you should align the data to be 32 byte aligned and perform full 128-bit accesses with a burst length of 2.

GUIDELINE: When L2 ECC is enabled, ensure that cacheable accesses to the FPGA-to-HPS bridge are aligned to 8-byte boundaries.

If you enable error checking and correction (ECC) in the L2 cache you must also ensure each 8-byte group of data is completely written. The L2 cache performs ECC operations on 64-bit boundaries so when performing cacheable accesses you must always align the access to 8-byte boundaries and write to all eight lanes at once. Failing to follow these rules results in double bit errors, which cannot be recovered.

Regardless whether ECC is enabled or disabled, 32-byte cache transactions result in the best performance. Refer to "GUIDELINE: Access 32 bytes per cacheable transaction." in Example 3 for more information about 32-byte cache transactions.

GUIDELINE: When L2 ECC is enabled, ensure that cacheable accesses to the FPGA-to-HPS bridge have groups of eight write strobes enabled.

- For 32-bit FPGA-to-HPS accesses, burst length must be 2, 4, 8, or 16 with all write byte strobes enabled.
- For 64-bit FPGA-to-HPS accesses, all write byte strobes must be enabled.
- For 128-bit FPGA-to-HPS accesses, the upper eight or lower eight (or both) write byte strobes must be enabled.

Related Links

- HPS-to-FPGA Bridges Design Example
  This design example forces Avalon-MM transactions to be cacheable.
- Example 2: FPGA Writing Data into HPS SDRAM Directly on page 19
- Overview of HPS Memory-Mapped Interfaces on page 10
  For figure abbreviations, refer to the key in the "Arria 10 HPS Connectivity" figure.
- Example 3: FPGA Reading Cache Coherent Data from HPS on page 20
  "GUIDELINE: Access 32 bytes per cacheable transaction." discusses 32-byte cacheable transactions

3 Design Guidelines for HPS Portion of Arria 10 SoC FPGAs

3.1 Start your SoC FPGA design here

3.1.1 Recommended Starting Point for HPS-to-FPGA Interface Designs

Depending on your topology, you can choose one of the two hardware reference designs as a starting point for your hardware design.

GUIDELINE: Intel recommends that you start with the Golden Hardware Reference Design (GHRD) as an example of interfacing the HPS to soft IP in the FPGA.

The Golden Hardware Reference Design (GHRD) has the optimum default settings and timing that you can use as a basis of your "getting started" system. After initial evaluation, you can move on to the Arria 10 HPS-to-FPGA Bridge Design Example reference design to compare performance among the various FPGA-HPS interfaces.

Refer to Golden Hardware Reference Design (GHRD) on page 61 for more information.

GUIDELINE: Intel recommends that you use the Arria 10 HPS-to-FPGA Bridge Design Example reference design to determine your optimum burst length and data-width for accesses between FPGA logic and HPS.

The Arria 10 FPGA-to-HPS bridge design example contains modular SGDMAs in the FPGA logic that allow you to program the burst length for data accesses from the FPGA logic to the HPS.

3.1.2 Determining your SoC FPGA Topology

To determine which system topology best suits your application, you must first determine how to best partition your application into hardware and software.

GUIDELINE: Profile your software using any profiling tool (e.g. DS-5 streamline profiler) to help you identify functions that are good candidates for hardware acceleration and isolate those functions that are best implemented in software.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2008 Registered

3.2 Design Considerations for Connecting Device I/O to HPS Peripherals and Memory

One of the most important considerations when configuring the HPS is to understand how the I/O is organized in the Arria 10 SoC devices.

1. HPS Dedicated I/O

These 17 I/O are physically located inside the HPS, are dedicated for the HPS, and are primarily used for the HPS boot flash, clock, and resets. All other I/O are located in I/O columns in the FPGA logic.

2. HPS Shared I/O (shared with FPGA)

There is a single I/O bank of 48 pins (shared I/O) available for either HPS peripheral signals or FPGA signals.

3. HPS External Memory Interface I/O (shared with FPGA)

Depending on the device and package you have selected, there are two or three modular I/O banks that can connect to SDRAM memory. One of the I/O banks is used to connect the address, command and ECC data signals. The other one or two banks are for connecting the data signals.

4. FPGA I/O

You can use general purpose I/O for FPGA logic, FPGA external memory interfaces and high speed serial interfaces.

The table below summarizes the characteristics of each I/O type.

Table 7. Summary of SoC-FPGA I/O Types

	Dedicated HPS I/O	HPS External Memory Interface I/O	Shared HPS-FPGA IO48	FPGA I/O
Number of Available I/O	17	Up to three IO48 banks	48	All other device I/O
Location	Dedicated bank of I/Os in the HPS block	I/O Banks (all banks are in the same column) 2I, 2J, 2K (adjacent to HPS)	FPGA I/O bank 2L (adjacent to HPS block)	FPGA I/O banks
Voltages Supported	1.8V, 2.5V and 3.0V	LVDS I/O in support of DDR3 and DDR4 protocols	3V I/O buffer type voltage support[1]	LVDS I/O, 3V I/O and high speed serial I/O (HSIO) buffer types voltage support[2]
Purpose	Clock, Reset, Boot Source and UART	HPS main memory	High speed HPS peripherals	General purpose and transceiver I/O
Timing Constraints	Fixed	Provided by memory controller IP [3]	Fixed for legal combinations [3]	User defined
Recommended Peripherals	QSPI, NANDx8, eMMC, SD/MMC card, and UART	DDR3 and DDR4 SDRAM	EMAC, USB (Refer to HPS Qsys Component for legal combinations)	Slow speed peripherals (I^2C, SPI, EMAC-MII)

Related Links

I/O Standards Support for FPGA I/O in Arria 10 Devices

For a list of supported I/O standards and voltage levels for 3V I/O, LVDS I/O, and HSIO, see the "Supported I/O Standards in FPGA I/O for Arria 10 Devices" table in the *Intel Arria 10 Core Fabric and General Purpose I/Os Handbook*.

3.2.1 HPS Pin Multiplexing Design Considerations

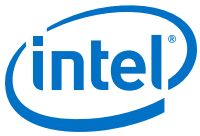
Because the HPS peripheral signals total more than the Shared I/O bank of 48 pins, the HPS component in Qsys offers pin multiplexing settings as well as the option to route most of the peripherals into the FPGA fabric. Any unused pins in the Shared I/O bank can be used as general purpose I/O by the FPGA in groups of 12 pins. The HPS Shared I/O resides in 3V I/O Bank 2L in the FPGA I/O column adjacent to the HPS and generally supports the full feature set of a 3V I/O bank. However, any use of the Shared I/O for HPS peripherals limits voltage level support to either 1.8V, 2.5V or 3V operation, subject to the same rules for I/O Standard compatibility and support as any other FPGA I/O bank. All peripherals connected to the I/O bank must support the selected voltage.

---

1 For details of voltage level support, see the *Intel Arria 10 Core Fabric and General Purpose I/Os Handbook*. HPS peripherals using shared I/O have only been characterized for 3.0V, 2.5V and 1.8V LVTTL/LVCMOS operation. All FPGA interfaces assigned to shared I/O must the compatible with HPS peripheral I/O in the same bank.

2 For details of voltage level support, see the *Intel Arria 10 Core Fabric and General Purpose I/Os Handbook*.

3 *Note:* You can access the timing information to perform off-chip analysis by reviewing the HPS timing in the Arria 10 Device Datasheet.



GUIDELINE: You must route the USB, EMAC and Flash interfaces to the HPS Dedicated and Shared I/O first, starting with USB.

It is recommended that you start by routing high speed interfaces such as USB, Ethernet, and flash to the Dedicated and Shared I/O first. You must route USB signals to shared I/O because it is not available to the FPGA fabric. The flash boot source must be routed to the HPS dedicated I/O because these are the only I/O that are functional before the shared and FPGA I/O have been configured.

GUIDELINE: Ensure that you reserve an entire quadrant of shared I/O if needed for FPGA usage.

The Shared I/O can be used by designs residing in the FPGA fabric, but the I/O must be made available on a quadrant basis (groups of 12 pins). The Shared I/O is divided into four quadrants with each quadrant being routed to either the HPS peripherals or FPGA logic, but not both. As a result, if you wish to use HPS Shared I/O for FPGA, you must reserve one of the quadrants and route the HPS peripherals to the remaining three quadrants of the shared I/O bank.

Refer to the ***Peripheral Pin Multiplexing tab*** in the ***HPS configuration dialog box* in Qsys** when selecting I/O Sets for peripherals. Pay attention to errors that may occur when adding peripherals. The console box at the bottom of the HPS configuration dialog box aids in resolving conflicts.

3.2.2 HPS I/O Settings: Constraints and Drive Strengths

GUIDELINE: Ensure that you have correctly configured the I/O settings for the HPS Shared and Dedicated I/O.

The HPS pin location assignments are managed automatically when you generate the Qsys system containing the HPS. Likewise, timing and I/O constraints for the HPS SDRAM controller are managed by the Arria 10 External Memory Interfaces for HPS. The only HPS I/O constraints you must manage are for the Dedicated and Shared I/O. Constraints such as drive strength, I/O standards, and weak pull-up enables are added to the Quartus project just like FPGA constraints; and are applied to the HPS at boot time when the second stage bootloader configures the I/O. I/O constraints for dedicated I/O are stored in the device tree for the boot loader software. For shared and FPGA I/O, the I/O constraints are applied to the FPGA configuration file.

3.3 HPS Clocking and Reset Design Considerations

The main clock and resets for the HPS are HPS_CLK1, HPS_nPOR and HPS_nRST. HPS_CLK1 (also referred to as EOSC1) is the external clock source for the HPS PLLs, which generate clocks for the MPU Subsystem, L3 Interconnect, HPS peripherals and HPS-to-FPGA user clocks. HPS_nPOR provides a cold reset input, and HPS_nRST provides a bidirectional warm reset resource.

This section supplements the following sections of the Arria 10 Device Design Guidelines document: "Pin Connection Considerations for Board Design" and "I/O and Clock Planning."

3.3.1 HPS Clock Planning

HPS clock planning involves choosing clock sources and defining frequencies of operation for the following HPS components:

- HPS PLLs
- MPU Subsystem
- L3 Interconnect
- HPS Peripherals
- HPS-to-FPGA user clocks

HPS clock planning is interdependent on system-level design planning in the areas of board-level clock planning, clock planning for the FPGA portion of the device, and HPS peripheral external interfacing. Therefore, it is important to validate your HPS clock configuration before finalizing your board design.

GUIDELINE: Verify the MPU and peripheral clocks using Qsys.

Use Qsys to initially define your HPS component configuration. Set the HPS input clocks, peripheral source clocks and frequencies. Note any Qsys warning or error messages and address them by modifying clock settings or verifying that a particular warning does not adversely affect your application.

3.3.2 Early Pin Planning and I/O Assignment Analysis

HPS clock and reset I/O reside in the HPS Dedicated I/O Bank on fixed pin locations and typically share the bank with HPS peripherals such as boot flash and UART console.

GUIDELINE: Choose an I/O voltage level for the HPS Dedicated I/O.

HPS_CLK1, HPS_nPOR and HPS_nRST are located in the HPS Dedicated I/O bank on fixed pin locations. The HPS Dedicated I/Os are LVCMOS/LVTTL supporting 1.8V, 2.5V and 3V voltage levels. The I/O signaling voltage for this bank is established by the supply level applied to VCCIO_HPS, which can be either 1.8V, 2.5V or 3V. Make sure the supply level chosen for VCCIO_HPS is compatible with any HPS peripheral interfaces (e.g. boot source, UART console) configured to use the HPS Dedicated I/O bank as well as board-level clock and reset circuitry for the HPS.

3.3.3 Pin Features and Connections for HPS Clocks, Reset and PoR

The HPS external reset I/O pins have certain functional behaviors and requirements that you should consider when planning for and designing your board-level reset logic and circuitry.

GUIDELINE: Use the HPS_nRST pin to initiate a warm reset of the HPS or to drive a reset externally on the board.

HPS_nRST is an active low, open-drain-type, bidirectional I/O. Externally asserting a logic low to the HPS_nRST pin initiates a warm reset of the HPS. HPS warm and cold reset can also be asserted from internal sources such as software-initiated resets and reset requests from the FPGA fabric. When the HPS is internally placed under warm or cold reset, the HPS component becomes a reset source and drives the HPS_nRST pin low, resetting any connected board-level components. Externally asserting the HPS_nPOR pin also results in the HPS asserting reset on the HPS_nRST pin.

GUIDELINE: Observe the minimum assertion time specifications of HPS_nPOR and HPS_nRST.

Reset signals on the HPS_nPOR and HPS_nRST pins must be asserted for a minimum number of HPS_CLK1 cycles as specified in the HPS section of the Arria 10 Datasheet.

3.3.4 Internal Clocks

Once you have validated the HPS clock configuration as described in the HPS Clock Configuration Planning guidelines, you must implement your HPS clock settings under software control, which is typically done by the boot loader software. You must also follow guidelines for transferring reference clocks between the HPS and FPGA.

GUIDELINE: Avoid cascading PLLs between the HPS and FPGA.

Cascading PLLs between the FPGA and HPS has not been characterized. Unless you perform a jitter analysis, do not chain the FPGA and HPS PLLs together. Output clocks from HPS are not intended to be fed into PLLs in the FPGA.

There are specific requirements for managing for managing HPS PLLs and clocks under software control.

The boot loader software provided by SoC EDS meets all requirements for managing HPS PLLs and clocks. If you are developing your own boot loader software, see the related documentation.

Related Documentation

Refer to the Clock Manager chapter and the specific peripheral and subsystem chapters in the Arria 10 Hard Processor System Technical Reference Manual for the required software flow.

Refer to the Arria 10 SX Device Errata for the "Correct Sequence Required When Raising HPS PLL Frequency" for requirements specific to ramping HPS PLL frequencies to their final values.

3.3.5 HPS Reset during FPGA reconfiguration and FPGA configuration failures

During FPGA reconfiguration and FPGA configuration failure, the device I/O are tri-stated, and the HPS loses access to the HPS Shared I/O and HPS External Memory Interface I/O.

GUIDELINE: Ensure that software performs an integrity check of the FPGA configuration image before initiating a FPGA configuration through the HPS.

GUIDELINE: Ensure that the HPS is in reset while the FPGA is being fully configured by an external source. This will ensure the shared I/O and HPS SDRAM I/O are configured by the time those resources are needed by the HPS.

GUIDELINE: If you want to operate the HPS during reconfiguration, then design the reconfiguration bitstream as a partial reconfiguration image.

Refer to the Arria 10 SoC FPGA Partial Reconfiguration Sequence Through FPGA Manager section in the Arria 10 HPS Technical Reference Manual for further information.

3.3.6 HPS Peripheral Reset Management

HPS peripherals and subsystems have specific reset sequencing requirements. The boot loader software provided in SoC EDS implements the reset management sequence according to the requirements in the Reset Manager chapter.

GUIDELINE: Use the latest boot loader software in SoC EDS to manage HPS reset.

Refer to the Reset Manager chapter and the specific peripheral and subsystem chapters in the Arria 10 Hard Processor System Technical Reference Manual for the required software flow.

3.4 HPS EMIF Design Considerations

A critical component to the HPS is its external SDRAM memory. The following design considerations help you properly design the interface between SDRAM memory and the HPS.

When connecting external SDRAM to the HPS, refer to the following essential documentation:

Arria 10 Core Fabric and General Purpose I/O Background

The Arria 10 Core Fabric and General Purpose I/Os Handbook includes information on the Hard Memory Controller (HMC) block and hardened feature support for DDR SDRAM memories in the I/O elements. The handbook also shows the I/O column architecture, where the specific HMC block accessible to the HPS resides, and the number of supported interfaces of a given type for the available device/package combinations. The handbook is a central source of documentation for the FPGA portion of SoC devices.

For guidance on connecting the HPS-accessible hard memory controller block to the HPS, study the following sections of the handbook:

- **Chapter 5: I/O and High Speed I/O in Arria 10 Devices - GPIO Banks, SERDES and DPA Locations in Arria 10 Devices under I/O Resources in Arria 10 Devices section**

  This section shows the I/O column and bank locations for all device and package combinations across all Arria 10 family variants, including the relative location of the HPS to its accessible banks.

- **Chapter 6: External Memory Interfaces in Arria 10 Devices - Memory Interfaces Support in Arria 10 Device Packages**

  This section shows the number of supported memory types and widths supported by Arria 10 SX device/package combinations.

External Memory Interface Handbook, Volume 3: Reference Material

The External Memory Interface (EMIF) Handbook includes the details required to understand what specific I/O banks are used for HPS external memory interfaces, where address/command, ECC and data signals are located. The handbook also consists of important information on restrictions on the placement of these external memory interface signals within the banks and any flexibility the designer has in varying from the default placement. While Intel recommends that you familiarize yourself with all the content available in the three volumes that make up The EMIF Handbook, understanding the following section found in volume 3 is a prerequisite to properly design the Arria 10 EMIF for the HPS IP in your application.

- **Chapter 2: Functional Description – Arria 10 EMIF, section Arria 10 EMIF for Hard Processor Subsystem**

  This section states the specific external SDRAM memory types, speeds, widths, and interface and device configurations supported for HPS external memory interfaces in Arria 10 SX devices. A diagram is provided that shows the specific I/O bank and lane locations for address/command, ECC, and data signals. See the "**Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS**" subsection for detailed information on memory interface signal placement when varying from the Arria 10 EMIF for the HPS IP default locations.

The following design guidelines supplement the information found in the above referenced documentation.

3.4.1 Considerations for Connecting HPS to SDRAM

The hard memory controller for the Arria 10 HPS is located in the FPGA I/O columns along with the other hardware memory controllers. The HPS can use only one hard memory controller bank, and it is located closest to the HPS block in I/O bank 2K, is where the address/command and ECC signals reside. Use I/O Bank 2J for 16-bit and 32-bit interface DQ/DQS data group signals. I/O Bank 2I, available only in the KF40 package, is used for 64-bit interface DQ/DQS data group signals for wider. Bank 2I is for the upper 32-bits of 64-bit interfaces.

Instantiating the Arria 10 HPS EMIF IP

Connecting external SDRAM to the Arria 10 HPS requires the use of an EMIF IP that is specific to the HPS. Follow the below guidelines for properly instantiating and configuring the correct EMIF IP for the HPS.

GUIDELINE: Instantiate the "Arria 10 External Memory Interfaces for HPS" IP in Qsys.

You must use a specific EMIF IP in Qsys to connect the HPS to external SDRAM. This Qsys component is named "Arria 10 External Memory Interfaces for HPS" and can be found in the "IP Catalog" pane inQsys in the "Hard Processor Components" subgroup under the "Processors and Peripherals" group.

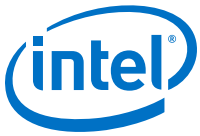
GUIDELINE: Connect the `hps_emif` conduit to the HPS component

The hard memory controller connected to the HPS has a dedicated connection that must be connected in Qsys. The Arria 10 EMIF for HPS IP component exposes this connection through a conduit called "`hps_emif`" that must be connected to the HPS component's "emif" conduit.

GUIDELINE: Make sure the HPS EMIF IP block is not reset while the HPS is accessing external SDRAM or resources in the L3 SDRAM Interconnect.

Asserting reset to the HPS EMIF IP block should coincide with the HPS reset assertion unless the application is capable of saving and recovering context in co-ordination with HPS EMIF IP reset assertion. This can be achieved simply by connecting the HPS EMIF reset input to one or a combination of resets from the following sources: HPS reset outputs (e.g. `h2f_reset`, `h2f_cold_reset`), other resets in the system that also source an HPS cold or warm reset input (e.g. HPS_nPOR, HPS_nRST, FPGA-to-HPS cold/warm reset requests).

If the HPS EMIF IP is reset without resetting the HPS as described above, the application must put the L3 SDRAM Interconnect in reset using the `brgmodrst` register, bit 6 (ddrsch) in the Reset Manager before HPS EMIF IP reset assertion and not release it until after the HPS EMIF IOPLL has locked. Failure to do so can result in locking up the processor on subsequent accesses to external SDRAM or resources in the L3 SDRAM Interconnect.



GUIDELINE: Ensure that the HPS memory controller Data Mask (DM) pins are enabled.

When you instantiate the memory controller in Qsys, you must select the checkbox to enable the data mask pins. If this control is not enabled, data corruption occurs any time a master accesses data in SDRAM that is smaller than the native word size of the memory.

GUIDELINE: Ensure that you choose only DDR3 or DDR4 components or modules in configurations that are supported by the Arria 10 EMIF for HPS IP and your specific device/package combination.

Intel's External Memory Interface Spec Estimator is a parametric tool that allows you to compare supported external memory interface types, configurations and maximum performance characteristics in Intel FPGA and SoC devices. Not all device/package combinations support 64/72-bit wide interfaces. See Chapter 6 of the **Arria 10 Core Fabric and General Purpose I/O Handbook**.

3.4.2 HPS SDRAM I/O Locations

The Arria 10 EMIF for HPS IP includes default pin location assignments for all of the external memory interface signals in constraint files created at IP generation time and read by Quartus® Prime during design compilation.

GUIDELINE: It is recommended that you use these automated default pin location assignments.

GUIDELINE: Verify the HPS memory controller I/O locations in the Quartus project pinout file in the "`output_files`" sub-folder before finalizing board layout.

By default, Quartus Prime generates output reports, log files and programming files in the "`output_files`" subfolder of the project folder. See the `.pin` text file after compilation for the pinout for your design, including the pin locations for the HPS EMIF.

GUIDELINE: When using the Early I/O Release boot flow, make sure all I/O associated with the HPS memory interface are located within the active HPS I/O banks indicated in the table below.

It is critical that you ensure all I/O necessary for a functioning HPS memory interface are located within the active banks for your HPS memory width as shown in the table below when using the Early I/O Release boot flow.

Typically Quartus Prime compilation flags an error for any HPS memory I/O that are not placed in the I/O bank and lane locations shown in the table below for a given HPS memory width.

An exception is the RZQ pin, which generally can be placed in any RZQ pin location in the I/O column. For successful HPS memory interface calibration with the Early I/O Release boot flow, the RZQ pin for the HPS memory interface must be placed in either I/O Bank 2K or 2J for memory widths up to and including 32/40-bits. For 64/72-bit wide HPS memory interfaces, RZQ must be in I/O Bank 2K, 2J or 2I.

In addition, when using the Early I/O release flow, the EMIF reference clock must be placed in bank 2K.

Refer to the table below for the I/O mapping of the HPS EMIF. I/O lanes not utilized by the HPS EMIF Controller are available to the FPGA fabric as either General Purpose Inputs-only (GPI) or General Purpose I/O (GPIO).

Table 8. HPS SDRAM I/O Locations

EMIF Width	Bank 2K Lanes				Bank 2J Lanes				Bank 2I Lanes			
	3	2	1	0	3	2	1	0	3	2	1	0
16-bit	GPI	Address/Command			GPIO		Data [15:0]		GPIO	GPIO	GPIO	GPIO
16-bit + ECC	ECC	Address/Command			GPIO		Data [15:0]		GPIO	GPIO	GPIO	GPIO
32-bit	GPI	Address/Command			Data [31:0]				GPIO	GPIO	GPIO	GPIO
32-bit + ECC	ECC	Address/Command			Data [31:0]				GPIO	GPIO	GPIO	GPIO
64-bit	GPI	Address/Command			Data [31:0]				Data [63:32]			
64-bit + ECC	ECC	Address/Command			Data [31:0]				Data [63:32]			

GUIDELINE: When varying from the automated, default pin locations, refer to the "Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS" section of Chapter 2 of the EMIF Handbook, vol 3.

While the interface is mostly fixed to the I/O banks and lanes as shown in the above table, there is some flexibility in shuffling non-ECC DQ/DQS data group lanes and DQ signals within the fixed pin locations. Validate any non-default pin locations with a Quartus Prime compilation.

GUIDELINE: Unused pin locations within I/O lanes utilized by the Arria 10 EMIF for HPS IP are accessible by the FPGA fabric.

See the below I/O bank-specific sections for details.

3.4.2.1 I/O Bank 2K, Lanes 0,1,2 (Addr/Cmd)

The Arria 10 External Memory Interface (EMIF) for HPS IP core uses the Hard Memory Controller (HMC) located in I/O Bank 2K, which results in lanes 0, 1 and 2 being used for the address and command signals. The address and command signals are at fixed locations within these I/O lanes.

GUIDELINE: Unused pins in I/O Lanes 0, 1 and 2 of I/O Bank 2K are available as FPGA GPIO.

Pins not utilized by the Arria 10 EMIF for HPS IP core for address and command in I/O Bank 2K, lanes 0, 1 and 2 are available to the FPGA fabric as general purpose I/O. FPGA GPIO signals assigned to unused pin locations in these lanes support I/O standards compatible with I/O Bank 2K's VCCIO and VREF supply levels, which are dictated by the external SDRAM's signaling standard.

3.4.2.2 I/O Bank 2K, Lane 3 (ECC)

The Arria 10 EMIF for HPS IP core fixes the location for the ECC-related DQ/DQS data group signals in I/O Lane 3 of I/O Bank 2K.

GUIDELINE: Lane 3 of I/O Bank 2K is for the exclusive use of ECC data by the Arria 10 EMIF for HPS IP core.

If you use ECC on the HPS EMIF, the DQ/DQS data lane signals corresponding to the ECC data must be located in this specific I/O lane. If you don't use ECC, general HPS EMIF data cannot be located in this I/O lane.

GUIDELINE: Unused pins in I/O Lane 3 of I/O Bank 2K are available as FPGA GPI.

Pins not utilized by the Arria 10 EMIF for HPS IP core for ECC in I/O Lane 3 of I/O Bank 2K are available to the FPGA fabric as general purpose inputs (input-only). If your Arria 10 EMIF for HPS IP configuration does not use ECC and therefore does not use Lane 3, the unused pins are still only available as inputs-only. FPGA GPI signals assigned to unused pin locations in Lane 3 support I/O standards compatible with I/O Bank 2K's VCCIO and VREF supply levels, which are dictated by the external SDRAM's signaling standard.

3.4.2.3 I/O Bank, 2J (Data)

The Arria 10 EMIF for HPS IP core uses I/O Bank 2J for all non-ECC DQ/DQS data lane signal groups for 16/24/32/40-bit interfaces. For 64/72-bit interfaces, the lower four non-ECC DQ/DQS data lane signal groups are located in this bank.

GUIDELINE: I/O pins in lanes NOT utilized by the Arria 10 EMIF for HPS IP are available as FPGA GPIO.

For 16/24-bit interfaces, the Arria 10 EMIF for HPS IP utilizes two of the I/O lanes in I/O Bank 2J for the non-ECC DQ/DQS data lane signals. The other two I/O lanes in I/O Bank 2J are available to the FPGA fabric as general purpose I/O. FPGA GPIO signals assigned to pins in unused lanes support I/O standards compatible with I/O Bank 2J's VCCIO and VREF supply levels, which are dictated by the external SDRAM's signaling standard.

GUIDELINE: Unused pins in lanes that ARE utilized by the Arria 10 EMIF for HPS IP are available as FPGA GPI.

In I/O lanes utilized by the Arria 10 EMIF for HPS IP core for DQ/DQS data lane signals in I/O Bank 2J, any unused pins are available to the FPGA fabric as general purpose inputs-only. FPGA GPI signals assigned to unused pin locations in these utilized lanes support I/O standards compatible with I/O Bank 2J's VCCIO and VREF supply levels, which are dictated by the external SDRAM's signaling standard.

3.4.2.4 I/O Bank, 2I (Data, 64/72-bit interfaces)

The Arria 10 EMIF for HPS IP core uses I/O Bank 2I for the upper four non-ECC DQ/DQS data lane signal groups for 64/72-bit interfaces. For interfaces narrower than 64-bits, I/O Bank 2I is not utilized.

GUIDELINE: I/O pins in lanes NOT utilized by the Arria 10 EMIF for HPS IP are available as FPGA GPIO.

For 16/24/32/40-bit interfaces, the Arria 10 EMIF for HPS IP does not utilize I/O Bank 2I, which leaves the entire bank available to the FPGA fabric as general purpose I/O with no restrictions placed on available I/O standards from the HPS EMIF external SDRAM signaling standard. The usual rules apply.

GUIDELINE: Unused pins in lanes that ARE utilized by the Arria 10 EMIF for HPS IP are available as FPGA GPI.

For 64/72-bit interfaces, all I/O lanes in I/O Bank 2I are utilized, and any unused pins are available to the FPGA fabric as general purpose inputs-only. FPGA GPI signals assigned to unused pin locations in these utilized lanes support I/O standards compatible with I/O Bank 2I's VCCIO and VREF supply levels, which are dictated by the external SDRAM's signaling standard.

3.4.3 Integrating the Arria 10 HPS EMIF with the SoC FPGA Device

Consider the following when integrating the Arria 10 EMIF for HPS IP core with the rest of the SoC system design.

GUIDELINE: Follow the guidelines for optimizing bandwidth for all masters accessing the HPS SDRAM.

Accesses to SDRAM connected to the HPS EMIF go through the HPS SDRAM L3 Interconnect. When designing and configuring high bandwidth DMA masters and related buffering in the FPGA core, refer to the DMA Considerations on page 37. The principles covered in that section apply to all high bandwidth DMA masters (e.g. Qsys DMA Controller components, integrated DMA controllers in custom peripherals) and related buffering in the FPGA core that access HPS resources (e.g. HPS SDRAM) through the FPGA-to-SDRAM and FPGA-to-HPS bridge ports, not just tightly-coupled HPS hardware accelerators.

GUIDELINE: Instances of the Arria 10 EMIF IP (non-HPS version) cannot be located in the same I/O column as the Arria 10 EMIF for HPS IP.

Arria 10 SoC devices have two I/O columns. The Arria 10 EMIF for HPS IP must be located in the column that contains I/O Bank 2K. When your design uses the Arria 10 EMIF for HPS IP, locate other non-HPS Arria 10 EMIF IP instances in the other column. If your design does not instance the Arria 10 EMIF for HPS IP, you can place non-HPS Arria 10 EMIF IP in either column. PHYLite IP instances can be located in the same column as the Arria 10 EMIF for HPS IP. The Quartus Prime software reports an error if an Arria 10 non-HPS EMIF is in the same I/O column as the Arria 10 EMIF for HPS.

3.4.4 HPS Memory Debug

GUIDELINE: Verify the memory interface is operational using an FPGA EMIF and the external memory tool kit.

Because the HPS SDRAM controller does not support the external memory interface toolkit, it is recommended that you verify that the memory interface is operational using the non-HPS memory controller first. Create a design that instantiates the FPGA memory controller and routes it to the same I/O that the HPS memory controller uses. Once you have verified that the interface is operational with the EMIF toolkit, ensure that you properly instantiate the Arria 10 External Memory Interfaces for HPS IP as described in the sub-section on Instantiating the Arria 10 HPS EMIF IP described in section Considerations for Connecting HPS to SDRAM on page 32.

Refer to External Memory Interfaces in Arria 10 Devices and External Memory Interface Handbook and Arria 10 External Memory Interface Pin Interface for additional information.

3.5 DMA Considerations

3.5.1 Choosing a DMA Controller

Choose the DMA implementation best suited to your design

When a DMA controller is required to improve system performance, you have the option to use the DMA controller integrated into the HPS or a soft DMA module in the FPGA. When making the choice of which option to use, you should consider the following:

- HPS DMA: primarily used to move data to and from other slow-speed HPS modules, such as SPI and I^2C, as well as to perform internal memory copies on behalf of software.
- Soft DMAs: primarily used to move data between the FPGA and HPS.

3.5.2 Optimizing DMA Master Bandwidth through HPS Interconnect

FPGA DMA masters have access to HPS resources through the FPGA-to-HPS Bridge and FPGA-to-SDRAM ports, configurable in the HPS Qsys Component. The L3 and SDRAM L3 Interconnects in the HPS provide arbitration for these resources and enforce secure region and Quality of Service (QoS) settings. When planning for and designing DMA masters and related buffering that access resources through the HPS interconnect, study the architecture of the HPS interconnect and consider the following guidance and resources available for optimizing bandwidth through the interconnect.

GUIDELINE: Utilize the *FPGA-to-HPS Bridge Design Example* to tune for performance.

The FPGA-to-HPS Bridge Design Example is a useful platform for modeling specific data traffic access patterns between the FPGA and HPS resources. The example design includes a utility that runs on the ARM Cortex A-9 processor in the HPS that allows selecting datapaths between endpoints, transaction characteristics (e.g., burst lengths), and reporting transfer bandwidth.

4 Board Design Guidelines for Arria 10 SoC FPGAs

4.1 Power On Board Bring Up and Boot ROM/Boot Loader Debugging

On initial power on, if the device is in an unconfigured state, only the dedicated HPS I/Os are available to the processor. To have visibility into the boot process before the other device I/Os are configured, dedicated I/Os must be used to connect to the serial interface.

GUIDELINE: Ensure that you have connected a UART serial interface to the dedicated I/O in the HPS device in order to gain visibility into the boot process.

Note: When the HPS boots from NAND flash, the UART I/Os are not available from the HPS dedicated I/Os.

4.2 HPS Power Design Considerations

For design considerations and recommendations on power consumption and thermal analysis, SoC device pin connections, supply design and decoupling, refer to the Arria 10 Device Design Guidelines.

The following sections are supplemental for SoC devices.

4.2.1 Early System and Board Planning

4.2.1.1 Early Power Estimation

Follow the guidelines in the **Early Power Estimation** section of the Arria 10 Device Design Guidelines for using the PowerPlay Early Power Estimation (EPE) spreadsheets.

In addition, consider the following guidelines for Arria 10 SoC devices when using the Arria 10 EPE spreadsheet.

4.2.1.1.1 Main Worksheet

GUIDELINE: Select "Maximum" for the Power Characteristics setting.

When estimating power consumption for the purposes of designing an adequate power supply that can meet the maximum power requirements across process, voltage and temperature (PVT), use the device maximum power characteristics.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2008 Registered

4.2.1.1.2 IO Worksheet

This tab is where you describe the various configurations of I/O Elements (IOEs) in your application. Use the IO-IP tab to describe the controller IP behind each set of I/Os.

GUIDELINE: Add HPS peripherals assigned to FPGA I/O.

For HPS peripherals assigned to FPGA I/O, add rows to the spreadsheet as necessary to describe the different HPS peripheral I/O characteristics in your design with the Bank Type set to either "3VIO" or "LVDSIO" as appropriate.

GUIDELINE: Add HPS peripherals assigned to Shared I/O.

For HPS peripherals assigned to HPS Shared I/O, add rows to the spreadsheet as necessary to describe the different HPS peripheral I/O characteristics in your design with the Bank Type set to "HPS."

4.2.1.1.3 IO-IP Worksheet

Use this tab to describe the controller IP behind the I/Os described in the IO tab, including the HPS peripheral controllers.

GUIDELINE: Add the HPS peripherals configured in the Qsys HPS component for your application.

For each HPS peripheral assigned to the HPS Dedicated I/O, select the "… (Dedicated HPS)" variant for the "IP" parameter – there would be no entries in the IO tab for these peripheral I/Os. For each HPS peripheral assigned to either Shared or FPGA I/O, select the variant without the "… (Dedicated HPS)" designation for the "IP" parameter – you should have an entry in the IO tab for these peripheral I/O.

4.2.1.1.4 HPS Worksheet

GUIDELINE: Select either 900 mV or 950 mV for VCCL_HPS.

Select either the standard 900 mV level or 950 mV overdrive level in support of HPS MPU overclocking. For HPS MPU overclocking capabilities, see the Arria 10 Device Datasheet for maximum achievable MPU frequency and the associated requirements for device speed grade and HPS power supply overdrive level.

GUIDELINE: Select the Frequency, Application, and if applicable, the Application Mode for each CPU in the HPS tab of the spreadsheet.

The Application/Application Mode settings for each CPU allow you to select from a list of industry standard benchmarks to model CPU utilization in your application. You can also select "Custom" for defining a unique set of CPU utilization parameters across the ALUs and cache memories.

4.2.2 Design Considerations for HPS and FPGA Power Supplies for SoC FPGA devices

4.2.2.1 Consider Device Power Consumption and HPS Performance

Arria 10 SoC devices can operate the HPS at a faster speed by providing a higher supply voltage (0.95V vs. 0.9V) to the HPS (VCCL_HPS).

For HPS MPU overclocking capabilities, refer to the Arria 10 Device Datasheet for maximum achievable MPU frequency and the associated requirements on device speed grade and HPS supply overdrive level.

The FPGA power is derived from VCC supply and can be 0.9V or 0.95V. The FPGA and device I/O supplies contribute to the major portion of the device power consumption.

GUIDELINE: Provide the capability for the board to supply 0.95V power rail and provide separate power rails to the HPS (VCCL_HPS) and FPGA (VCC).

Separate power rails for VCC and VCCL_HPS enables the following flexibility:

- If lower power is desired, set VCC=VCCL_HPS=0.9V
- If higher performance is desired, set VCC=VCCL_HPS=0.95V
- If an optimum balance of maximum HPS performance with lower total device power consumption is desired, set VCC=0.9V, VCCL_HPS=0.95V

4.2.2.2 Consider Desired HPS Boot Clock Frequency

The Arria 10 SoC device supports a default boot clock mode of 10-50 MHz as well as faster boot clock modes.

To use faster boot clock frequencies, you can change the CSEL fuses from their default setting of 0x0 to any value between 0x7 and 0xE. These settings allow boot ROM code to configure the HPS PLL to run faster frequencies, depending on the speed of the external oscillator.

Refer to the table with CSEL Options and corresponding External Oscillator Frequency in the Clock Select section of the Arria 10 HPS TRM.

GUIDELINE: If the faster boot clock frequencies enabled with CSEL value 0x7 to 0xE are desired, then the HPS requires the VCCL_HPS voltage to be at least 0.95V to prevent boot failures or system instability issues.

The FPGA supply Vcc may be 0.9V if desired (to reduce device power consumption) as long as separate power rails are used for HPS VCCL_HPS and FPGA supplies.

For default boot mode where the CSEL fuses are set to 0x0, the VCCL_HPS voltage can be 0.9V.

4.2.3 Pin Connection Considerations for Board Designs

4.2.3.1 Device Power-Up

Follow the guidelines in the Device Driver Power-Up section of the Arria 10 Device Design Guidelines. In addition, consider the following guidelines for Arria 10 SoC devices.

Power-Up and Power-Down Sequencing

Arria 10 SoC devices have the following additional power rails to consider for power sequencing:

- VCCL_HPS
- VCCPLL_HPS, VCCIOREF_HPS
- VCCIO_HPS

Note: Refer to the *Power-Up and Power-Down Sequences* section of the *Power Management in Arria 10 Devices* chapter in the Arria 10 Core Fabric and General Purpose I/O Handbook.

4.2.3.2 Power Pin Connections and Power Supplies

Follow the guidelines in the *Power Pin Connections and Power Supplies* section of the Arria 10 Device Design Guidelines.

Follow the guidelines for Arria 10 SoC devices as documented in the *Arria 10 SX Pin Connection Guidelines* section of the Arria 10 GX, GT and SX Device Family Pin Connection Guidelines. The guidelines that follow supplement the above references.

GUIDELINE: Consider ramp times for maximum transient currents on supplies when designing the Power Distribution Network (PDN).

When using the *PDN Tool* to calculate the required target impedance of your application's PDN for the core fabric's VCC supply, model the ramp time of the maximum transient current on VCC using the **Core Clock Frequency** and **Current Ramp Up Period** parameters. This procedure relaxes the target impedance requirements relative to the default step function analysis, resulting in a more efficient PDN with fewer decoupling capacitors.

Initial transient current estimates can be obtained from the *EPE Spreadsheet*, and more accurate analysis is possible with the *PowerPlay Power Analysis Tool* in Quartus Prime when the design is closer to completion.

Refer to AN 750: Using the Altera PDN Tool to Optimize Your Power Delivery Network Design for further details.

GUIDELINE: Overdrive for maximum HPS MPU clock frequency.

Arria 10 SoC devices support HPS MPU overclocking.

Refer to the Arria 10 Device Datasheet for the maximum achievable MPU frequency and the associated requirements on device speed grade and HPS supply overdrive levels.

4.2.4 Power Analysis

GUIDELINE: Follow the guidelines in the *Power Analysis* section of the Arria 10 Device Design Guidelines

Access the AN 738-Arria 10 Device Design Guidelines here.

4.2.5 Power Optimization

Follow the guidelines in the Power Optimization section of the Arria 10 Device Design Guidelines.

While the HPS is a relatively low contributor to power consumption in the Arria 10 SoC device, there are some design choices and architectural features to consider as elaborated below.

4.2.5.1 Processor and memory clock speeds

The biggest contribution to power consumption from the HPS is the processor clock speed and the type, size and speed of the external SDRAM program memory.

Careful selection of these system parameters to satisfy the functional and performance requirements of the application while not being overdesigned helps to minimize system power consumption.

GUIDELINE: Use the Arria 10 FPGA-to-HPS bridge design example to tune HPS clock and memory interface parameters for your application's performance requiremements.

The design example is delivered as a Qsys subsystem to allow performance-related parameters, including MPU clock speed, Interconnect speed, and memory type, configuration and speed.

Arria 10 SoC development kits offer a selection of convenient hardware platforms for such analysis in the early stages of design. The Intel Arria 10 SoC Development Kit features high speed socketed external SDRAM memory daughtercards to experiment with different memory types and speeds.

With an optimal set of design parameters for the HPS and external SDRAM, your design will be optimized for lowest power consumption while still satisfying your application's performance requirements.

For more information, refer to SoC Series Kits, Arria 10 FPGA-to-HPS bridge design example and Arria 10 SoC Development Kit links on Intel's website.

4.2.5.2 MPU Standby Modes and Dynamic Clock Gating

CPU standby modes and dynamic clock gating logic can be utilized throughout the MPU subsystem. Each CPU can be placed in standby mode, Wait for Interrupt, or Wait for Event mode to further minimize power consumption.

GUIDELINE: Refer to the Cortex-A9 Processor Power Control section in the ARM Cortex-A9 Technical Reference Manual for more information on standby modes.

GUIDELINE: Utilize the power optimization examples available at the SoC Design Examples web page.

Please refer to the ARM Cortex-A9 Technical Reference Manual and SoC Design Examples for more information regarding the above guidelines.

4.2.5.3 Managing Peripheral Power

GUIDELINE: When configuring the HPS component in Qsys, enable only those peripherals your application uses.

GUIDELINE: Configure the peripherals for the lowest clock speed while maintaining functional and performance requirements.

GUIDELINE: To save additional power, design your application software to place inactive peripherals in reset and gate off their clock sources.

For a given peripheral, refer to the corresponding chapter in the Arria 10 Hard Processor System Technical Reference Manual under the section covering clocks and resets.

Refer to the Clock Manager and Reset Manager chapters in the Arria 10 Hard Processor Technical Reference Manual for more information on peripheral module clock and rest control.

4.2.5.4 Managing Power by Shutting Down Supplies

Lowering any of the supplies monitored by the internal POR circuitry on the SoC device (for example, the FPGA core supply (VCC) or the HPS supply (VCCL_HPS)) below their specified trip levels causes the FPGA fabric to go into a reset state.

GUIDELINE: Shutting down the FPGA core supply (VCC) effects the operation of the HPS.

When the FPGA is in POR reset, the Hard Memory Controller (HMC) I/O, Shared I/O and FPGA I/O are all in reset, resulting in the HPS losing connectivity to any external SDRAM and peripherals connected to these I/O. For the HPS to be fully operational, the FPGA supply voltages monitored by the POR circuitry must be above their POR values as described in the Arria 10 Core Fabric and General Purpose I/Os Handbook. FPGA I/O supply voltages must be at their recommended operating levels as specified in the Arria 10 Device Datasheet.

GUIDELINE: Shutting down the HPS voltage supply (VCCL_HPS) does not affect the FPGA core.

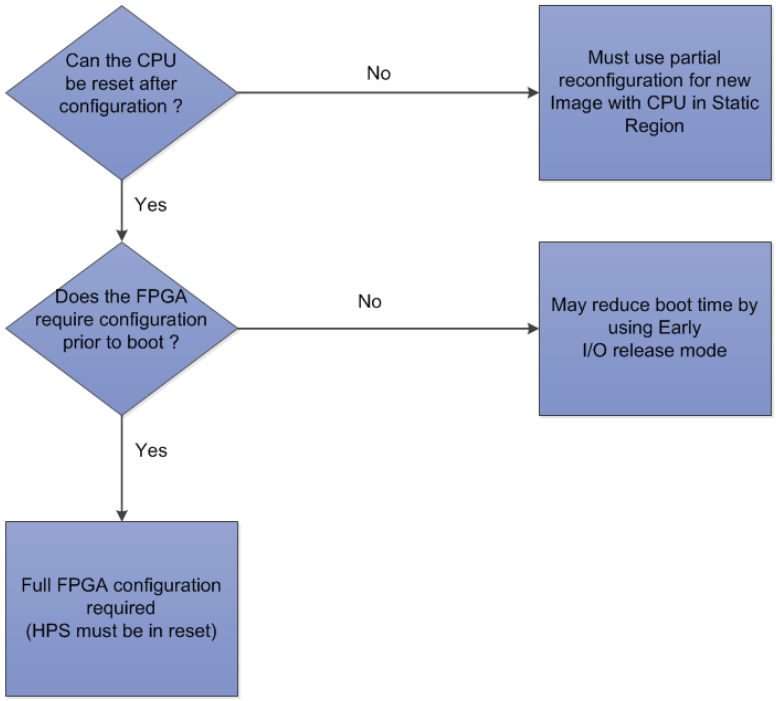
It is possible to shut down power to the HPS without affecting the FPGA fabric, FPGA I/O or any Shared I/O quadrants reserved for the FPGA portion, but you must observe the power-down sequencing requirements for the HPS supplies.

Refer to the *Power Management in Arria 10 Devices* chapter in the Arria 10 Core Fabric and General Purpose I/Os Handbook.

4.3 FPGA Reconfiguration

To determine the best method for reconfiguring the FPGA in your system, refer to the diagram below.

Figure 7. Flowchart to determine the best method for FPGA reconfiguration



Related Links

Traditional Configuration on page 76

4.3.1 Flash Update with HPS Reboot

GUIDELINE: If your system can tolerate having the HPS undergo a reboot in order to perform FPGA reconfiguration, then you should use the flash update with processor reboot method.

Instead of reconfiguring the FPGA, you can update the FPGA image stored in flash and reset the HPS. When the HPS reboots, the new FPGA image in flash is loaded and configured into the FPGA. This flow is similar to the one that many other embedded products such as cellular telephones, network routers, and television sets use to update firmware where the firmware is updated in flash, and the device is then rebooted for the new firmware to take effect.

This method of reconfiguring the FPGA hardware is ideal because it does not require hardware designers to partition the FPGA logic into a hierarchical design consisting of static and dynamic regions combined with partial reconfiguration. This method is supported in both Standard and Pro editions of Quartus Prime.

4.3.2 Partial Reconfiguration of the SoC FPGA

GUIDELINE: If the HPS cannot be reset to perform FPGA reconfiguration, then you must leverage the partial reconfiguration (PR) capabilities of the SoC FPGA device.

By using partial reconfiguration, you can ensure that the device I/O remain functional while a portion of the FPGA (known as a persona) is being replaced.

GUIDELINE: Ensure that you have Quartus Prime Pro Edition because it is the only edition that supports the hierarchical design flow and generation of configuration files supporting partial reconfiguration.

It is recommended that you ensure only the hardware that needs to be replaced is located in the dynamic region of the design. This partitioning ensures that the reconfiguration time is minimized and simplifies the hardware design because less hardware must be frozen before the partial reconfiguration occurs. Freezing a partial reconfiguration region ensures that while the partial reconfiguration occurs, any outputs from the PR region are driven to a known user-defined state so that surrounding hardware is not adversely affected.

GUIDELINE: Ensure that the region undergoing partial reconfiguration is isolated with freezing logic to ensure outputs are driven to a known safe state.

GUIDELINE: Ensure that the region undergoing partial reconfiguration allows any memory accesses or data movements to complete before freezing commences.

GUIDELINE: Ensure that the region undergoing partial reconfiguration has completed configuration before it is unfrozen and data movements to and from the replaced logic commence.

If the hardware being replaced conforms to an interconnect standard such as Avalon-MM, Avalon-ST, or AXI, then during the freezing process you must also ensure that there are no outstanding memory transactions or data movements before the hardware is frozen. Freeze logic should ensure that outstanding memory transactions or data movements complete before the freezing is allowed to commence.

Note: For additional information, please refer to the following documents:

- Arria 10 HPS Booting and Configuration Appendix
- Configuration, Design Security and Remote System Upgrades in Arria 10 Devices
- Design Planning for Partial Reconfiguration
- AN-798: Partial Reconfiguration with the Arria 10 HPS

4.4 Boundary Scan for HPS

GUIDELINE: Ensure that the HPS is held in cold reset before performing a boundary scan test of the FPGA and HPS I/O.

Most of the HPS dedicated and shared I/O support boundary scan testing. The exceptions are the HPS clock and reset inputs in the dedicated I/O bank.

4.5 Design Guidelines for HPS Interfaces

This section outlines the design guidelines for HPS Interfaces like EMAC, USB, QSPI, SD/MMC, NAND, UART and I^2C.

4.5.1 HPS EMAC PHY Interfaces

When configuring an HPS component for EMAC peripherals within Qsys, you must select from one of the following supported PHY interfaces for each EMAC instance:

- **Reduced Media Independent Interface (RMII) using Shared I/O**
- **Reduced Gigabit Media Independent Interface (RGMII) using Shared I/O**
- **Media Independent Interface (MII) interface to FPGA fabric**
- **Gigabit Media Independent Interface (GMII) interface to FPGA fabric**

Any combination of supported PHY interface types can be configured across multiple HPS EMAC instances.

GUIDELINE: For RMII and RGMII using Shared I/O, develop an early I/O floor-planning template design to ensure that there are enough Shared I/O to accommodate the chosen PHY interfaces in addition to other HPS peripherals planned for Shared I/O usage.

Note: Refer to the HPS Component section for guidelines on configuring the HPS component.

It is possible to adapt the MII/GMII PHY interfaces exposed to the FPGA fabric by the HPS component to other PHY interface standards such as RMII, RGMII, SGMII, SMII and TBI. through the use of soft adaptation logic in the FPGA and features in the general-purpose FPGA I/O and transceiver FPGA I/O.

GUIDELINE: When selecting a PHY device, conisder the desired Ethernet rate, available I/O and available transceivers, PHY devices that offer the skew control feature, and device driver availability.

Note: Refer to the device drivers available for your OS of choice or the Linux device driver provided with the Arria 10 SoC development kit. (Golden System Reference Design)

You can connect the Arria 10 HPS embedded Ethernet MAC (EMAC) PHY interfaces directly to industry standard Gigabit Ethernet PHYs using the RGMII interface and 10/100 Ethernet PHYs using the RMII interface at any supported I/O voltage using the Shared I/O pins in the HPS 3V I/O bank. These voltages typically include 1.8V, 2.5V and 3.0V. If you use Shared I/O pins for the PHY interface, then no FPGA routing resources are used and timing is fixed, simplifying timing on the interface. This document describes the design guidelines for RGMII and RMII, the most typical interfaces.

You can also connect PHYs to the HPS EMACs through the FPGA fabric using the GMII and MII bus interfaces for Gigabit and 10/100 Mbps access respectively.

GUIDELINE: A GMII-to-SGMII adapter is available to automatically adapt to transceiver-based SGMII optical modules.

4.5.1.1 PHY Interfaces Connected Through Shared I/O

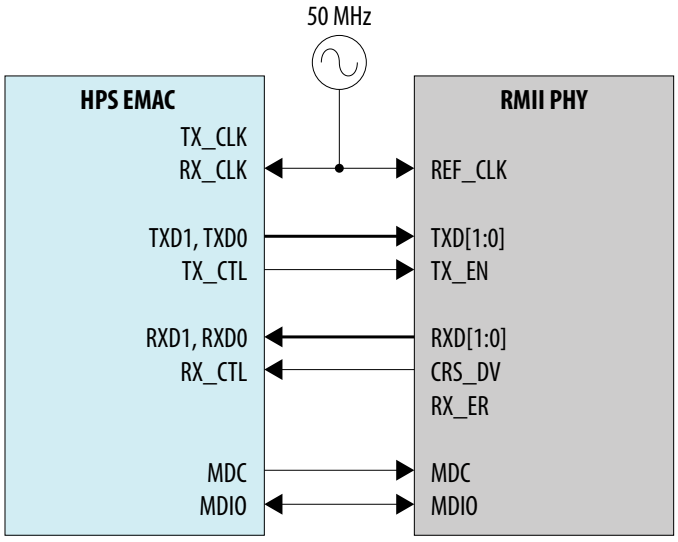
This section discusses design considerations for RMII and RGMII PHY interfaces through the HPS Shared I/O.

4.5.1.1.1 RMII

RMII uses a single centralized system-synchronous 50 MHz clock source (REF_CLK) for both transmit and receive paths across all ports. This simplifies system clocking and lowers pin counts in high port density systems, because your design can use a single board oscillator as opposed to per port TX_CLK/RX_CLK source synchronous clock pairs.

RMII uses two-bit wide transmit and receive datapaths. All data and control signals are synchronous to the REF_CLK rising edge. The RX_ER control signal is not used. In 10Mbps mode, all data and control signals are held valid for 10 REF_CLK clock cycles.

Figure 8. RMII MAC/PHY Interface



Interface Clocking Scheme

EMACs and RMII PHYs can provide the 50 MHz REF_CLK source. Using clock resources already present such as HPS_CLK1 input, internal PLLs further simplifies system clocking design and eliminates the need for an additional clock source.

This section discusses system design scenarios for both HPS EMAC-sourced and PHY-sourced REF_CLK.

GUIDELINE: Consult the PHY datasheet for specifics on the choice of REF_CLK source in your application.

Make sure your choice of PHY supports the REF_CLK clocking scheme in your application. Note any requirements and usage considerations specified in the PHY's datasheet.

You can use one of the following two methods for sourcing REF_CLK:

- HPS-Sourced REF_CLK
- PHY-Sourced REF_CLK

Figure 9. HPS Sourced REF_CLK

In this scheme, connect the EMAC's HPS RMII I/O TX_CLK output to both the HPS RMII I/O RX_CLK and PHY REF_CLK inputs.

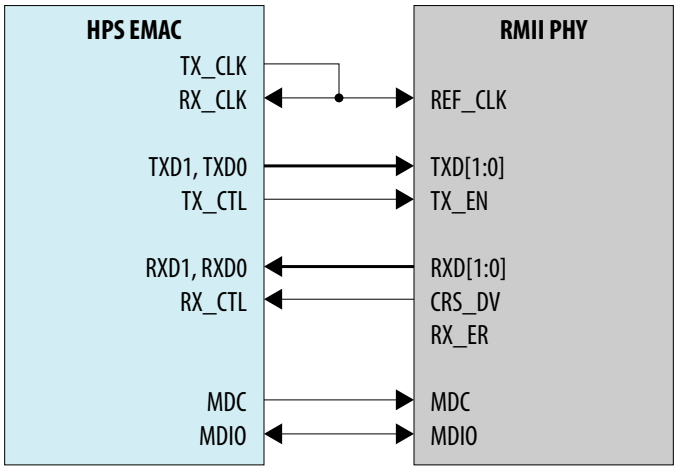
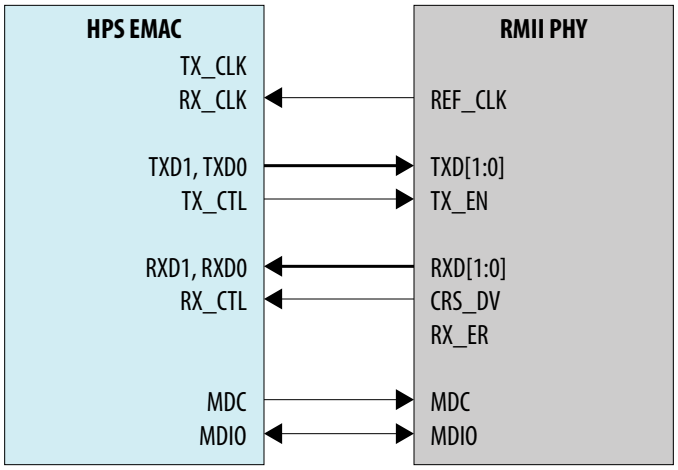


Figure 10. PHY Sourced REF_CLK

In this scheme, connect the PHY's REF_CLK output to the EMAC's HPS RMII I/O RX_CLK input. Leave the EMAC's HPS RMII I/O TX_CLK output unconnected. TX_CLK is always part of the HPS EMAC I/O signal set for RMII and cannot be made available as an additional Shared I/O even when not used. PHYs capable of sourcing REF_CLK are typically configured to do so through pin bootstrapping and require an external crystal or clock input to generate REF_CLK.



I/O Pin Timing

Note: **Account for routing delay differences from the REF_CLK source to REF_CLK input pins between the HPS EMAC and PHY**

If RX_CLK is routed daisy-chain from source to MAC to PHY or source to PHY, you must account for the flight time difference as both REF_CLK loads will see the clock at different times.

GUIDELINE: Take into account routing delays and skews on the data and control signals to ensure meeting setup and hold as specified in the HPS SoC Device datasheet and PHY datasheet.

Signal length matching is not necessary unless you have signal lengths in excess of 24”, in which case you must perform some basic timing analysis with clock delays versus data delays.

The period is 20 ns with the 50 MHz REF_CLK and remains at this frequency regardless of whether the PHY is set to 10Mbps or 100Mbps mode.

All clocking in the HPS EMAC is based on the RX_CLK, so the Tco and PCB flight time of REF_CLK from either the EMAC or PHY can be ignored. Typical board traces up to 12 inches yield only 2 ns of flight time and Tsu of RXD to RX_CLK is 4 ns minimum, well under the 20 ns period.

There is a 2 ns hold requirement of RXD versus RX_CLK which is easily satisfied as well because the Tco of TXD with respect to RX_CLK for either the MAC or the PHY is typically over 2 ns. For Arria 10 SoC device, the Tco of TXD with respect to RX_CLK is 7 ns to 10 ns.

GUIDELINE: Ensure the REF_CLK source meets the duty cycle requirement.

There is no jitter specification for the REF_CLK, but there is a duty cycle requirement of 35% to 65%. This requirement is met by Arria 10 PLLs and clock outputs for GPIO or for the TX_CLK signal coming from the HPS IP specifically.

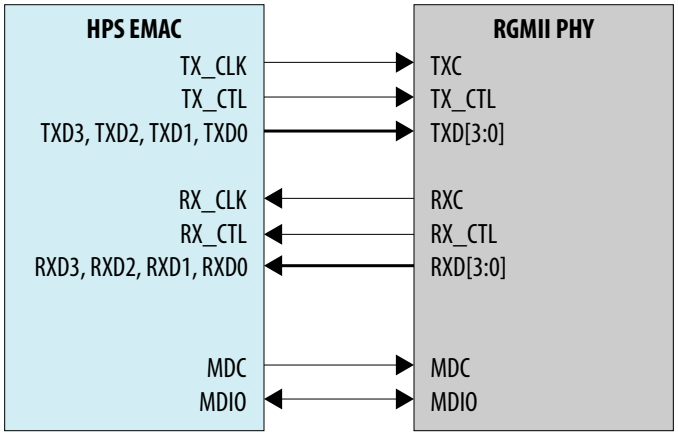
4.5.1.1.2 RGMII

RGMII is the most common interface because it supports 10 Mbps, 100 Mbps, and 1000 Mbps connection speeds at the PHY layer.

RGMII uses four-bit wide transmit and receive datapaths, each with its own source-synchronous clock. All transmit data and control signals are source synchronous to TX_CLK, and all receive data and control signals are source synchronous to RX_CLK.

For all speed modes, TX_CLK is sourced by the MAC, and RX_CLK is sourced by the PHY. In 1000 Mbps mode, TX_CLK and RX_CLK are 125 MHz, and Dual Data Rate (DDR) signaling is used. In 10 Mbps and 100 Mbps modes, TX_CLK and RX_CLK are 2.5 MHz and 25 MHz, respectively, and rising edge Single Data Rate (SDR) signaling is used.

Figure 11. RGMII MAC/PHY Interface



I/O Pin Timing

This section addresses RGMII interface timing from the perspective of meeting requirements in the 1000 Mbps mode. The interface timing margins are most demanding in 1000 Mbps mode, thus it is the only scenario we consider here.

At 125 MHz, the period is 8 ns, but because both edges are used, the effective period is only 4 ns. The TX and RX busses are completely separate and source synchronous, simplifying timing. The RGMII specification calls for CLK to be delayed from DATA at the receiver in either direction by a minimum 1.0 ns and a maximum 2.6 ns.

In other words, the TX_CLK must be delayed from the MAC output to the PHY input and the RX_CLK from the PHY output to the MAC input. The signals are transmitted source synchronously within the +/- 500 ps RGMII skew specification in each direction as measured at the output pins. The minimum delay needed in each direction is 1 ns but it is recommended to target a delay of 1.5 ns to 2.0 ns to ensure significant timing margin.

Transmit path setup/hold

Only setup and hold for TX_CLK to TX_CTL and TXD[3:0] matter for transmit. The Arria 10 I/O can provide up to 800 ps additional delay on outputs. This delay is enabled using the output delay logic option within the assignment editor in Quartus Prime.

GUIDELINE: For TX_CLK from the Arria 10, you must introduce at least 200 ps of delay beyond the 800 ps I/O delay to meet the 1.0 ns PHY minimum input setup time in the RGMII spec.

It is strongly recommended to increase this to HPS-provided 800 ps I/O delay plus 700 ps-1200 ps other delay for a total recommended delay of 1.5 ns to 2.0 ns. Many PHYs offer programmable skew, and some support RGMII 2.0 which defaults to skew enabled on both transmit and receive datapaths.

GUIDELINE: Between PHY delay and FPGA I/O delay features, you must ensure either 2 ns of delay to CLK versus CTL and D[3:0] or 1.2 ns typical minimum setup skew typical of most PHYs.

Consult the datasheet for your PHY vendor for more information.

GUIDELINE: Ensure your design includes the necessary Intel settings to configure the HPS EMAC outputs for the required delays.

On the Arria 10 SoC Development Kit and the associated Arria 10 Golden Hardware Reference Design (the GHRD is the hardware component of the GSRD), a combination of PHY skew and FPGA skew is implemented with the Micrel PHY. Refer to the Quartus Prime settings and PHY driver code in the Golden System Reference Design (GSRD).

Receive path setup/hold

Only setup and hold for RX_CLK to RX_CTL and RXD[3:0] are necessary to consider for receive timings. The Arria 10 I/O can provide up to 3200 ps additional delay on inputs. For Arria 10 inputs, the 3.2 ns I/O delay can achieve this timing for RX_CLK without any other considerations on the PHY side or board trace delay side.

GUIDELINE: Hardware developers must specify the required FPGA skew so that software developers can add the skew to the device driver code.

An example of this code is available in the Linux device driver for the Arria 10 GSRD.

4.5.1.2 PHY Interfaces Connected Through FPGA I/O

Using FPGA I/O for an HPS EMAC PHY interface can be helpful when there are not enough Shared I/O left to accommodate the PHY interface or when you want to adapt to a PHY interface not natively supported by the HPS EMAC.

GUIDELINE: Specify the PHY interface transmit clock frequency when configuring the HPS component in Qsys.

For either GMII or MII, including adapting to other PHY interfaces, specify the maximum transmit path clock frequency for the HPS EMAC PHY interface: 125 MHz for GMII, 25 MHz for MII. This configuration results in the proper clock timing constraints being applied to the PHY interface transmit clock upon Qsys system generation.

4.5.1.2.1 GMII/MII

GMII and MII are only available in Arria 10 by driving the EMAC signals into the FPGA core routing logic instead of the Shared I/O, then ultimately to FPGA I/O pins or to internal registers in the FPGA core.

GUIDELINE: Apply timing constraints and verify timing with TimeQuest.

Because routing delays can vary widely in the FPGA core and I/O structures, it is important to read the timing reports, and especially for GMII, create timing constraints. GMII has a 125 MHz clock and is single data rate unlike RGMII. GMII does not have the same considerations for CLK-to-DATA skew though; its signals are automatically centered by design by being launched with the negative edge and captured with the rising edge.

GUIDELINE: Register interface I/O at the FPGA I/O boundary.

With core and I/O delays easily exceeding 8 ns, it is recommended to register these buses in each direction in I/O Element (IOE) registers, so they remain aligned as they travel across the core FPGA logic fabric. On the transmit data and control, maintain the `clock-to-data/control` relationship by latching these signals on the falling edge of the `emac[0,1,2]_gtx_clk` output from the HPS EMAC. Latch the receive data and control at the FPGA I/O inputs on the rising edge of the RX_CLK sourced by the PHY.

GUIDELINE: Consider transmit timing in MII mode.

MII is 25 MHz when the PHY is in 100 Mbps mode and 2.5 MHz when the PHY is in 10 Mbps mode, so the shortest clock period is 40 ns. The PHY sources the clock for both transmit and receive directions. Because the transmit timing is relative to the TX_CLK clock provided by the PHY, the turnaround time may be of concern, but this is usually not an issue due to the long 40 ns clock period.

Since the reference clock is transmitted through the FPGA, then out for the data – the round-trip delay must be less than 25 ns as there is a 15 ns input setup time. Note that the transmit data and control are launched into the FPGA fabric by the HPS EMAC transmit path logic on the negative edge of the PHY-sourced TX_CLK, which removes 20 ns of the 40 ns clock-to-setup timing budget.

With the round-trip clock path delay on the data arrival timing incurring PHY-to-SoC board propagation delay plus the internal path delay from the SoC pin to and through the HPS EMAC transmit clock mux taking away from the remaining 20 ns setup timing budget, it may be necessary to retime the transmit data and control to the rising edge of the `phy_txclk_o` clock output registers in the FPGA fabric for MII mode transmit data and control.

4.5.1.2.2 Adapting to RGMII

The GMII-to-RGMII Adapter core allows you to adapt the GMII HPS EMAC PHY signals to an RGMII PHY interface at the FPGA I/O pins using logic in the FPGA. While it is possible to design custom logic for this adaptation, this section describes using Qsys adapter IP.

GUIDELINE: Use the GMII-to-RGMII Adapter IP available in Qsys.

Configure the HPS component in Qsys for an EMAC "To FPGA" I/O instance and choose GMII as the PHY interface type along with a management interface. Do not export the resulting HPS component GMII signals in Qsys. Instead, add the Intel GMII to RGMII Adapter IP to the Qsys subsystem and connect to the HPS component's GMII signals. The GMII-to-RGMII Adapter IP makes use of the Intel HPS EMAC Interface Splitter IP in Qsys to split out the "`emac`" conduit from the HPS component for use by the GMII to RGMII Adapter. See the Embedded Peripherals User Guide for information on how to use the Intel GMII-to-RGMII Adapter IP.

GUIDELINE: Provide a glitch-free clock source for the 10/100Mbps modes.

In an RGMII PHY interface, the TX_CLK is always sourced by the MAC, but the HPS component's GMII interface expects TX_CLK to be provided by the PHY device in 10/100 Mbps modes. The GMII to RGMII adaptation logic must provide the 2.5/25 MHz TX_CLK on the GMII's `emac[0,1]_tx_clk_in` input port. The switch between

2.5 MHz and 25 MHz must be accomplished in a glitch-free manner as required by the HPS EMAC. An FPGA PLL can be used to provide the 2.5 MHz and 25 MHz TX_CLK along with an ALTCLKCTRL IP block to select between counter outputs glitch-free.

4.5.1.2.3 Adapting to RMII

It is possible to adapt the MII HPS EMAC PHY signals to an RMII PHY interface at the FPGA I/O pins using logic in the FPGA.

GUIDELINE: Provide a 50 MHz REF_CLK source.

An RMII PHY uses a single 50 MHz reference clock (REF_CLK) for both transmit and receive data and control. Provide the 50 MHz REF_CLK either with a board-level clock source, a generated clock from the FPGA fabric, or from a PHY capable of generating the REF_CLK.

GUIDELINE: Adapt the transmit and receive data and control paths.

The HPS EMAC PHY interface exposed in the FPGA fabric is MII, which requires separate transmit and receive clock inputs of 2.5 MHz and 25 MHz for 10 Mbps and 100 Mbps modes of operation, respectively. Both transmit and receive datapaths are 4-bits wide. The RMII PHY uses the 50 MHz REF_CLK for both its transmit and receive datapaths and at both 10 Mbps and 100 Mbps modes of operation. The RMII transmit and receive datapaths are 2-bits wide. At 10 Mbps, transmit and receive data and control are held stable for 10 clock cycles of the 50 MHz REF_CLK. You must provide adaptation logic in the FPGA fabric to adapt between the HPS EMAC MII and external RMII PHY interfaces: four bits at 25MHz and 2.5 MHz, to and from two bits at 50 MHz, and 10x oversampled in 10 Mbps mode.

GUIDELINE: Provide a glitch-free clock source on the HPS EMAC MII `tx_clk_in` clock input.

The HPS component's MII interface requires a 2.5/25 MHz transmit clock on its `emac[0,1,2]_tx_clk_in` input port. The switch between 2.5 MHz and 25 MHz must be done glitch free as required by the HPS EMAC. An FPGA PLL can be used to provide the 2.5 MHz and 25 MHz transmit clock along with an ALTCLKCTRL IP block to select between counter outputs glitch-free.

4.5.1.2.4 Adapting to SGMII

You can use the GMII-to-SGMII Adapter core to adapt the GMII HPS EMAC PHY signals to an SGMII PHY interface at the FPGA transceiver I/O pins using logic in the FPGA and the multi-gigabit transceiver I/O or LVDS SERDES in soft CDR mode. While it is possible to design custom logic for this adaptation, this section describes using Qsys adapter IP.

GUIDELINE: Use the GMII to SGMII Adapter IP available in Qsys.

Configure the HPS component in Qsys for an EMAC "To FPGA" I/O instance and choose GMII as the PHY interface type along with a management interface. Do not export the resulting HPS component GMII signals in Qsys. Instead, add the Intel GMII to SGMII Adapter IP to the Qsys subsystem and connect to the HPS component's GMII signals. The GMII to SGMII Adapter IP makes use of the Intel HPS EMAC Interface Splitter IP in Qsys to split out the "emac" conduit from the HPS component for use by the GMII to SGMII Adapter. The adapter IP instantiates the Intel Triple Speed Ethernet (TSE)

MAC IP, configured in 1000BASE-X/SGMII PCS PHY-only mode (i.e., no soft MAC component). See the Embedded Peripherals User Guide for information on how to use the Intel GMII to SGMII Adapter IP.

4.5.1.3 MDIO

The MDIO PHY management bus has two signals per MAC: MDC and MDIO. MDC is the clock output, which is not free running. At 2.5 MHz, it has a 400 ns minimum period. MDIO is a bidirectional data signal with a High-Z bus turnaround period.

When the MAC writes to the PHY, the data is launched on the falling edge, meaning there is 200 ns -10 ns = 190 ns for flight time, signal settling, and setup at the receiver. Because data is not switched until the following negative edge, there is also 200 ns of hold time. These requirements are very easy to meet with almost any board topology. When the MAC reads from the PHY, the PHY is responsible to output the read data from 0 to 300 ns back to the MAC, leaving 100 ns less 10 ns setup time, or 90 ns for flight time, signal settling, and setup at the receiver. This requirement is also very easy to meet.

GUIDELINE: Board pull-ups on MDC/MDIO.

Both signals require an external pull-up resistor. Consult your PHY's datasheet for the correct pull-up resistor value. 1K Ohm is a typical resistor value.

GUIDELINE: Ensure interface timing that MDIO requires.

MDIO requires a 10 ns setup and hold time for data with respect to MDC.

4.5.1.4 Common PHY Interface Design Considerations

4.5.1.4.1 Signal Integrity

GUIDELINE: Make use of the SoC device's On-Chip Termination (OCT).

Arria 10 devices can tune their outputs to many settings, with 50 ohm output impedance often being the best value. Quartus Prime automatically uses series OCT without calibration on RGMII outputs. Check the Quartus Prime fitter report to verify the OCT settings on the interface's outputs.

GUIDELINE: Use appropriate board-level termination on PHY outputs.

Not many PHYs offer I/O tuning for their outputs, so it is recommended that you verify the signal path to the Arria 10 device with a simulator. Place a series resistor on each signal near the PHY output pins to reduce the reflections if necessary.

GUIDELINE: Ensure reflections at PHY TX_CLK and EMAC RX_CLK inputs are minimized to prevent double-clocking.

Be cognizant if the connection is routed as a "T" as signal integrity must be maintained such that no double-edges are seen at REF_CLK loads. Ensure reflections at REF_CLK loads are minimized to prevent double-clocking.

GUIDELINE: Use a Signal Integrity (SI) simulation tool.

It is fairly simple to run SI simulations on these unidirectional signals. These signals are almost always point-to-point, so simply determining an appropriate series resistor to place on each signal is usually sufficient. Many times, this resistor is not necessary, but the device drive strength and trace lengths as well as topology should be studied when making this determination.

4.5.2 USB Interface Design Guidelines

The Arria 10 HPS can connect its embedded USB MACs directly to industry-standard USB 2.0 ULPI PHYs using the shared I/O pins in the HPS 3V I/O bank that support 1.8V, 2.5V and 3.0V I/O standards. No FPGA routing resources are used and timing is fixed, which simplifies design.

This guide describes the design guidelines covering all supported speeds of PHY operation: High-Speed (HS) 480 Mbps, Full-Speed (FS) 12 Mbps, and Low-Speed (LS) 1.5 Mbps.

GUIDELINE: It is recommended that you design the board to support both USB PHY modes where the device supplies the clock versus where an external clock is the source.

The interface between the ULPI MAC and PHY on the Arria 10 SoC consists of DATA[7:0], DIR and NXT from the MAC to the PHY and STP from the MAC to the PHY. Lastly a static clock of 60MHz is driven from the PHY or from an external oscillator and is required for operation, including some register accesses from the HPS to the USB MAC. Ensure the PHY manufacturer recommendations for RESET and power-up are followed.

If your USB PHY supports both input and output clock modes, it is recommended that you design your board to support both modes to mitigate potential timing issues. Typically, these modes are selected through passive bootstrap pins that are either pulled high or low.

GUIDELINE: Ensure that the USB signal trace lengths are minimized.

At 60 MHz, the period is 16.67 ns and in that time, for example, the clock must travel from the external PHY to the MAC and then the data and control signals must travel from the MAC to the PHY. Because there is a round-trip delay, the maximum length of the CLK and ULPI signals are important. Based on preliminary timing data the maximum length is recommended to be less than 7 inches. This is based on a PHY with a 5 ns Tco spec. If the specification is slower the total length must be shortened accordingly.

If there is little setup timing margin on the USB PHY end of the bus, sometimes you can switch the PHY to input clock mode and supply a 60 MHz clock source from the board.

GUIDELINE: Ensure that signal integrity is considered.

Signal integrity is important mostly on the CLK signal driven from the PHY to the MAC in the HPS. Because these signals are point-to-point with a maximum length, they can usually run unterminated but it is recommended to simulate the traces to make sure

the reflections are minimized. Using the 50-ohm output setting from the FPGA is typically recommended unless the simulations show otherwise. A similar setting should be used from the PHY vendor if possible.

GUIDELINE: Design properly for OTG operation, if used.

When On-the-Go (OTG) functionality is used, the SoC can become a host or endpoint. When in host mode consider the power delivery, such as when you are supporting a USB Flash drive, or potentially a USB Hard Drive. These power requirements and reverse currents must be accounted for typically through the use of external diodes and current limiters such as those used on the Intel development kits for Arria 10 SoC.

Refer to the Arria 10 SoC Development Board Schematics for more information.

4.5.3 QSPI Flash Interface Design Guidelines

Up to four QSPI chip selects can be used with Arria 10 SoCs. The device can boot only from QSPI connected to the chip select zero.

GUIDELINE: Ensure that the QSPI_SS signals are used in numerical order.

Quartus Prime assumes that the QSPI_SS signals are used in order. It is not possible to use SS0 and SS2, for example, without using SS1.

Note that the QSPI_SS1 pin is also used as BSEL0. Therefore, when selecting BSEL=0x6 (1.8V QSPI), the QSPI_SS1 function is unusable.

Note: Refer to Supported Flash Devices for Arria 10 SoC for a list of supported QSPI devices. The RocketBoards.org website also provides useful information about Booting from QSPI Flash and Programming QSPI Flash.

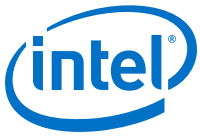
4.5.4 SD/MMC and eMMC Card Interface Design Guidelines

GUIDELINE: If the SD/MMC power enable is used in the design, implement one of the required workarounds to ensure that your power enable functions properly.

The SD/MMC power enable is intended to function such that a logic high enables power to the SD/MMC card and a logic low disables the power. However, the SD/MMC power enable (SDMMC_PWR_ENA_HPS) signal and the BSEL[1]signal share the same dedicated I/O pin. When booting from the SD/MMC card, BSEL[1] is pulled low during a power-on reset and prevents the boot ROM from copying the second-stage boot loader from flash into on-chip RAM.

There are three workaround options for this issue:

- Force the power enable high on the board
- Use a GPIO to control the power enable
- Invert the power enable line on the board so that when software disables the power (SDMMC_PWR_ENA_HPS is high), the board inverts the signal to turn off the card



GUIDELINE: Ensure that voltage translation transceivers are properly implemented if using 1.8V SD card operation.

SD cards initially operate at 3V, and some cards can switch to 1.8V after initialization. In addition, some MMC cards can operate at both 1.8V as well as 3.3V. Since the HPS I/O use a fixed voltage level and cannot be changed dynamically, transceivers are required to support level-shifting and isolation for cards that can operate at 1.8 V.

Follow the guidelines in the Voltage Switching section of the SD/MMC Controller chapter in the Arria 10 Hard Processor System Technical Reference Manual. Some MMC cards can operate with only 1.8V I/O operation and initial operation at 3.0V is not required. In this situation, a level shifter is not needed.

Table 9. SD Card Implementations Requiring a Level Shifter

HPS I/O Bank Voltage	SD Card Voltage	Level Shifter Needed
3.0 V	3.0 V	No
3.0 V	1.8 V	Yes
1.8 V	3.0 V	Yes
1.8 V	1.8 V	Yes

GUIDELINE: Ensure that timing is considered for initial ID mode and data transfer mode as well as normal operation.

SD cards initially operate at a maximum of 400 KHz frequency during the ID process. After the device has been identified, the bootrom switches to data transfer mode where the clock can operate up to 12.5 MHz. Typically, the second stage bootloader will increase the interface speed further up to a maximum operating frequency of 50 MHz.

Refer to the "SD/MMC Controller Clock Options Based on CSEL and HPS_CLK fuse settings" table in the Booting and Configuration appendix of the Arria 10 Hard Processor System Technical Reference Manual.

GUIDELINE: Ensure that the SD/MMC card is reset when the HPS is reset

To allow the system to boot from SD/MMC, whenever the HPS is reset, ensure that the SD/MMC card is also reset, so that the memory card is in the state expected by the boot code.

4.5.5 NAND Flash Interface Design Guidelines

GUIDELINE: Ensure that the selected NAND flash device is an 8- or 16-bit ONFI 1.0 compliant device.

The NAND flash controller in the HPS requires:

- The external flash device to be 8- or 16-bit ONFI 1.0 compliant
- x8 interface for boot devices, x16 supported for mass storage (non-boot) usage
- Single-level cell (SLC) or multi-level cell (MLC)
- Only one ce# and rb# pin pair is available for the boot source. Up to three additional pairs are available for mass storage

- Page size: 512 bytes, 2 KB, 4 KB or 8 KB
- Pages per block: 32, 64, 128, 256, 384 or 512
- Error correction code (ECC) sector size can be programmed to 512 bytes (for 4, 8 or 16 bit correction) or 1024 bytes (24-bit correction)
  - When the NAND device is used for booting, the boot ROM uses a 512B sector size with ECC support for up to eight correctable bits per sector.

Note: When selecting to boot from NAND, all the dedicated HPS I/O lines are used, so the UART signals (if needed) must be routed through FPGA fabric. Therefore, the UART logging is not available until the Shared I/O is configured.

Refer to the Supported Flash Devices for Arria 10 SoC web page for a list of supported NAND devices.

4.5.6 UART Interface Design Guidelines

GUIDELINE: Properly connect flow control signals when routing the UART signals through the FPGA fabric.

When routing UART signals through the FPGA, the flow control signals are available. If flow control is not being used, connect the signals in the FPGA as shown in the following table:

Table 10. UART Interface Design

Signal	Direction	Connection
CTS	input	low
DSR	input	high
DCD	input	high
RI	input	high
DTR	output	No-Connection
RTS	output	No-Connection
OUT1_N	output	No-Connection
OUT2_N	output	No-Connection

4.5.7 I²C Interface Design Guidelines

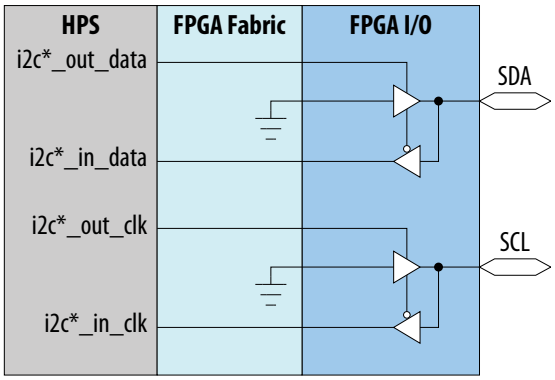
GUIDELINE: Instantiate the open-drain buffer when routing I²C signals through the FPGA fabric.

When routing I²C signals through the FPGA, note that the I²C pins from the HPS to the FPGA fabric (`i2c*_out_data`, `i2c*_out_clk`) are not open-drain and are logic level inverted. Thus, to drive a logic level zero onto the I²C bus, drive the corresponding pin high. This implementation is useful as they can be used to tie to an output enable of a tri-state buffer directly. You must use the `altiobuf` to implement the open-drain buffer.

GUIDELINE: Ensure that the pull-ups are added to the external SDA and SCL signals in the board design.

Because the I^2C signals are open drain, pull-ups are required to make sure that the bus is pulled high when no device on the bus is pulling it low.

Figure 12. I^2C Wiring to FPGA pins



5 Embedded Software Design Guidelines for Arria 10 SoC FPGAs

5.1 Embedded Software for HPS : Design Guidelines

5.1.1 Purpose

This chapter covers the design considerations for assembling your software development platform for the Arria 10 Hard Processor System.

You must follow the provided recommendations to select the components of your software platform that suits the performance, support and time-to-market requirements of your end application.

5.1.2 Assembling the components of your Software Development Platform

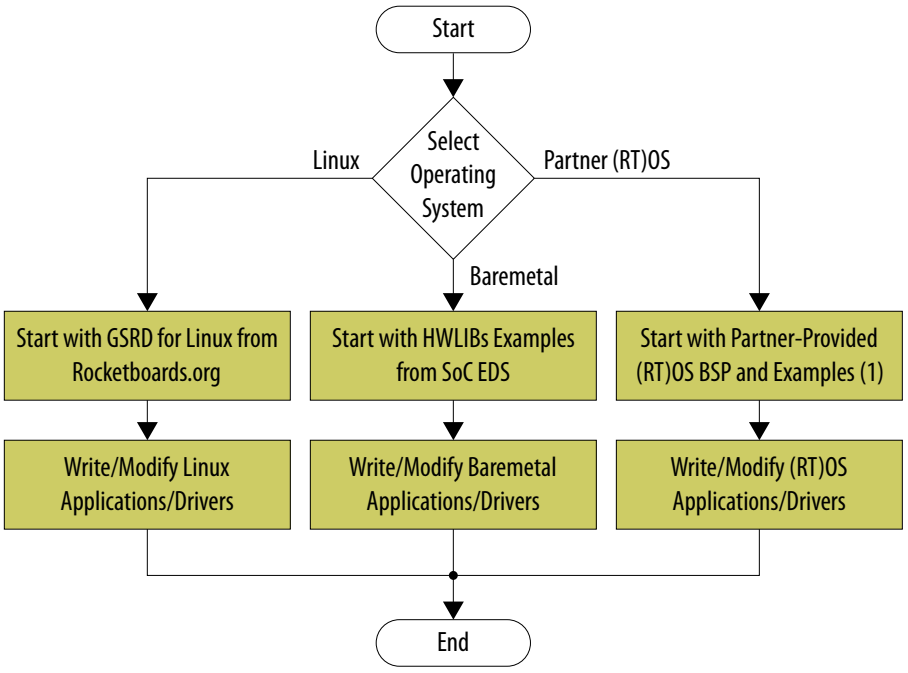
To successfully build your software development platform, it is recommended that you start with a baseline project; a known good configuration of an HPS system, and then modify the baseline project to suit your end application.

The following diagram presents the recommended procedure to follow in order to determine the software development platform components.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2008 Registered

Figure 13. Assembling Software Development Platform



(1) Some, but not all, partner-provided BSPs are based on the GHRD.

Summarizing, the flow consists of following steps:

1. Select the operating system: bare-metal, Linux or partner operating system, or real-time operating system
2. Write and/or update end application and/or drivers

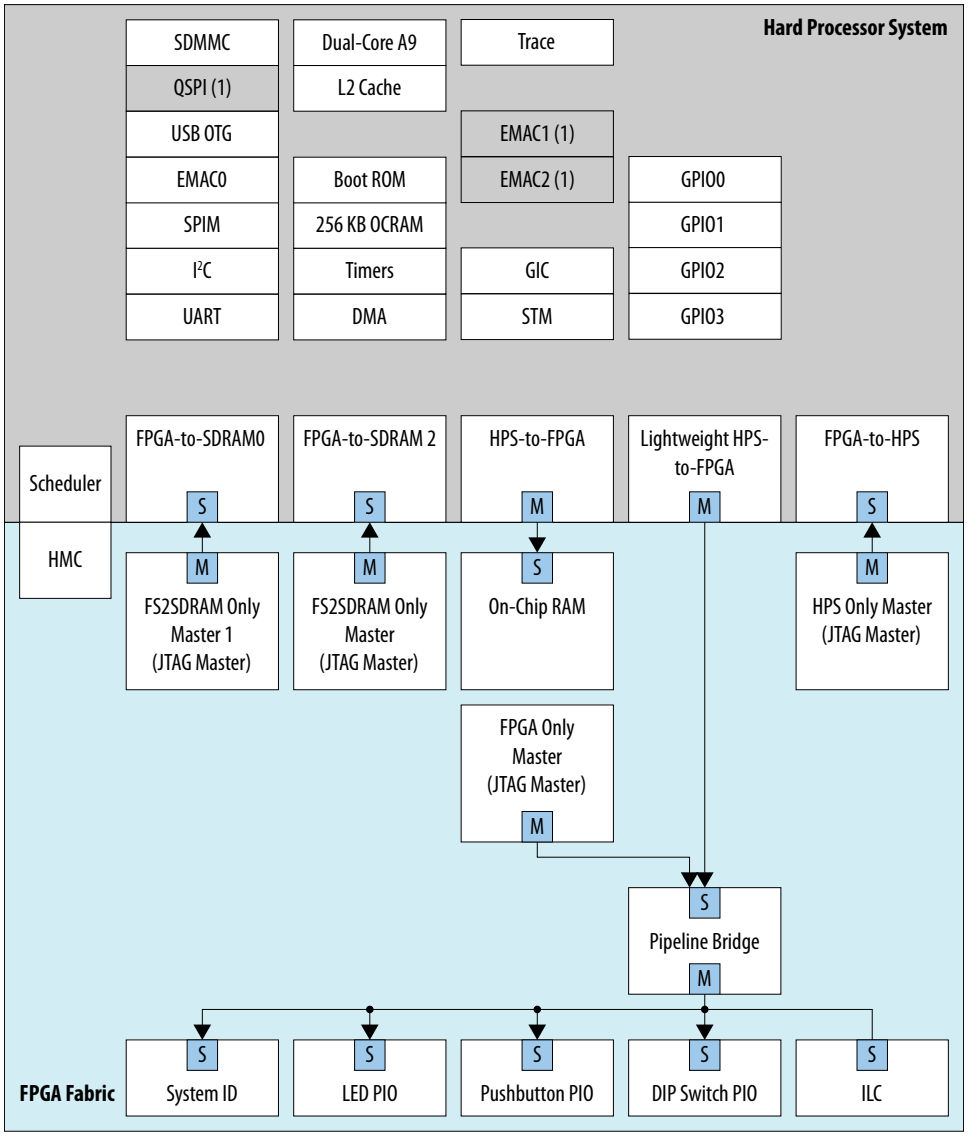
5.1.2.1 Golden Hardware Reference Design (GHRD)

The GHRD is a Quartus Prime project that contains a full HPS design for the Arria 10 SoC Development Kit. The GHRD has connections to a boot source, SDRAM memory and other peripherals on the development board.

You must always use a hardware design with the Arria 10 SoC if you wish to take advantage of the HPS's features. The purpose of the hardware design is to configure the SoC, including the FPGA portion, the HPS pin multiplexers and I/Os, and the DDRAM. All SoC EDS software projects depend on a hardware design.

The GHRD is regression tested with every major release of the Quartus Prime Design Suite (QPDS) and includes the latest bug fixes for known hardware issues. As such, the GHRD serves as a known good configuration of a SoC FPGA hardware system.

Figure 14. Arria 10 Golden Hardware Reference Design (GHRD) Overview



(1) These components are not configured in the default GHRD. However, it is easy to add them using Tcl scripts that accompany the reference design in the SoC EDS.

GUIDELINE: It is recommended that you use the latest GHRD as a baseline for new SoC FPGA hardware projects. You may then modify the design to suit your end application needs.

The GHRD can be obtained from:

- GSRD for Linux page for the latest version which is the best known configuration.
- SoC EDS Installation folder - `<SoC EDS Installation directory>\examples\hardware\a10_soc_devkit_ghrd` - for the version supported by the corresponding SoC EDS version, used as a basis for the provided HWLIBs design examples in SoC EDS. This may not be the latest configuration.

5.1.3 Selecting an Operating System for your application

5.1.3.1 Using Linux or RTOS

There are several operating systems that support the Arria 10 SoC, including Linux OS. For more information, go to Intel's SoC Partner OS ecosystem webpage.

Intel supports a variety of Linux kernel choices for the SoC devices. Supported kernels include the latest stable kernel, latest Long Term Support Initiative (LTSI) kernel, and the LTSI kernel with real-time preemption (PREEMPT_RT).

From a user-space perspective, Intel enables the Yocto Project compatible, Angstrom distribution.

Partner OS providers offer board support packages and commercial support for the SoC FPGA devices. The Linux community also offers board support packages and community support for the SoC FPGA device.

There are many factors that go into the selection of an operation system for SoC FPGAs including the features of the operating system, licensing terms, collaborative software projects and frameworks based on the OS, available device drivers and reference software, in-house legacy code and familiarity with the OS, real time requirements of your system, functional safety and other certifications required for your application.

To select an appropriate OS for your application, it is recommended that you familiarize yourself with the features and support services offered by the commercial and open source operating systems available for the SoC FPGA. Intel’s OS partners, industry websites are a good source of information you can use to help make your selection.

There are a number of misconceptions when it comes to real time performance of operating systems versus bare metal applications. For an ARM Cortex A-class of processor there are a number of features that real time operating systems provide that make efficient use of the processor’s resources in addition to the facilities provided to manage the run-time application.

You may find that these efficiencies result in sufficient real time performance for your application, enabling you to inherit a large body of available device drivers, middleware packages, software applications and support services. It is important to take this into account when selecting an operating system.

5.1.3.2 Developing a Bare-Metal Application

The HPS can be used in a bare-metal configuration (without an OS) and Intel offers Hardware Libraries (HWLibs) that consist of both high-level APIs, and low level macros for most of the HPS peripherals.

Typically bare-metal software is used for board bring-up, but bare metal can also be used as the actual application platform. However, to develop a bare-metal application for the HPS, you must be familiar with developing run-time capabilities to ensure that your bare metal application makes efficient use of resources available in your MPU subsystem.

For example:

- A typical bare-metal application uses only a single core, you must develop run time capabilities to manage both cores and the cache subsystem if you want to fully utilize the MPU subsystem.
- As your application increases in complexity you may need to build capabilities to manage and schedule processes, handle inter-process communication and synchronize between events within your application.

To this end, even a small lightweight RTOS offers simple scheduling, inter-process communication and interrupt handling capabilities that will make a more efficient use of the resources in your MPU subsystem.

5.1.3.3 Using the Bootloader as a Bare-Metal Framework

If your application is relatively simple, and does not require complex features such as multi-core or multi-tasking, one option is to include it in the bootloader.

Including your application in the bootloader has the following advantages:

- Potentially faster boot time
- Access to features already implemented in the bootloader, such as mass storage and networking

The following bootloaders are available, with source code:

- U-Boot: open-source GPL License
- MPL: open-source BSD License
- UEFI: open-source BSD license

5.1.3.4 Using Symmetrical vs. Asymmetrical Multiprocessing (SMP vs. AMP) Modes

The Dual Core ARM Cortex-A9 MPCore in the Arria 10 HPS can support both Symmetrical Multi Processing (SMP) and Asymmetrical Multi-processing (AMP) configuration modes.

In SMP mode, a single OS instance controls both cores. The SMP configuration is supported by a wide variety of OS manufacturers and is the most common and straightforward configuration mode for multiprocessing.

Linux and commercially developed operating systems offer features that take full advantage of the CPU cores resources and use them in an efficient manner resulting in optimum performance and ease of use. For instance, SMP enabled operating systems offer the option of setting processor affinity. This means that each task/thread can be assigned to run on a specific core. This feature allows the software developer to better control the workload distribution for each ARM Cortex-A9 core and making the system more responsive as an alternative to AMP.

GUIDELINE: Familiarize yourself with the performance and optimizations available in commercial operating systems to see if an SMP-enabled OS or RTOS meets your performance and real time requirements.

In the AMP (Asymmetrical Multi-Processing) configuration, two different operating systems or two instances of a single operating system run on the two cores. Because the two instances of the operating systems have no inherent knowledge of how they

share CPU resources, there are several complexities that need to be taken into account in order to ensure the applications make efficient use of the resources available in your MPU subsystem.

Note: Use AMP only if you are familiar with the techniques to manage and schedule processes, handle inter-process communication, synchronize between events, manage secure processes between the two instances of the operating systems.

Note: OS providers do not generally offer support for using their OS in an AMP mode, so a special support agreement is typically needed in this case.

5.1.4 Assembling your Software Development Platform for Linux

This section presents design guidelines to be used when you have selected Linux as the OS for your end application.

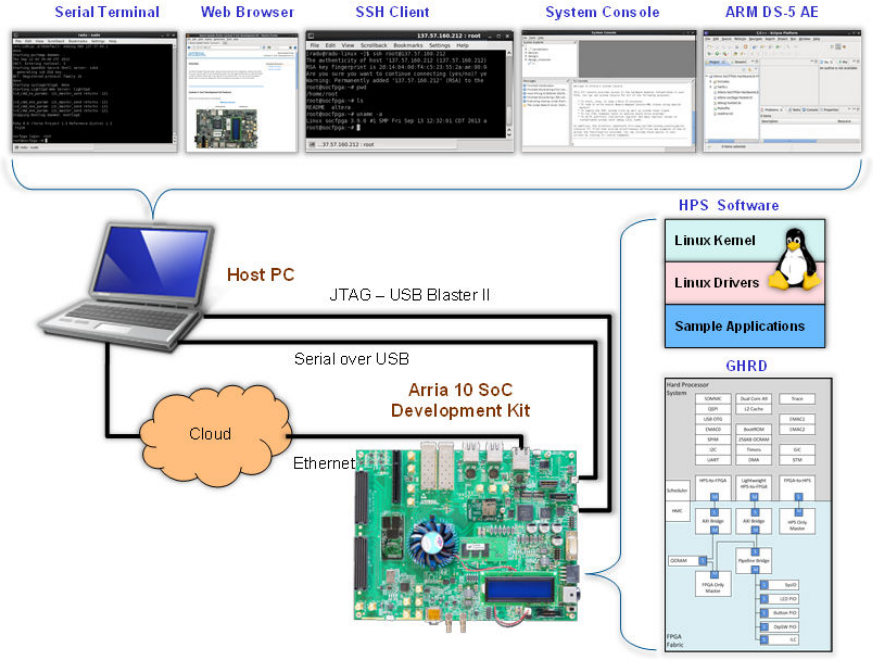
5.1.4.1 Golden System Reference Design (GSRD) for Linux

Intel provides the GSRD for Linux, which consists of the following:

- GHRD - A Quartus Prime project
- Reference U-Boot based Bootloader
- Reference Linux BSP
- Sample Linux Applications

The GSRD configuration is shown in the figure below.

Figure 15. GSRD for Linux - Overview



The GSRD for Linux is a well-tested known good design showcasing a system using both HPS and FPGA resources, intended to be used as a baseline project.

GUIDELINE: To successfully build your software development platform, Intel recommends that you use the GSRD as a baseline project.

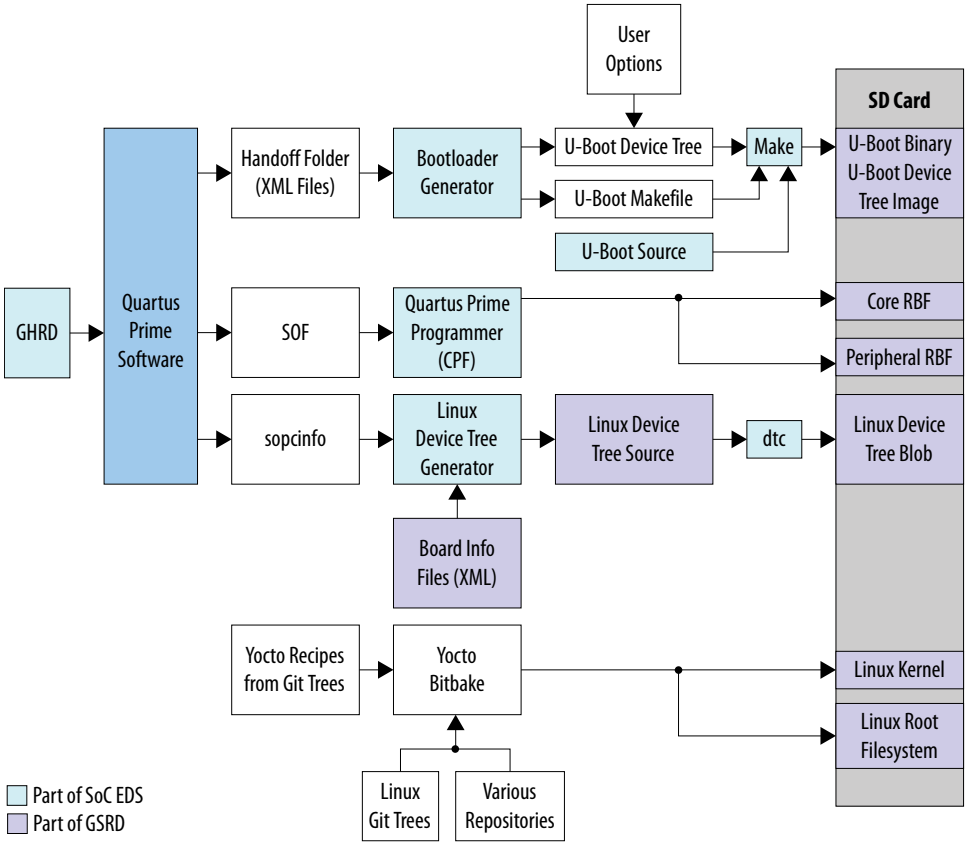
The GSRD, which targets the Intel SoC Development Boards, is provided both in source and pre-compiled form. Download the GSRD from RocketBoards.org, then modify it to suit your application needs.

GUIDELINE: It is recommended that all new projects use the latest version of GSRD as a baseline.

5.1.4.2 GSRD for Linux Build Flow

The figure below presents a detailed build flow for the GSRD. Refer to Golden System Reference Design (GSRD) User Manuals for more details.

Figure 16. GSRD for Linux - Build Flow



The above build flow is the one used for the GSRD for Linux but it can be tweaked to match the individual needs of each project. For example:

- Linux kernel could be built separately without using Yocto Project
- Linux filesystem could be built separately without using Yocto Project
- Linux Device Tree could be managed without using the Device Tree Generator. For example, it can be manually edited

5.1.4.3 Linux Device Tree Design Considerations

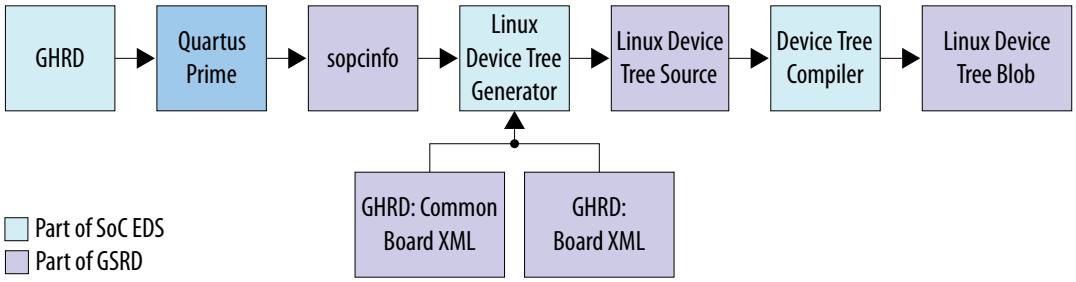
The Linux Device Tree is a data structure that describes the underlying hardware to the Linux operating system kernel. By passing this data structure the OS kernel, a single OS binary may be able to support many variations of hardware. This flexibility is particularly important when the hardware includes an FPGA.

The recommended procedure for managing the Linux Device Tree is:

1. Start with the SoC FPGA reference Device Trees provided in the Linux kernel source code that targets the Intel SoC Development Kits. They cover the HPS portion of the device but do not cover the FPGA portion which changes on a per-project basis. SD/MMC, QSPI and NAND versions are provided with the kernel source code.
2. Edit the Device Tree as necessary to accommodate any board changes as compared to the Intel SoC Development Kit.
3. Edit the Device Tree as necessary to accommodate the Linux drivers targeting FPGA Soft IP.

Note: The GSRD for Linux uses a different flow that the one recommended above relying on a custom tool called "Linux Device Tree Generator" that is provided as part of SoC EDS.

Figure 17. Device Tree Generation Flow for GSRD for Linux



Note: The Linux device tree generator is tested with and supports only the Linux kernel version targeted by the associated GSRD. Intel recommends against using the Linux device tree generator if your design targets a different Linux kernel version. Instead, manage the device tree manually. Use the device tree files provided with the kernel as a baseline, and add the FPGA IP and board information manually.

Refer to the Linux Device Tree Generator User Guide for more information.

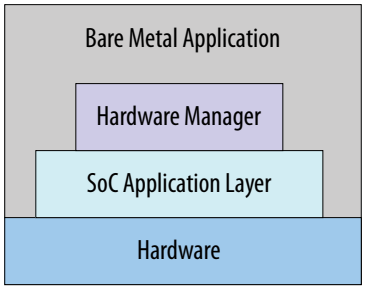
5.1.5 Assembling your Software Development Platform for a Bare-metal Application

The Intel hardware libraries (HWLibs) are collections of low level bare metal software utilities provided with SoC EDS and designed for controlling various components of the HPS. The HWLibs are also typically used by Intel's OS partners to build board support packages for operating systems.

The HWLibs have two components:

- SoC Abstraction Layer (SoCAL): Register abstraction layer that enables direct access and control of device registers within the HPS address space.
- Hardware Manager (HWMgr): APIs that provide more complex functionality and drivers for higher level use case scenarios.

Figure 18. HWLibs Overview



Note that not all hardware is covered by SoCAL and HWMgr, therefore writing custom code might be necessary depending on application

Software applications that use HWlibs should have run time provisions to manage the resources of the MPU subsystem, the cache and memory. These provisions are typically what operating systems provide.

GUIDELINE: It is recommended using HWlibs only if you are familiar with developing run time provisions to manage your application.

GUIDELINE: Use the HWLIBs examples from <SoC EDS installation folder>/embedded/examples/software/ as a starting point for your bare-metal development.

Note: Review the additional HWLIBs examples available at Design Examples.

Note: Refer to the SoC EDS User Guide for more information about HWLIBs

Related Links

Getting Started with HwLibs Baremetal Development
Complete set of guides for getting started with bare-metal development

5.1.6 Assembling your Software Development Platform for Partner OS or RTOS

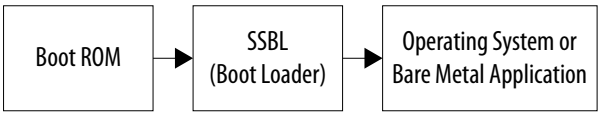
Partner OS providers offer board support packages and commercial support for the SoC FPGA devices. Typically the support includes example getting started projects and associated documentation.

Note: Please refer to the partner documentation and support services for information on how to assemble the software development platform when targeting a Partner OS or RTOS.

5.1.7 Choosing Boot Loader Software

The Arria 10 boot flow is depicted in the figure below:

Figure 19. Arria 10 Boot Flow



The boot loader software is one of the most important components of your software development platforms. The job of the Bootloader is to initialize the system and then load and pass control to the next boot image which is either an OS or a baremetal application.

The boot loader software is designed to run out of available HPS on-chip memory and provides essential initial hardware settings to configure the HPS as well as software features to control the flash and peripheral components of the HPS and utilities to enable early debugging and troubleshooting.

For instance, the U-boot boot-loader software configures the ECC registers in the HPS. The DDR SDRAM ECC configuration settings in the Arria 10 External Memory Interfaces for HPS IP component are translated into the bit stream peripheral raw binary file **(.rbf)**. The boot loader software extracts the ECC SDRAM configuration settings from the bit stream and sets up the ECC registers in the HPS.

A typical HPS system has hundreds of registers that must be set for a given configuration of the MPU subsystem, the network-on-chip interconnect component, the SDRAM memory, flash boot source and peripheral interfaces. The settings used for boot or initialization purposes are encapsulated in the following places:

- Raw Binary File (RBF) Files(s) - containing register settings for SDRAM also shared I/O and FPGA pin configuration.
- Bootloader Device Tree Structure (DTS) - containing user settings and some default settings for clock tree, I/O muxing and dedicated pin configuration (slew rate, pull up, pull down and some NoC settings)
- U-Boot source code - for rest of the settings

Note: The Bootloader Device Tree Structure is different from the Linux Device Tree Structure and the two should not be confused.

Intel provides two types of boot loader options:

- **U-Boot Boot Loader**: Inherits a number of features available from the open source community and is popular with Linux OS users and is governed by GPL licensing. Available as a part of SoC EDS and used by GSRD for Linux.
- **UEFI Boot Loader**: Feature rich and popular with RTOS users and are governed by an open-source BSD style license. It is available at the UEFI Bootloader Wiki.

The following figure represents the decision flow for selecting the best loader to suit your needs:

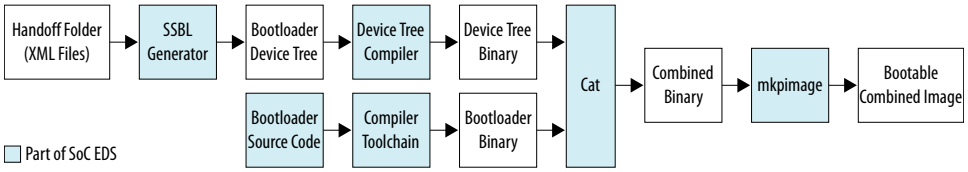
GUIDELINE: To select the right boot loader for your software development platform, familiarize yourself with the GPL and open-source BSD licenses and consider which licensing terms best suit your requirements.

GUIDELINE: It is recommended to use the the latest version of the boot loader software.

GUIDELINE: Given the amount of various initialization settings that are required, it is not recommended to write a bootloader from scratch. The provided bootloader options contain optimum and default configuration settings for various parts of the HPS.

The following figure represents the U-Boot Boot Loader flow which is similar for UEFI:

Figure 20. Bootloader Build Flow



5.1.8 Selecting Software Tools for Development, Debug and Trace

This section describes design considerations for selecting various software development tools.

Note: When using a specific Partner OS or RTOS, consult the OS vendor and the OS documentation for any specific tools that are required. Some OS vendors also provide a full set of tools that are recommended to be used with that OS.

Note: Familiarize yourself with the available tools for development, compilation and debug. A list of supported tools is available at the SoC Products Ecosystem link on Intel's website.

5.1.8.1 Selecting Software Build Tools

GUIDELINE: Decide which software development tools (compiler, assembler, linker archiver etc) will be used. Identify what version of the tools will be used.

The ARM DS-5 Intel SoC FPGA Edition includes the following software build tools:

- ARMCC Bare-metal Compiler
- Mentor Graphics CodeSourcery Lite GCC-based bare-metal compiler
- Linux Linaro compiler

There are also other development tools offerings from 3rd party providers.

5.1.8.2 Selecting Software Debug Tools

GUIDELINE: Decide which software debug tools will be used.

The ARM DS-5 Intel SoC FPGA edition includes a fully featured Eclipse-based debugging environment. There are also other debugging tool offerings from third party providers such as Lauterbach T32.

The debug tools require a JTAG connection to the SoC FPGA device. The connection could be achieved in a couple of ways:

- An embedded Intel FPGA Download Cable II could be available on-board like it is on the Arria 10 SoC Development Kit.
- External JTAG hardware may be required when using the Lauterbach T32 tools.

5.1.8.3 Selecting Software Trace Tools

Tracing can be very helpful for profiling performance bottlenecks, debugging crash scenarios and debugging complex cases. Tracing can be performed in two ways:

- **Non-real-time**: by storing trace data in system memory (e.g. SDRAM) or the embedded trace buffer, then stopping the system, downloading the trace information through JTAG analyzing it.
- **Real-time**: by using an external adapter to capture trace data from the trace port. The target board needs to support this scenario.

Typically, the debug tools also offer tracing of the embedded software program execution, but external hardware may be required. For example, the DS-5 provided with the SoC EDS supports both non-real-time and real-time tracing. When used for real-time tracing, an additional external trace unit called "DSTREAM" is required. Lauterbach T32 is a similar example, in that it needs additional external hardware for real-time tracing.

5.1.9 Board Bring Up Considerations

This section describes the considerations that are useful for board bring up.

Reserved BSEL Setting

During initial stages of bringup, if a JTAG connection cannot be established to the target, it may be beneficial to set BSEL to `0x0` "Reserved" setting to prevent the BootROM from trying to boot from a specific boot source. Then a test program could be downloaded and ran with a debugger.

Bootloader UART Logging

On initial power on, if the device is in unconfigured state, only the dedicated HPS I/O will be available to the processor. To have visibility into the boot process before the other device I/O are configured, dedicated I/O must be used to connect to the serial interface.

GUIDELINE: Ensure that you have connected a UART serial interface to the dedicated I/O in the HPS device in order to get visibility into the boot process. This option is not available for NAND.

5.1.9.1 Plan the SDRAM Initialization

GUIDELINE: Determine how the SDRAM will be initialized

The Arria 10 I/O must be configured before the SDRAM can be configured to be used by HPS. This goal can be achieved in a number of ways:

- The FPGA is fully configured externally before the HPS bootloader is executed
- The FPGA is fully configured by the HPS bootloader, then the bootloader brings up the SDRAM
- The FPGA I/O and shared I/O are configured by the HPS bootloader, and then the bootloader brings up the SDRAM. The FPGA fabric is configured at a later time. This option is called early I/O release and it not available on all devices. Refer to the Arria 10 Device Errata.

Determine which SDRAM initialization scenarios are to be supported and plan accordingly.

Note: Refer to Configuration on page 76 and FPGA Reconfiguration on page 44 sections of this document for more information.

5.1.10 Boot and Configuration Design Considerations

5.1.10.1 Boot Design Considerations

5.1.10.1.1 Boot Source

GUIDELINE: Determine which boot source is to be supported.

- The HPS of the Arria 10 SoC can be booted from a variety of sources:
  - SD/MMC Flash
  - QSPI Flash
  - NAND Flash
  - FPGA Fabric

  Note that more than one source can be supported. For example, most of the development could be done with an SD card, which is more convenient, and then the final testing and production release could target booting from QSPI.

Each possible boot source has its own considerations:

- SD cards are cheap, universally available, and have large storage capacities. Industrial versions are available, with improved reliability. They are managed NAND flash, so wear leveling and bad block management are performed internally.
- eMMC devices have smaller packages, are available in large capacities, and can be more reliable than SD. They are not removable, with can be a plus, allowing more rugged operation.
- QSPI devices are very reliable, typically with a minimum of 100,000 erase cycles per sector. However they have a reduced capacity as compared to the other options. They are typically used as a boot source, but not as an application filesystem.
- NAND devices are available in large sizes, but they are unmanaged NAND, which means that techniques such as wear leveling and bad block management must be implemented in software.
- FPGA boot allows the HPS to boot without the need of an external Flash device. The FPGA boot memory can be synthesized out of FPGA resources (typically pre-initialized embedded memory blocks) or can be memory connected to the FPGA such as an external SRAM or SDRAM. In order to boot from the FPGA, the FPGA must be configured using a traditional configuration mechanism.

5.1.10.1.2 Select Desired Flash Device

GUIDELINE: Select the boot flash device.

When choosing a flash device to incorporate with SoC FPGA devices, it is important to consider the following:

- **Will the flash device work with the HPS BootROM?**

  The HPS can only boot from flash devices supported in the BootROM
- **Is the device verified to work and supported by software like the Preloader, U-Boot and Linux?**

  For supported devices, Intel provides the Preloader, U-Boot and Linux software.

  For other devices, this software must be developed by the user.
- **Is the flash device supported by the HPS Flash Programmer?**

  The HPS Flash Programmer enables writing to flash using a JTAG connection, primarily to program the initial pre-loader/bootloader image.

  If the device is not supported by the HPS Programmer, other flash programming methods may be used, such as using the HPS to program flash. For example, the flash programming capabilities of U-Boot can be used.

Refer to the Supported Flash Devices for Arria 10 SoC web page for more information.

5.1.10.1.3 BSEL Options

GUIDELINE: Configure the BSEL pins for the selected boot source.

The boot source is selected by means of BSEL pins.

It may be beneficial to change the boot source for debugging purposes, even if the board does not have another available boot source. For example on a board booting from QSPI, it may be beneficial to select the reserved boot so that the BootROM would not do anything. Alternatively, you can select boot from FPGA and put a test image in the FPGA fabric.

If the system allows it (space constraints), plan to provide either switches or at least resistors to be able to change BSEL as needed.

5.1.10.1.4 Boot Clock

GUIDELINE: Determine boot clock source.

Configure the boot clock, as it influences the boot duration. This choice is primarily determined with the system boot time requirements. The system boot time requirements must take into account how fast the FPGA needs to be configured in order to be responsive and how fast the HPS software must be booted. The HPS software boot speed is influenced by the following factors:

- Value of external clock to the HPS (i.e. OSC1 clock)
- Boot flash source interface operational frequency
- Fuse options

The possible combinations are described in the Arria 10 HPS Technical Reference Manual and are selected with the CSEL pins.

Note: CSEL pins are not used when booting from FPGA fabric.

5.1.10.1.5 Determine Boot Fuses Usage

GUIDELINE: If possible, defer blowing fuses to the end of development, to allow more flexibility for debugging purposes. Once the fuses are blown, they cannot be returned to their original values. While in the development stage of the design, you may experiment with modifying CSEL values without having to permanently program the CSEL fuse bits, by using HPS_FUSESEC shadow registers as a proxy for CSEL fuse bits. In order to revert device to its deafult behavior simply power cycle the device.

GUIDELINE: Determine how boot fuses are to be configured.

Different fuses are available, and they can influence the boot flow. For example, if the boot from FPGA fuse is blown, the BSEL pins are not used to determine the boot source and the HPS boots exclusively from the FPGA fabric. This protocol can be useful in some security scenarios.

Refer to the Arria 10 HPS Technical Reference Manual for a complete list of fuses and their behavior and to determine in advance how they must be set for the project.

5.1.10.1.6 CSEL Options

GUIDELINE: Provide a method to configure CSEL options.

Note that for debugging purposes, it may be beneficial to allow use of the fuse shadow registers to use various CSEL values even if the end product requires just one CSEL setting. If possible, design the board in such a way that the CSEL configuration can be varied even if a single value will eventually be used. This configurability may be useful for debugging and could be done by resistors, jumpers or switches.

5.1.10.1.7 Determine Flash Programming Method

GUIDELINE: Ensure that the board is configured properly to support flash programming

The HPS Flash Programmer is a tool provided with SoC EDS that can be used to program QSPI and NAND flash devices on Arria 10 boards. The tool is intended to write relatively small amounts of data (for example the bootloader) because it works over JTAG and has limited speed.

If the HPS Flash Programmer tool is to be used, confirm that it supports the device you are planning to use. The supported devices are listed in the Altera SoC Embedded Design Suite User Guide.

Other ways to program the flash devices are:

- Program Flash using a debugger (for example ARM® DS-5)
- Program Flash from U-Boot
- Program Flash from Linux (or other OS) console
- Program Flash by means of dedicated hardware

5.1.10.1.8 Selecting NAND Flash Devices

GUIDELINE: Select a NAND flash that is ONFI 1.0 compliant.

When booting from NAND, ensure that the selected device is ONFI 1.0 compliant.

The NAND device used for booting must also have a x8 interface, and only a single pair of ce# and rb# pins. A x16 interface, and/or up to 4 pairs of ce# and rb# pins can only be used when NAND is used as a mass storage device, connected to the FPGA fabric I/O.

Although some non-ONFI 1.0 compliant devices are compatible with the BootROM, the HPS Flash Programmer only supports ONFI compliant devices.

The HPS Flash Programmer is expected to support NAND in the 16.1 and later releases.

5.1.10.1.9 Provide Flash Memory Reset for QSPI and SD/MMC/eMMC

GUIDELINE: Ensure that the QSPI and SD/MMC/eMMC devices have a mechanism to be reset when the HPS is reset.

The QSPI and SD/MMC/eMMC flash devices can potentially be put in a state by software where the BootROM cannot access them successfully, which may trigger a boot failure on the next reset. This problem can occur because the HPS is reset, but the flash part is not reset.

It is therefore required to reset the QSPI and SD/MMC/eMMC boot flash devices each time there is an HPS reset (warm or cold).

Note that some of the devices do not have a reset pin. In such a case you need to power cycle the flash using, for example, a MOSFET. Pay attention to the minimum required reset pulse duration.

5.1.10.1.10 Reference Materials

Refer to the following reference materials for additional information.

- Arria 10 Documentation including TRM, registers, pin connection guidelines.
- SoC Embedded Design Suite User Guide
- SoC Embedded Design Suite Getting Started Guides
- Golden System Reference Design
- Arria 10 SoC Boot User Guide

5.1.10.2 Configuration

The Arria 10 SoC devices support two main types of configuration flows:

- Traditional FPGA configuration
- HPS-initiated FPGA configuration

HPS-initiated configuration uses fast passive parallel (FPP) mode allowing the HPS to configure the FPGA using storage locations accessible to the HPS such as QSPI, SD/MMC and NAND flash. The FPGA configuration flows for the Arria 10 SoC are the same for the Arria 10 FPGA devices where an external configuration data source is connected to the configuration sub-system (CSS) block in the FPGA.

5.1.10.2.1 Traditional Configuration

If you use the traditional FPGA configuration flow where the FPGA is configured by an external source such as JTAG, active serial, or fast passive parallel then the HPS boot software must be configured to avoid configuring the FPGA and HPS Shared I/O. When the external source configures the FPGA all of the I/O except the HPS dedicated boot I/O are configured so the HPS second stage bootloader must be set up to not perform this role. If the HPS boots while the FPGA is being configured the bootloader waits until the FPGA enters user mode. It is important that once the FPGA is configured that it is not reconfigured since this action causes the HPS Shared I/O to temporarily go offline.

Note: Refer to the *FPGA Reconfiguration* section of this document for more information.

Related Links

FPGA Reconfiguration on page 44

5.1.10.2.2 HPS-Initiated Configuration

When the device is powered and the HPS begins executing the software in the boot ROM, all the device I/O default to an input tri-state mode of operation. The boot ROM configures the dedicated boot I/O based on the sampled BSEL pins. At this time during the boot process the only I/O that are configured are the HPS Dedicated I/O which are connected to the HPS flash device and all other I/O remain in the default input tri-state mode of operation. The second stage bootloader configures the remaining dedicated HPS boot I/O as well as the FPGA and HPS Shared I/O by either configuring the entire FPGA or just the HPS Shared I/O and HPS DDR I/O by configuring the early I/O release bitstream.

HPS Early I/O Release

The HPS-initiated configuration flows supports a feature called early I/O release. The early I/O release feature allows you to minimize the HPS boot time by configuring the device I/O independent of configuring the majority of the FPGA fabric.

This feature reduces the HPS boot time because the majority of the FPGA configuration file is used to configure FPGA fabric with a small portion being used to configure the I/O. The I/O portion of the configuration file is used to configure the HPS shared I/O, HPS SDRAM I/O and remaining FPGA I/O. The HPS dedicated I/O are not affected by the FPGA configuration bitstream.

By configuring the I/O independent of the FPGA fabric, the HPS SDRAM controller becomes functional so that the HPS boot loader can populate the SDRAM with the next stage boot contents. After the I/O has been configured, the rest of the FPGA fabric can be configured immediately or by an application after booting completes.

GUIDELINE: With early I/O release, use a fixed power supply for startup.

If you are using the early I/O release configuration flow, you cannot initially use SmartVID to power your device. Instead, you can use a fixed power supply until after the FPGA is configured. When the FPGA is configured, you can then enable SmartVID.

GUIDELINE: Since the dedicated HPS I/O are the only pins not affected by configuring the device I/O, it is important to place any interfaces that must remain functional through a configuration cycle in the dedicated I/O bank. If access to HPS peripherals and HPS SDRAM is desired during boot, then the device must be configured using HPS Early I/O release mode.

For example, if your system requires a UART to remain operational while the FPGA is being configured, you should use dedicated I/O for the UART. If the HPS boot source is going to be external flash then the flash must be connected to the dedicated HPS I/O so that they remain functional while the rest of the I/O is being configured. For other use cases refer to the FPGA Reconfiguration on page 44 section of this document.

HPS Initiated FPGA Core Configuration

If FPGA configuration fails (due to a bit error for example), the HPS shared and SDRAM I/Os are placed in an input tristate mode. If the HPS initiates FPGA configuration after an early I/O release, and configuration fails, any HPS access to a shared I/O peripheral or HPS SDRAM fails because of the tri-stated I/Os. HPS dedicated I/Os are not affected.

This behavior has no impact on full FPGA configuration using a Raw Binary File (`.rbf`) that contains both the FPGA core and configuration. In this case, the HPS can access HPS shared I/O and HPS SDRAM only after the device has been successfully configured.

GUIDELINE: Provide a configuration failure avoidance or recovery mechanism that does not rely on access to HPS shared I/Os or HPS SDRAM.

If software executes from SDRAM, you can avoid this issue by performing a configuration bitstream checksum in software before initiating configuration. For example, the `.rbf` can have a cyclic redundancy check (CRC), which software can validate before programming the file contents into the FPGA core.

GUIDELINE: Enable ECC to avoid correctable bitstream corruption issues.

If software executes entirely from on-chip RAM, and a configuration failure occurs, software can recover by reprogramming the peripheral configuration `.rbf` to reactivate the HPS shared and SDRAM I/Os.

After the I/Os have been reconfigured, software can reconfigure the FPGA fabric, either with the same image or with a fallback image (to prevent the same failure from recurring).

It is recommended that before configuring the rest of the FPGA fabric that software perform an integrity check of the bitstream first. The purpose of this integrity check is to avoid configuration issues due to bitstream corruption. If the bitstream is corrupt then configuration of the FPGA fabric will fail and all of the device I/O except the HPS Dedicated I/O will revert to an input tri-state mode of operation.

5.1.11 Flash Device Driver Design Considerations

The SoC FPGAs support the following types of flash devices: QSPI, NAND, SD/MMC/ eMMC.

Note: Please refer to Supported Flash Devices for Arria 10 SoC for a list of supported flash devices. Use an "Intel Tested and Supported" device if at all possible, to minimize any potential development effort and eliminate the risk of incompatibility. The next best option is to select a "Known to Work" device. This means that the device is compatible with BootROM and was proven to work with at least one Bootloader - but it may not be the Bootloader you need. It may also not have HWLIBs, OS support or HPS Flash Programmer support.

5.1.12 HPS ECC Design Considerations

ECC is implemented throughout the entire HPS on all RAMs, including the external HPS EMIF, L2 cache data RAMs and all peripheral RAMs. All HPS ECC controllers are based on an extended Hamming code algorithm, providing single-error correction and double-error detection (SECDED). Parity protection is provided for the ARM Cortex-A9 MPCore L1 cache memories and L2 tag RAM. ECC can be selectively enabled on the HPS EMIF and internal HPS RAMs. Diagnostic test modes and error injection capability are available under software control. ECC is disabled by default upon power-up or cold reset.

The generated boot code configures, initializes, and enables ECC according to user options selected during BSP generation. Custom firmware and bare metal application code access to the ECC features is facilitated with the Intel-provided HWLIBS library, which provides a simple API for programming HPS hardware features.

For more information, refer to *Boot Tools User Guide* and *Hardware Library* chapters of the Altera SoC Embedded Design Suite User Guide.

5.1.12.1 General ECC Design Considerations

Each RAM in the HPS has its own ECC controller with a unique set of features and requirements; however there are some general system integration design considerations.

GUIDELINE: Properly configure HPS ECC-related resets in the Reset Manager module.

The Reset Manager manages HPS ECC-related resets. Refer to Chapter 3: Reset Manager of the Arria 10 Hard Processor System Technical Reference Manual

GUIDELINE: Properly configure ECC Control, Status and Interrupt Management in the System Manager Module.

The System Manger contains a set of ECC-related registers for system-level control and status for all the ECC controllers in the HPS. ECC-related interrupts are also managed through this set of registers. Refer to System Manager chapter of the Arria 10 Hard Processor System Technical Reference Manual

GUIDELINE: Ensure that memories are initialized before enabling ECC.

Before enabling ECC for RAMs in your HPS, writes to the ECC protected memories are required to initialize the memory data content and ECC syndrome bits, otherwise spurious bit error interrupts are generated from reads out of uninitialized locations after ECC is enabled. Refer to the appropriate peripheral chapters of the Arria 10 Hard Processor System Technical Reference Manual for details on initialization procedures.

5.1.12.2 ECC for External SDRAM Interface

ECC on the HPS external SDRAM interface is implemented in the SDRAM Adapter. The SDRAM Adapter's ECC controller performs SECDED on both the address and data to ensure valid data accesses at the intended locations in memory.

Note: For details on the SDRAM Adapter's ECC controller, refer to System Interconnect chapter, Functional Description of the SDRAM Adapter section, and the "`ecc_hmc_ocp_slv_block`" Address Map under Register Definitions section of the Arria 10 Hard Processor System Technical Reference Manual.

GUIDELINE: Initialize All SDRAM Regions.

You must initialize all regions of the external SDRAM that any part of the SoC system may access through the SDRAM Adapter before enabling ECC, including program code and data regions and other regions of memory accessed by FPGA masters and L2 cache line fetches. Consider extended regions for initialization when the L2 cache is enabled, because cache line fetches could extend beyond the program code and data footprint, resulting in likely bit errors from uninitialized RAM locations. You can define and configure memory regions for ECC scrubbing with the bsp-editor utility in the Altera SoC EDS tool chain.

GUIDELINE: Consider SDRAM Adapter ECC Write Behavior.

For partial writes (less than the SDRAM memory interface width), the adapter implements a read-modify-write sequence to maintain the ECC bits for the entire interface word width. Consider how narrow accesses might affect bandwidth and latency for your application while writing firmware and application software as well as accesses from other masters in the system.

Refer to the Arria 10 SX Device Errata for information about using the EMIF with 16-bit data and ECC enabled.

5.1.12.3 ECC for L2 Cache Data Memory

The L2 cache memory is ECC protected, while the tag RAMs are parity protected. L2 cache ECC is enabled through a control register in the System Manager.

Note: For details on the L2 cache ECC Controller, refer to the following sections of the Arria 10 Hard Processor System Technical Reference Manual, Chapter 9: Cortex-A9 Microprocessor Unit Subsystem: "Single Event Upset Protection" under the L2 Cache section, and "L2 Cache Controller Address Map for Arria 10" under the Cortex-A9 MPU Subsystem Register Implementation section.

GUIDELINE: The L1 and L2 cache must be configured as write-back and write-allocate for any cacheable memory region with ECC enabled.

For BSPs supported through the Intel SoC EDS, you can configure your BSP for ECC support with the bsp-editor utility. For baremetal firmware, refer to the Arria 10 Hard Processor System Technical Reference Manual, chapter on Cortex-A9 Microprocessor Unit Subsystem, L2 Cache Controller Address Map for Arria 10 section.

GUIDELINE: Cache-coherent accesses through the L3 interconnect using the ACP must perform 64-bit wide, 64-bit aligned write accesses when ECC is enabled in the L2 Cache Controller.

Enabling ECC does not affect the performance of the L2 cache, but accesses using the ACP must be 64-bit wide, 64-bit aligned in memory. This includes FPGA masters accessing the ACP over the FPGA-to-HPS Bridge. Refer to the Arria 10 Hard Processor System Technical Reference Manual, chapter covering HPS-FPGA Bridges, FPGA-to-HPS Access to ACP section, Table 8-3 for a list of possible combinations of bridge width and FPGA master width, alignment and burst size and length.

5.1.12.4 ECC for Flash Memory

All peripheral RAMs in the HPS are ECC protected. Each peripheral has its own instance of the Error Checking and Correction Controller detailed in Chapter 11 of the Arria 10 Hard Processor System Technical Reference Manual

GUIDELINE: Software must update the ECC during NAND read-modify-write operations.

The NAND flash controller ECC hardware is not used when a read-modify-write operation is performed from the flash device's page buffer. Software must update the ECC during such read-modify-write operations. For a read-modify-write operation to work with hardware ECC, the entire page must be read into system memory, modified, then written back to flash without relying on the flash device's read-modify-write feature.

GUIDELINE: Consider that the NAND flash controller cannot do ECC validation during a copy-back command.

The NAND flash controller cannot do ECC validation during a copy-back command. The flash controller copies the ECC data but does not validate it during the copy operation.

5.1.13 Security Design Considerations

The Arria 10 SoC provides a framework for implementing secure systems through a layered hardware and software solution. When designing a secure system, you can implement several security levels depending on your system's security requirements.

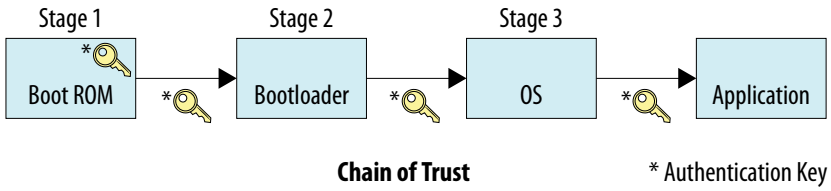
GUIDELINE: Determine which parts of your design must be encrypted. Determine which parts of your design must be authenticated.

Secure Boot – Chain Of Trust and Image Authentication

Secure Boot ensures that a Chain of Trust is established for all boot stages. Each boot stage must authenticate subsequent stages prior to loading and executing by verifying the image's signed certificate.

The boot stages can span from the initial Second Stage Bootloader to the final application loaded by the OS.

Figure 21. Secure Boot - Chain of Trust and Image Authentication

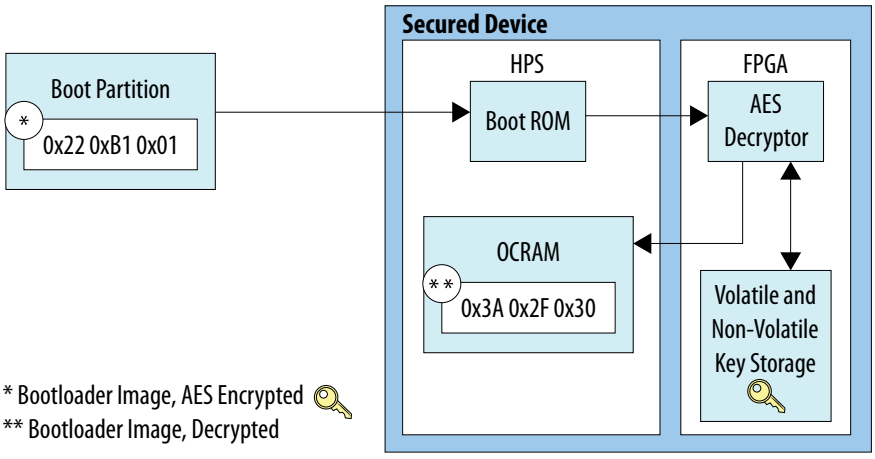


Refer to the Arria 10 SoC Secure Boot User Guide for more information.

Securing the Design IP - AES Encryption

To secure the FPGA design IP, use AES encryption. Encrypt the design IP before storing it on the intended boot device storage area. If the AES security keys are verified by the SoC, then the image is decrypted during configuration load time.

Figure 22. AES Encryption



Secured Boot and IP - Authentication and Encryption

This level offers the most security because all runtime SW and Data IP is authenticated and successfully decrypted during system bring up.

Anti-Tamper

This Security Level uses a defined logic to post notification when an attempt to tamper the device has been detected.

5.1.14 Embedded Software Debugging and Trace

This device has just one JTAG port with FPGA and HPS JTAGs chained.

GUIDELINE: It is recommended to have an available JTAG connection to the board that could be used for development as well as to debug and diagnose field issues.

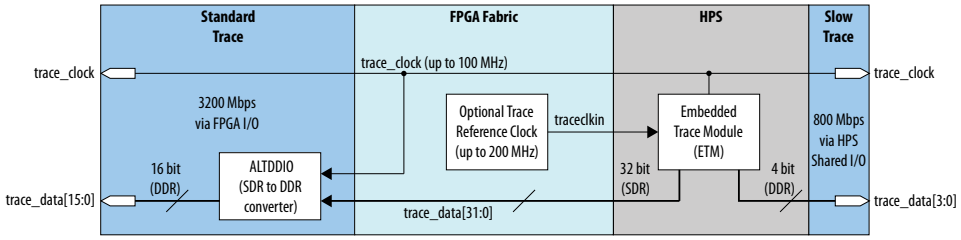
The HPS offers two trace interfaces either through HPS Shared I/O or FPGA I/O. The interface through HPS Shared I/O is a slow trace interface that you can use to trace low bandwidth traffic (such as the MPU operating at a low frequency). Because the interface is only a four-bit DDR interface, it provides limited bandwidth.

To improve the trace bandwidth, you can use the standard trace interface which is a 32-bit single data rate interface to the FPGA. Since trace modules typically expect trace data to be sent at a double data rate you need to convert the single data rate trace data to double data rate.

It is recommended that you instantiate the DDIO megawizard IP and set it up in output only mode to perform this conversion. The lowest 16 bits of trace data must be sent off chip first so you connect those bits to the `datain_l[15:0]` port of the DDIO IP.

Consult your trace vendor's datasheet to determine if the trace bus requires termination. Failure to include termination when the trace vendor requires it can lead to trace data corruption or limit the maximum operating frequency of the interface.

Figure 23. Trace Diagram



5.2 Support and Documentation

5.2.1 Support

Technical support for operating system board support packages (BSPs) that are ported to the Intel SoC development kit is provided by the operating system provider.

Support for the Intel SoC Embedded Development Suite (EDS) and the design tools for FPGA development is provided by Intel. The EDS includes the ARM Development Studio 5 (DS-5) Intel Edition Toolkit.

Support of the Intel development kit is provided by Intel.

Technical support for other boards is provided by the respective board provider or distributor.

Table 11.

Item	Supported Through
Commercial OS/Tools	OS/Tools Vendor
Hardware libs (baremetal)	Intel
Intel EDS	Intel
DS-5 Intel SoC FPGA Edition	Intel
FPGA Design Tools	Intel
Open Source/ Linux	RocketBoards.org
UBoot	RocketBoards.org

For additional information, please refer to these links:

- **SoC Ecosystem** in the **Intel FPGAs** page
- **Support Center**
- **End Market Related Designs**
- **Intel FPGA Design Solutions Network**
- **Embedded Support Center**
- **Design Examples**
- **Development Kits, Daughter Cards & Programming Hardware Support**
- **Documentation**
- **Altera Wiki**

5.2.2 Hardware Documentation

For more hardware information, please refer to the links below:

- Arria 10 Hard Processor System Technical Reference Manual
- Arria 10 Device Datasheet
- Arria 10 SoC Secure Boot User Guide
- Arria 10 SX Device Errata
- Arria 10 GX, GT, and SX Device Family Pin Connection Guidelines
- Arria 10 HPS Pin Information
- Arria 10 EMIF Pin Information
- Arria 10 Device Pin-Out Files

5.2.3 Software Documentation

Driven by the feature-updating nature of today's open source embedded software, most of our software documentation is also hosted on the community web site RocketBoards.org.

For more information, please refer to the links below:

- RocketBoards.org **Documentation Portal**
- RocketBoards.org SoC Linux Code
- RocketBoards.org **Boards**
- RocketBoards.org **Projects**
- *Intel SoC FPGA Embedded Design Suite User Guide*
- **Getting Started** on the **Intel SoC FPGA Embedded Design Suite** page

A Additional Information

This chapter provides additional information about the document and Intel.

A.1 Document Revision History

Table 12. Arria 10 SoC Device Design Guidelines History

Version	Release Date	Description
2017.05.08	May 2017	Added: • New chapter: *Guidelines for Interconnecting the Arria 10 HPS and FPGA* • Guidance about Standard and Pro editions of Quartus Prime and the SoC EDS • Information about voltage level support in *Design Guidelines for HPS Portion of Arria 10 SoC FPGAs* chapter • New section "Using the Bootloader as a Bare-Metal Framework" in the *Embedded Software Design Guidelines for Arria 10 SoC FPGAs* chapter
2016.09.16	September 2016	Initial Release

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2008 Registered