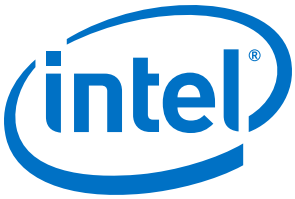


AN 802: Intel® Stratix® 10 SoC Device Design Guidelines

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

AN-802 | 2018.12.24
Latest document on the web: PDF | HTML

Contents

1. Introduction to the Intel® Stratix® 10 SoC Device Design Guidelines

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® Stratix® 10 SoC FPGA devices. This document assists you in the planning and early design phases of the Intel Stratix 10 SoC FPGA design, Platform Designer sub-system design, board design and software application design.

This application note does not include all the Intel Stratix 10 Hard Processor System (HPS) device details, features or information on designing the hardware or software system.

For more information about the Intel Stratix 10 HPS features and individual peripherals, refer to the *Intel Stratix 10 Hard Processor System Technical Reference Manual*.

Note: Intel recommends that you use Intel Quartus® Prime Pro Edition and the Intel SoC FPGA Embedded Development Suite Pro to develop Intel Stratix 10 SoC designs. Although Intel Quartus Prime Standard Edition and the Intel SoC FPGA Embedded Development Suite Standard continue to support the Intel Stratix 10 SoC family on a maintenance basis, future enhancements are going to be supported with the Pro software, only. Hardware developed with Intel Quartus Prime Pro Edition only supports software developed with the Intel SoC FPGA Embedded Development Suite Pro.

Hardware developed with Intel Quartus Prime Standard Edition only supports software developed with the Intel SoC FPGA Embedded Development Suite Standard.

Related Information

Intel Stratix 10 Hard Processor System Technical Reference Manual

1.1. Introduction to the Stratix 10 SoC Device Design Guidelines Revision History

Table 1. Introduction to the Stratix 10 SoC Device Design Guidelines Revision History

Document Version	Changes
2018.09.24	Maintenance release
2017.11.06	Initial release

2. Board Design Guidelines for Stratix 10 SoC FPGAs

2.1. HPS Clocking and Reset Design Considerations

The main clock and reset sources for the HPS are:

- HPS_OSC_CLK device I/O pin—The external clock source for the HPS PLLs, which generate clocks for the MPU Subsystem, CCU, SMMU, L3 Interconnect, HPS peripherals and HPS-to-FPGA user clocks.
- nCONFIG device I/O pin—An SoC device-wide reset input that reconfigures the FPGA and cold resets the HPS.
- HPS_COLD_nRESET device I/O pin—An optional reset input that cold resets only the HPS and is configured for bidirectional operation.

GUIDELINE: You can configure the HPS_COLD_nRESET pin to be on any open SDM I/O pin.

From Intel Quartus Prime,

1. Click **Assignments ➤ Device**.
2. Click the "Device and Pin Options" button.
3. Go to the "Configuration" tab.
4. Click the "Configuration Pin Options" button.
5. Click the "USE_HPS_COLD_nRESET" check box and select available SDM_IO pin.

For more information, refer to the “Pin Features and Connection for HPS Clocks, Reset and POR.” section.

2.1.1. 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 depends on board-level clock planning, clock planning for the FPGA portion of the device, and HPS peripheral external interface planning. Therefore, it is important to validate your HPS clock configuration before finalizing your board design.

GUIDELINE: Verify the MPU and peripheral clocks using Platform Designer.

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

2.1.2. Early Pin Planning and I/O Assignment Analysis

The HPS clock input resides in the HPS Dedicated I/O Bank shared with I/O from HPS peripherals such as Ethernet, mass storage flash, and UART console. It's location within this bank is user configurable.

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

The HPS Dedicated I/Os are LVCMOS/LVTTL supporting a 1.8V voltage level. Make sure any HPS peripheral interfaces (for example: Ethernet PHY, UART console) configured to use the HPS Dedicated I/O bank as well as board-level clock circuitry for the HPS are compatible with 1.8V LVCMOS signaling.

2.1.3. Pin Features and Connections for HPS Clocks, Reset and PoR

The HPS clock pin and optional reset pin have certain functional behaviors and requirements that you should consider when planning for and designing your board-level reset logic and circuitry.

GUIDELINE: Choose a pin location for the HPS clock input.

The HPS_OSC_CLK can be located anywhere within the HPS Dedicated I/O Bank. Use the HPS Platform Designer component to select the pin for HPS_OSC_CLK and verify its compatibility with other HPS peripheral I/O locations assigned to this bank.

GUIDELINE: Observe the minimum assertion time specifications of nCONFIG and HPS_COLD_nRESET.

Reset signals on the nCONFIG and HPS_COLD_nRESET pins must be asserted for the minimum time specified in the HPS section of the *Intel Stratix 10 Device Datasheet*.

GUIDELINE: Do not connect HPS_COLD_nRESET to SDM QSPI reset.

HPS_COLD_nRESET is a bi-directional pin that is input to the SDM to initiate a cold reset procedure to the HPS and its peripherals. The HPS_COLD_nRESET output can be used to reset any other devices on the board that should be reset when the HPS is reset. However, the SDM handles reset for the QSPI through software. Connecting HPS_COLD_nRESET to the SDM QSPI reset can result in undefined system behavior.

2.1.4. Direct to Factory Pin Support for Remote System Update (RSU) Feature

Intel Stratix 10 SoC supports the Remote System Update (RSU) feature. When you use this feature, you have the option to store multiple production images alongside a failsafe factory image on the external SDM flash. Upon exiting POR, SDM attempts to load the production images in your specific sequence. If all the production images fail to load, then the failsafe factory image is loaded.

GUIDELINE: Use the Direct to Factory Image pin to instruct the SDM to load either Factory or Application Image when exiting POR.

The Direct to Factory Image is an optional pin that can be used with the RSU feature. (1) If this pin is asserted during POR, the SDM does not attempt to load any production image; instead, the SDM loads the Factory Image directly from the external SDM flash.

2.1.5. 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 HPS PLLs and clocks under software control.

For more information, refer to the "Clock Manager" section in the *Intel Stratix 10 Hard Processor System Technical Reference Manual*.

Related Information

Intel Stratix 10 Hard Processor System Technical Reference Manual
For more information about the required software flow, refer to the "Clock Manager" chapter and the specific peripheral and subsystem chapters.

2.1.6. 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 section.

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

For more information about the required software flow, refer to the "Reset Manager" section in the *Intel Stratix 10 Hard Processor System Technical Reference Manual*.

Related Information

Intel Stratix 10 Hard Processor System Technical Reference Manual

(1) Both factory and application images are stored in the SDM flash.

2.2. 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 Intel Stratix 10 SoC devices.

1. HPS Dedicated I/O

These 48 I/O are physically located inside the HPS, are dedicated for the HPS, and are used for the HPS clock and peripherals, including mass storage flash memory.

2. SDM Dedicated I/O

The SDM has 24 dedicated I/Os, which include JTAG, clock, reset, configuration, reference voltages, boot and configuration flash interfaces, and MSEL.

3. HPS EMIF I/O

There are 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 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. It is possible to export most HPS peripheral interfaces to the FPGA fabric for custom adaptation and routing to FPGA I/O.

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

Table 2. Summary of SoC-FPGA I/O Types

	Dedicated HPS I/O	Dedicated SDM I/O	HPS EMIF I/O	FPGA I/O
Number of Available I/O	48	24	3 I/O 48 banks	All other device I/O
Location	Inside the HPS	Inside the SDM	FPGA I/O Banks 2L, 2M, 2N	I/O Columns are in the FPGA device
Voltages Supported	1.8V	1.8V	LVDS I/O bank support for DDR3 and DDR4 protocols	LVDS I/O bank, 3V I/O bank and high-speed serial transceivers
Purpose	HPS Clock, HPS peripherals, mass storage flash, HPS JTAG	FPGA JTAG through SDM dedicated pins, clock, reset, configuration, reference voltages, boot and configuration flash interfaces	HPS main memory	General purpose and transceiver I/O
Timing Constraints	Fixed	Fixed	Provided by memory controller IP	User defined
Recommended Peripherals	HPS peripheral I/O such as Ethernet PHY, USB PHY, mass storage flash (NAND, SD/MMC), TRACE debug.	Boot and configuration source, FPGA JTAG through SDM dedicated pins, and MSEL signals are connected to the SDM.	DDR3, DDR4, and SDRAM	Slow speed HPS peripherals (I^2C, SPI, EMAC-MII), FPGA I/O such as FPGA EMIFs, High Speed LVDS I/O, transceiver I/O, other parallel and control/ status I/O.

For more information about console output during boot and configuration, refer to the "UART Interface Design Guidelines" section.

Related Information

UART Interface Design Guidelines on page 25

2.2.1. HPS Pin Multiplexing Design Considerations

There is a total of 48 dedicated HPS I/O pins. The HPS component in Platform Designer offers pin multiplexing settings as well as the option to route most of the peripherals into the FPGA fabric.

GUIDELINE: Route the USB, EMAC and Flash interfaces to the HPS Dedicated I/O first, starting with USB.

Intel recommends that you start by routing high speed interfaces such as USB, Ethernet, and flash to the Dedicated I/O first.

2.2.2. HPS I/O Settings: Constraints and Drive Strengths

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

The HPS pin location assignments are managed automatically when you generate the Platform Designer system containing the HPS. Likewise, timing and I/O constraints for the HPS SDRAM interface are managed by the Intel Stratix 10 External Memory Interfaces for HPS IP. You must manage the following I/O constraints for the HPS Dedicated I/O using the Intel Quartus Prime software in the same way as for FPGA I/O: drive strength, weak pull-up enables, and termination settings. Any peripherals configured to use FPGA I/O must also be fully constrained, including pin locations, using the Intel Quartus Prime software.

2.3. Design Guidelines for HPS Interfaces

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

2.3.1. HPS EMAC PHY Interfaces

There are three EMACs based on the Synopsys* DesignWare* 3504-0 Universal 10/100/1000 Ethernet MAC IP version. When configuring an HPS component for EMAC peripherals within Platform Designer, you must select from one of the following supported PHY interfaces, located in the HPS Dedicated I/O Bank(2), for each EMAC instance:

- Reduced Media Independent Interface (RMII)
- Reduced Gigabit Media Independent Interface (RGMII)

(2) The HPS Dedicated I/O Bank consists of 48 I/O with 1.8V signaling.

Note: RGMII- Internal Delay (RGMII-ID) is a component of the RGMII version 2.0 specification that defines the capability of a transmitter to add delay on the forwarded data path clock to center that clock in the data. It adds delay on both: TX_CLK at the transmitter (MAC) and RX_CLK at the transmitter (PHY). Pin muxes in the HPS Dedicated I/O Bank feature the ability to add up to 2.4ns of delay in increments of 150ps, which is more than the required 1.5ns to center the clock in the transmit data at the PHY.

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

The Stratix 10 SoC Development Kit uses the Microchip KSZ9031 Ethernet PHY. This device is known to work with the Stratix 10 HPS Ethernet PHY interface and software device drivers.

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 using soft adaptation logic in the FPGA and features in the general-purpose FPGA I/O and transceiver FPGA I/O.

For more information, refer to the device drivers available for your operating system of choice or the Linux device driver provided with the Intel Stratix 10 SoC development kit.

The EMAC provides a variety of PHY interfaces and control options through the HPS and the FPGA I/Os.

For designs that are pin-limited on HPS I/O, the EMAC can be configured to expose either a GMII or MII PHY interface to the FPGA fabric, which can be routed directly to FPGA I/O pins. Exposing the PHY interface to the FPGA fabric also allows adapting the GMII/MII to other PHY interface types such as SGMII, RGMII and RMII using soft logic with the appropriate general purpose or transceiver I/O resources.

Note: You can 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.

The EMAC also provides the I^2C instead of the MDIO for their control interface. The HPS or FPGA can use three of the five general purpose I^2C peripherals for controlling the PHY devices:

- `i2c_emac_0`
- `i2c_emac_1`
- `i2c_emac_2`

Related Information

Golden System Reference Design

2.3.1.1. PHY Interfaces

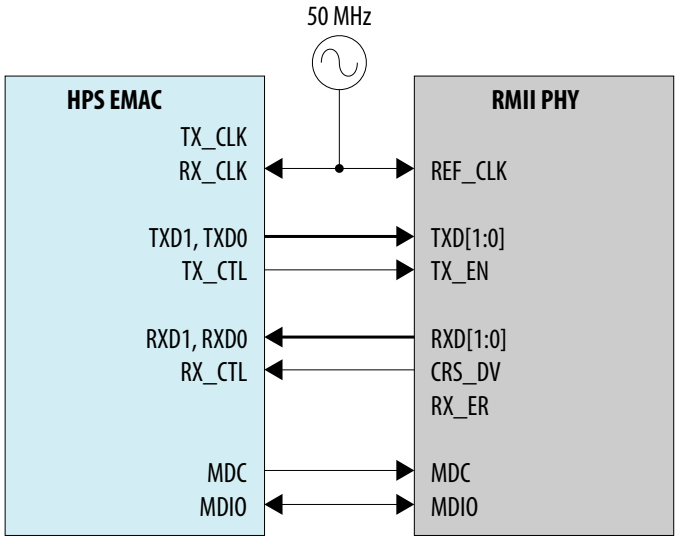
Considerations for RMII and RGMII PHY interfaces

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 data paths. 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 1. RMII MAC/PHY Interface



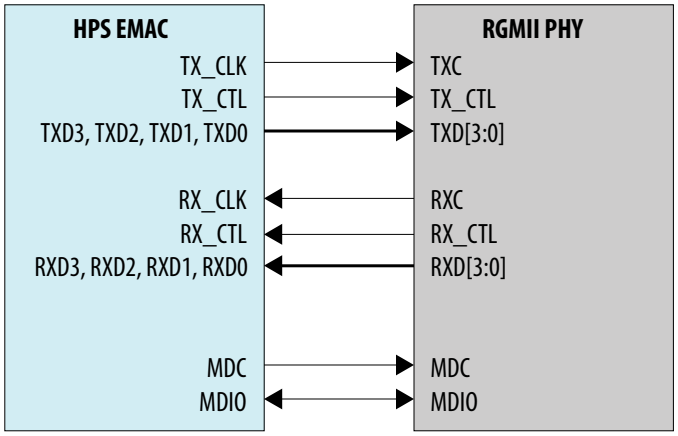
RGMII

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 data paths, 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 2. RGMII MAC/PHY Interface

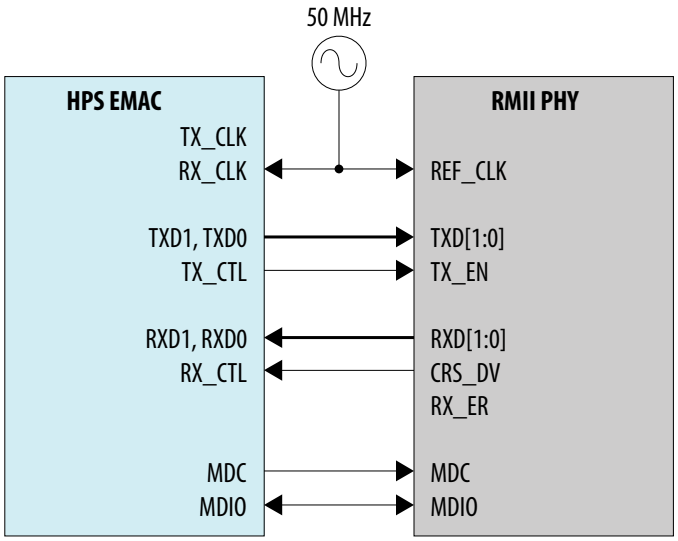


2.3.1.1.1. RMII

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 data paths. 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 3. 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_OSC_CLK 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.

Note: 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 4. 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.

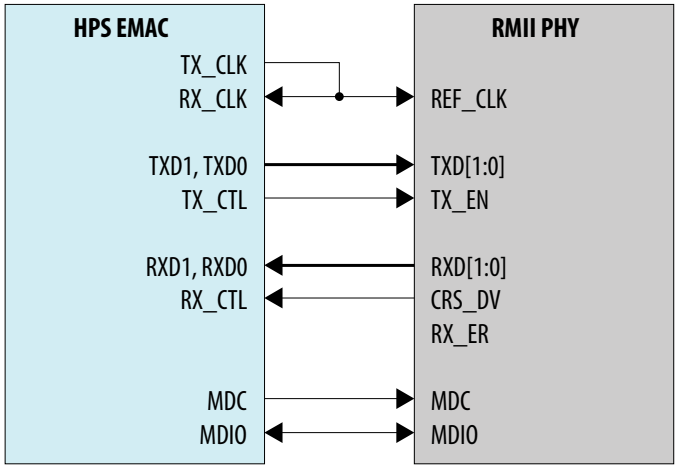
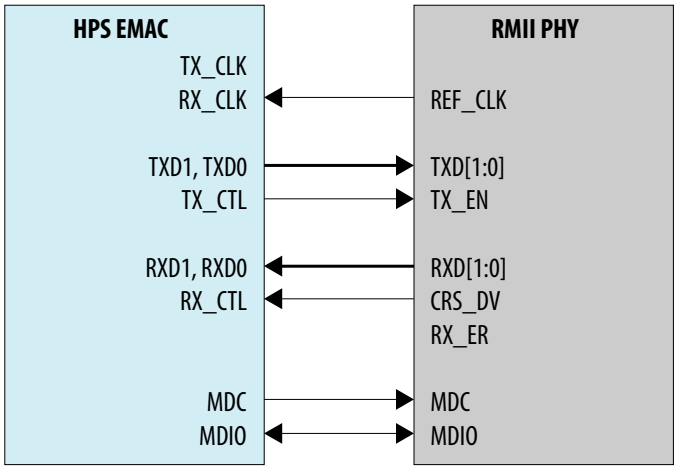


Figure 5. PHY Sourced REF_CLK

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. 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

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 you 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 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 recommends 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 Intel Stratix 10 I/O can provide up to 2.4 ns additional delay on outputs in 150 ps increments. This delay is enabled using the output delay logic option within the assignment editor in Intel Quartus Prime.

GUIDELINE: For TX_CLK from the Stratix 10, you must introduce 1.8 ns I/O delay to meet the 1.0 ns PHY minimum input setup/hold time in the RGMII spec.

The Stratix 10 SoC HPS dedicated I/O and FPGA I/O support adding up to 2.4 ns of output delay in 150 ps increments. The delay added to the MAC's TX_CLK output when using HPS dedicated I/O can be configured in the HPS Platform Designer IP component.

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

On the Intel Stratix 10 SoC Development Kit and the associated Intel Stratix 10 Golden Hardware Reference Design (the GHRD is the hardware component of the GSRD), an example for setting the output delay setting on TX_CLK can be found in the HPS Platform Designer IP component configuration.

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 Intel Stratix 10 I/O can provide up to 3200 ps additional delay on inputs. For Intel Stratix 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: If the PHY does not support RGMII-ID, use the configurable delay elements in the Stratix 10 SoC HPS dedicated I/O or FPGA I/O to center the RX_CLK in the center of the RX_DATA/CTL data valid window.

If using HPS I/O, configure delay on the RX_CLK in the HPS Platform Designer IP component. If using FPGA I/O, add delay on the RX_CLK input with an input delay setting in the project settings file (.qsf).

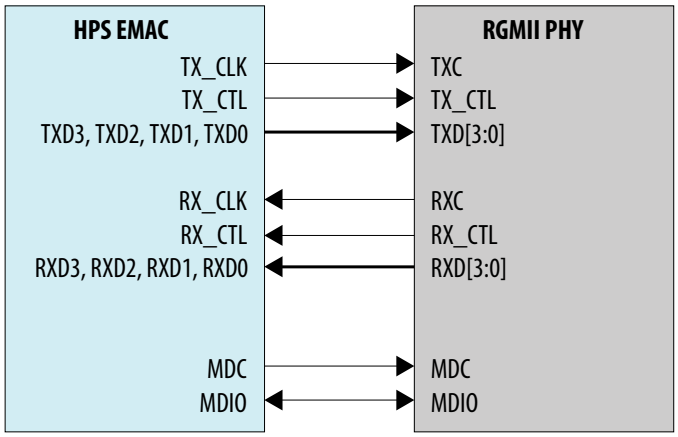
2.3.1.1.2. RGMII

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 data paths, 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 6. 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 you 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 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 Intel recommends 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 Intel Stratix 10 I/O can provide up to 2.4 ns additional delay on outputs in 150 ps increments. This delay is enabled using the output delay logic option within the assignment editor in Intel Quartus Prime.

GUIDELINE: For TX_CLK from the Stratix 10, you must introduce 1.8 ns I/O delay to meet the 1.0 ns PHY minimum input setup/hold time in the RGMII spec.

The Intel Stratix 10 SoC HPS dedicated I/O and FPGA I/O support adding up to 2.4 ns of output delay in 150 ps increments. The delay added to the MAC's TX_CLK output when using HPS dedicated I/O can be configured in the HPS Platform Designer IP component.

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

On the Intel Stratix 10 SoC Development Kit and the associated Intel Stratix 10 Golden Hardware Reference Design (the GHRD is the hardware component of the GSRD), an example for setting the output delay setting on TX_CLK can be found in the HPS Platform Designer IP component configuration.

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 Intel Stratix 10 I/O can provide up to 3200 ps additional delay on inputs. For Intel Stratix 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: If the PHY does not support RGMII-ID, use the configurable delay elements in the Stratix 10 SoC HPS dedicated I/O or FPGA I/O to center the RX_CLK in the center of the RX_DATA/CTL data valid window.

If using HPS I/O, configure delay on the RX_CLK in the HPS Platform Designer IP component. If using FPGA I/O, add delay on the RX_CLK input with an input delay setting in the project settings file (`.qsf`).

2.3.1.2. 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 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 Platform Designer.

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 Platform Designer system generation.

2.3.1.2.1. GMII/MII

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

GUIDELINE: Apply timing constraints and verify timing with Timing Analyzer.

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, recommends 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.

2.3.1.2.2. 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 Platform Designer adapter IP.

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

Configure the HPS component in Platform Designer 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 Platform Designer. Instead, add the Intel GMII to RGMII Adapter IP to the Platform Designer 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 Platform Designer 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.

2.3.1.2.3. 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.

2.3.1.2.4. Adapting to SGMII

You can use the GMII-to-SGMII Adapter core to adapt the GMII HPS EMAC PHY signals to an Serial Gigabit Media Independent Interface (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 Platform Designer adapter IP.

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

Configure the HPS component in Platform Designer 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 Platform Designer. Instead, add the Intel GMII to SGMII Adapter IP to the Platform Designer 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 Platform Designer 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 (that is, no soft MAC component). For more information about how to use the Intel GMII to SGMII Adapter IP, refer to the *Embedded Peripherals User Guide*.

GUIDELINE: Since the TSE MAC IP with 1000BASE-X PCS option no longer provides an option for the transceiver I/O, to implement an SGMII PHY interface using the FPGA transceiver I/O for an Intel Stratix 10 HPS EMAC instance, you must select "NONE" for the PCS I/O option, which gives you a TBI interface. The transceiver PHY IP must be separately instanced and connected in Intel Stratix 10.

Related Information

Embedded Peripherals IP User Guide

2.3.1.3. MDIO

The Management Data Input/Output (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.

2.3.1.4. Common PHY Interface Design Considerations

2.3.1.4.1. Signal Integrity

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

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

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

Only a few PHYs offer I/O tuning for their outputs, so recommends that you verify the signal path to the Intel Stratix 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 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.

2.3.2. USB Interface Design Guidelines

The Intel Stratix 10 HPS can connect its embedded USB MACs directly to industry-standard USB 2.0 ULPI PHYs using the 1.8 V dedicated HPS I/O. 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: Intel recommends 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.

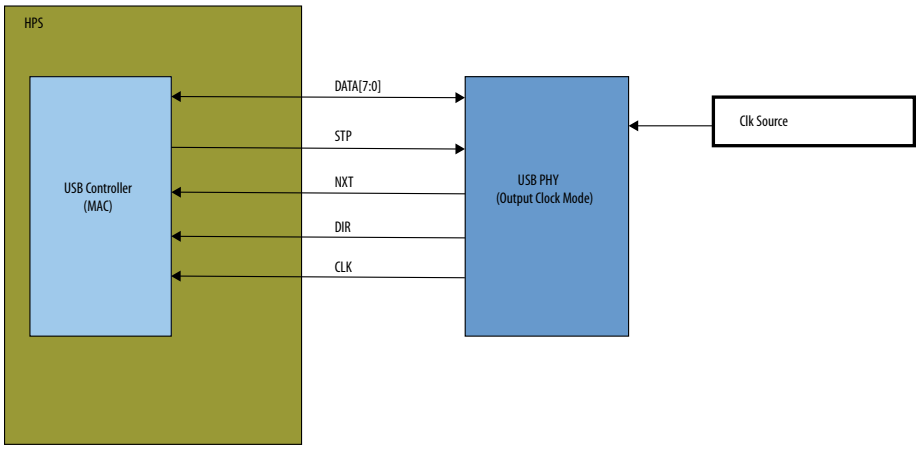
The Stratix 10 SoC Development Kit uses the Microchip USB3320 USB PHY. This device is known to work with the HPS USB module.

The interface between the ULPI MAC and PHY on the Stratix 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, Intel recommends 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.

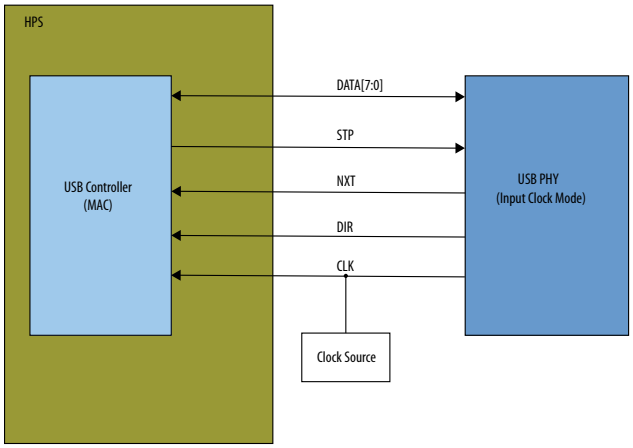
- Output Mode—In output clock mode, the clock is generated by the USB PHY. All signals are synchronized to this clock.

Figure 7. Output Mode



- Input Mode—In input clock mode, the PHY receives a clock from an external source. All signals are synchronized to the clock. In this mode, the clock can be generated by a PLL in the FPGA or by an external source.

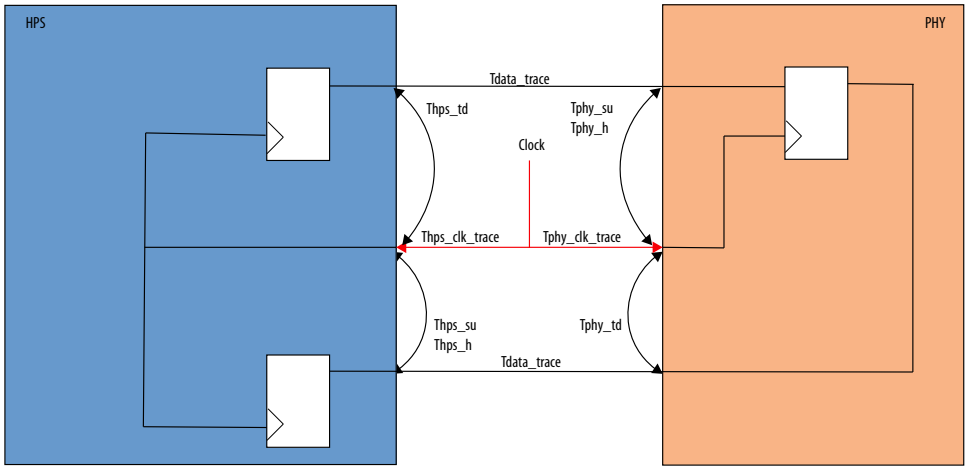
Figure 8. Input Mode



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

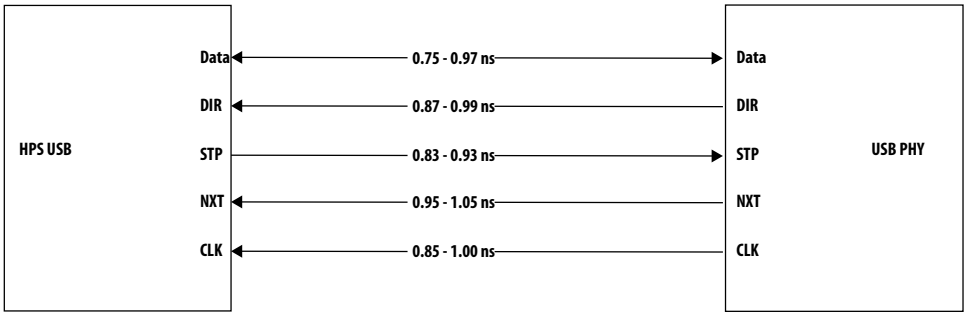
At 60 MHz, the period is 16.67 ns and in that time, for example, the clock must travel from the external PHY to the MAC and then the data and control signals must travel from the MAC to the PHY. Because there is a round-trip delay, the maximum length of the clock and ULPI signals are important. Based 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.

Figure 9. Trace Length



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.

Figure 10. System Diagram and Board Spec



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 Intel recommends 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 using external diodes and current limiters such as those used on the Intel FPGA development kits for Stratix 10 SoC.

For more information about the "Stratix 10 SoC Development Board Schematics", refer to the *Stratix 10 FPGA Development Kit User Guide*.

Related Information

Stratix 10 SoC Development Board Schematics

2.3.3. SD/MMC and eMMC Card Interface Design Guidelines

The Secure Digital/Multimedia Card (SD/MMC) controller, based on the Synopsys DesignWare attached to the hard processor system (HPS) is used for mass storage. This module supports:

- SD version 3.01, in addition to 3.0
- Embedded MMC (eMMC) version 4.51 and 5.0, in addition to 4.5(3)

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

HPS I/O use a fixed voltage level of 1.8 V. Many SD cards have an option to signal at 1.8 V or 3.3 V, although the initial power-up voltage requirement is 3.3 V. In cases when you want to use a 3.3 V SD card, voltage switching is required. To have the correct voltage level to power the card, voltage translation transceivers are required.

Follow the guidelines in the Voltage Switching section of the "SD/MMC Controller" chapter in the *Stratix 10 Hard Processor System Technical Reference Manual*

Table 3. Level Shifter Requirements

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

Related Information

SD/MMC Controller
Follow the guidelines detailed in the "Voltage Switching" section.

2.3.4. Design Guidelines for Flash Interfaces

GUIDELINE: Connecting the QSPI flash to the SoC device.

The HPS does not have a QSPI flash controller. The HPS has access to the QSPI controller in the SDM.

The Intel Stratix 10 SoC Development Kit uses the MT25QU02GCBB8E12-0SIT QSPI flash memory. This device is known to work with the SDM QSPI controller.

For more information about considerations when connecting QSPI flash to the SDM QSPI interface, refer to the *Intel Stratix 10 Hard Processor System Technical Reference Manual*.

Related Information

Intel Stratix 10 Hard Processor System Technical Reference Manual

(3) The HS400 mode is not supported.

2.3.4.1. 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 is 8- or 16-bit ONFI 1.0 compliant
- Supports x16 for mass storage 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)

For information about supported NAND devices, see the following table.

Table 4. Supported NAND Device

Part Number	Manufacturer	Capacity	Voltage	Support Category
MT29F8G16ABBCAH4-IT	Micron	8 GB	1.8 V	Known to work

2.3.5. UART Interface Design Guidelines

HPS boot firmware outputs console status messages throughout the boot process to the HPS UART port. If you want to view boot firmware console output, consider the following guidelines to assign the HPS UART peripheral to device I/O that are available at HPS boot time.

GUIDELINE: For the HPS First boot and configuration scheme, assign the HPS UART peripheral to the HPS Dedicated I/O Bank.

The SDM configures and releases to user-mode (Early I/O Release flow) the HPS Dedicated I/O and HPS SDRAM I/O before booting the HPS. The remaining FPGA I/O and fabric are not available until the rest of the FPGA is configured at a later point in the boot flow.

GUIDELINE: For the FPGA First boot and configuration scheme, you can assign the HPS UART to either HPS Dedicated or FPGA I/O.

The SDM configures the entire FPGA portion, including the entire I/O ring before booting the HPS.

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 5. 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

For more information, refer to the "UART Controller" section in the *Intel Stratix 10 Hard Processor System Technical Reference Manual*.

Related Information

- Intel Stratix 10 Hard Processor System Technical Reference Manual
- Intel Stratix 10 SoC Boot User Guide

2.3.6. I²C Interface Design Guidelines

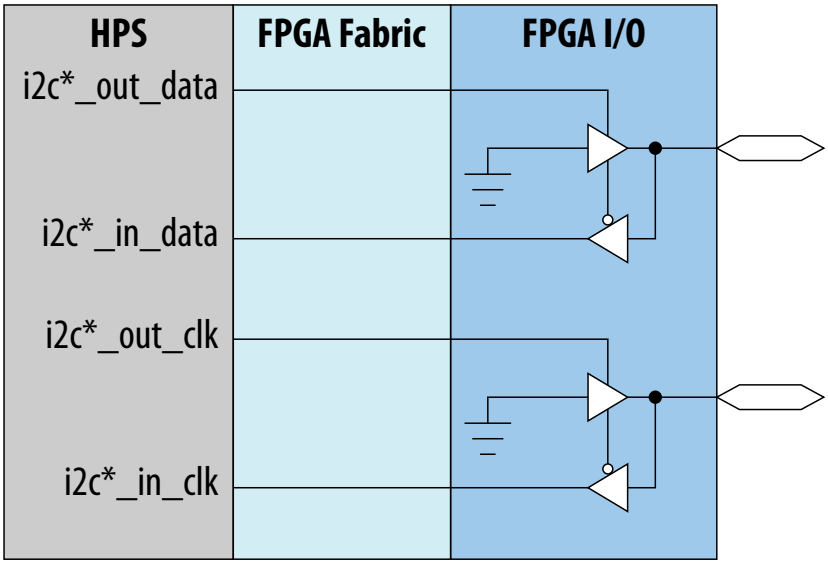
GUIDELINE: Instantiate the open-drain buffer when routing I²C signals through the FPGA fabric.

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 `altiobuff` 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 I²C 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 11. I²C Wiring to FPGA pins



GUIDELINE: Ensure that the high and low clock counts are configured correctly for the speed of the I²C interface

There is an I²C internal clock located in the:

- SDM—125 MHz
- HPS—100 MHz

The default settings for the high and low clock counts are configured for 125 MHz, so the default high and low clocks for the HPS I²C are longer than expected.

2.4. 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 EMIF planning tools and essential documentation:

EMIF Planning Tools

Tools	Description
External Memory Interfaces IP - Support Center	The External Memory Interfaces IP - Support Center is a collection of tools and documentation resources to aid in the design of external memory interfaces for FPGAs.
EMIF Device Selector	This is an easy to use tool for quickly identifying a list of Intel Stratix 10 SoC device packages based upon the type and performance characteristics of each external memory interface in your application. The tool reports on remaining general purpose I/O as well as transceiver counts for each device package in the list.
EMIF Spec Estimator	This is an easy to use tool to determine the required SoC device speed grade once you have identified device packages that can implement the number, type and performance characteristics of your application memory interfaces.

For more information about EMIF IP generation and Intel Quartus Prime compilation and timing closure aids, refer to the External Memory Interfaces IP - Support Center website.

Essential Documentation

Documentation	Description
Intel Stratix 10 General Purpose I/O User Guide	The *Intel Stratix 10 General Purpose I/O User Guide* describes the I/O column architecture and where the specific Hard Memory Controller block accessible to the HPS resides, For guidance on connecting the HPS-accessible hard memory controller block to the HPS, study **Section 1.3: Modular I/O Banks Location and Pin Counts in Stratix 10 Devices** of the *General Purpose I/O User Guide*. This section shows the I/O column and bank locations for all device and package combinations across all Intel Stratix 10 family variants, including the relative location of the HPS to its accessible banks.
External Memory Interfaces Intel Stratix 10 FPGA IP User Guide	The Intel Stratix 10 External Memory Interfaces User Guide includes the details required to understand what specific I/O banks are used for HPS external memory interfaces and where address/command, ECC and data signals are located. The user guide 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 this user guide, understanding the following sections is a prerequisite to properly design the Intel Stratix 10 EMIF for the HPS IP in your application. • **Section 5.3.3.1. General Guidelines**—This section shows the number of supported memory types and widths supported by Stratix 10 SX device/package combinations. • **Chapter 2: Intel Stratix 10 EMIF IP Product Architecture**—This section describes in greater detail the I/O Column, HMC, I/O lanes, and the hardened feature support for DDR SDRAM memories in the I/O elements. • **Section 2.7.1: Restrictions on I/O Bank Usage for Intel Stratix 10 EMIF IP with HPS**—This section provides a diagram that shows the specific I/O bank and lane locations for address/command, ECC, and data signals.

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

Related Information

- Intel Stratix 10 General Purpose I/O User Guide
- External Memory Interfaces IP - Support Center
- External Memory Interfaces Intel Stratix 10 FPGA IP User Guide

2.4.1. Considerations for Connecting HPS to SDRAM

The hard memory controller for the Intel Stratix 10 HPS is in the FPGA I/O columns along with the other hardware memory controllers. The HPS can use only one hard memory controller, and it is located closest to the HPS block in I/O bank 2M, where the address/command and ECC signals reside. Use I/O Bank 2N for 16-bit and 32-bit interface DQ/DQS data group signals. I/O Bank 2L is used for 64-bit interface DQ/DQS data group signals.

Instantiating the Intel Stratix 10 HPS EMIF IP

Connecting external SDRAM to the Intel Stratix 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 Intel Stratix 10 External Memory Interfaces for HPS IP in Platform Designer.

You must use a specific EMIF IP in Platform Designer to connect the HPS to external SDRAM memory.

The EMIF module is found in the IP catalog pane by selecting: **Library ➤ Processors and Peripherals ➤ Hard Processor Components ➤ External Memory Interfaces for HPS Intel Stratix 10**.

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

To connect the HPS to the EMIF in Platform Designer, you must connect the `hps_emif` conduit in the instantiated **emif_s10_hps_0** module to the `hps_emif` conduit in the **stratix10_hps_0 module**.

GUIDELINE: You must provide a free running and stable reference clock source to external memory interface before the start of device configuration.

For more information, refer to the *Intel Stratix 10 External Memory Interfaces IP User Guide*.

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 can save and recover 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 (for example: `h2f_reset`, `h2f_cold_reset`), other resets in the system that also source an HPS cold reset input (for example: `nCONFIG` and `HPS_COLD_nRESET` reset input pin.

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 Platform Designer, 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.

Note: The checkbox to enable data masking is found in the "Parameters" tab for the External Memory Interfaces for HPS Intel Stratix 10 Intel FPGA IP within the Topology section of the memory sub-tab.

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

Related Information

- External Memory Interface web page
- Intel Stratix 10 External Memory Interfaces IP User Guide

2.4.2. HPS SDRAM I/O Locations

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

GUIDELINE: Intel recommends that you use these automated default pin location assignments as a starting point.

You may need to modify the default pinout to meet the restrictions shown in this section.

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

By default, Intel 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: Make sure all I/O associated with the HPS memory interface are located within the active HPS EMIF I/O banks.

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:

Table 6. HPS SDRAM I/O Locations

EMIF Width	Bank 2N Lanes				Bank 2M Lanes				Bank 2L Lanes			
	3	2	1	0	3	2	1	0	3	2	1	0
16-bit	GPIO		16-bit Data		NC(4)	Address/ Command/RZQ/RefClk			GPIO			
16-bit + ECC	GPIO		16-bit Data + ECC			Address/ Command/RZQ/RefClk			GPIO			
32-bit	32-bit Data				NC	Address/ Command/RZQ/RefClk			GPIO			
32-bit + ECC	32-bit Data + ECC					Address/ Command/RZQ/RefClk			GPIO			
64-bit	64-bit Data				NC	Address/ Command/RZQ/RefClk			64-bit Data			
64-bit + ECC	64-bit Data + ECC					Address/ Command/RZQ/RefClk			64-bit Data + ECC			

Pin Assignments

Within a single data lane (which implements a single x8 DQS group):

- DQ pins must use pins at indices: 1, 2, 3, 6, 7, 8, 9, 10. You may swap the locations between the DQ bits (that is, you may swap location of DQ[0] and DQ[3]) so long as the resulting pin-out uses pins at these indices only.
- DM/DBI pin must use pin at index 11. There is no flexibility.
- DQS must use pin at index 4, and DQS# must use pin at index 5. There is no flexibility.
- Place ALERT# pin in I/O bank 2N, Lane 0, pin index 0 or I/O bank 2N, Lane 1, pin index 0. Otherwise, pin index 0 must have "no connect"; unless it is used for HPS REFCLK_P, Address/Command/RZQ/RefClk, or general-purpose I/O, where allowed.
- Assignment of data lanes must be as illustrated in the above table. You can swap the locations of entire byte lanes (that is, you may swap locations of byte 0 and byte 1) so long as the resulting pin-out uses only the lanes permitted by your HPS EMIF configuration, as shown in the above table.
- I/O bank 2M Lane 0, 1, and 2 must only be used for Address/Command/RZQ/ RefClk, otherwise “no connect”.
- You must not change placement of the address and command pins from the default.
- If not using ECC, IO bank 2M Lane 3 must be “no connect”. If using ECC, the ECC DQS group can be in any single data lane that is not otherwise restricted, that is, there is no requirement for the ECC DQS group to be placed in 2M.
- HPS REFCLK_P must use IO bank 2M Lane 2 pin index 0. HPS REFCLK_N must use IO bank 2M Lane 2 pin index 1.
- RZQ must use IO bank 2M Lane 2 pin index 2.

(4) NC indicates "no connect".

DQ/DQS Group Placement

Configuration	DQS Group Placement
16 bit	Must be placed in I/O lanes 0 and 1 of 2N.
16 bit + ECC	Must be placed in I/O lanes 0 and 1 of 2N and I/O lane 3 of 2M.
32 bit	Must be placed in 2N.
32 bit + ECC	Must be placed in 2N and I/O lane 3 of 2M.
64 bit	Must be placed in 2N and 2L.
64 bit + ECC	Must be placed in 2N, 2L, and I/O lane 3 of 2M.

Note: In all cases, the DQ/DQS groups can be swapped around in the I/O banks shown. There is no requirement for the ECC DQS group to be placed in 2M.

Unused HPS EMIF I/O Availability as FPGA GPIO

- Bank 2N (Data[31:0])—Unused lanes for 16-bit interfaces are available for FPGA GPIO.
- Band 2M (Addr/Cmd/ECC)
  - Lanes 0, 1, 2 are NOT AVAILABLE as FPGA GPIO
  - Lane 3 is NOT AVAILABLE as FPGA GPIO when not using ECC
- Bank 2L (Data[63:32])—Unused lanes for 32-bit or less interfaces are available as FPGA GPIO

2.5. 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, 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 Intel Stratix 10 External Memory Interfaces for HPS IP as described in the sub-section on Instantiating the Intel Stratix 10 EMIF IP described in the "Compiling Intel Stratix 10 EMIF IP with the Intel Quartus Prime Software" section of the *External Memory Interface Handbook Volume 3: Reference Material*.

For more information, refer to the following documentation:

- *External Memory Interface Handbook Volume 3: Reference Material*
- *External Memory Interfaces Intel Stratix 10 FPGA IP User Guide*
- *Intel Stratix 10 Device Pin-Out Files*

Related Information

- External Memory Interfaces Intel Stratix 10 FPGA IP User Guide
- Stratix 10 Device Pin-Out Files
- External Memory Interface Handbook Volume 3: Reference Material

2.6. Boundary Scan for HPS

The HPS JTAG interface does not support boundary scan tests (BST). To perform boundary scan testing on HPS I/Os, you must first chain the FPGA JTAG and HPS JTAG internally, and issue the boundary scan from the FPGA JTAG.

GUIDELINE: Chain the FPGA and HPS JTAG interfaces internally to perform boundary scan testing.

To chain the FPGA and HPS JTAG internally, go to Quartus **Device and Pins Options** and select the **Configuration** category. Under the **HPS debug access port (DAP)** settings, choose **SDM Pins** from the drop down option. If boundary scan is not being used, the FPGA JTAG and HPS JTAG interfaces can be used independently. To select HPS Dedicated I/O as the interface for HPS JTAG, select **HPS Pins** from the drop down option instead.

2.7. Embedded Software Debugging and Trace

This device has just one JTAG port with FPGA and HPS JTAGs that can be chained together or used independently.

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

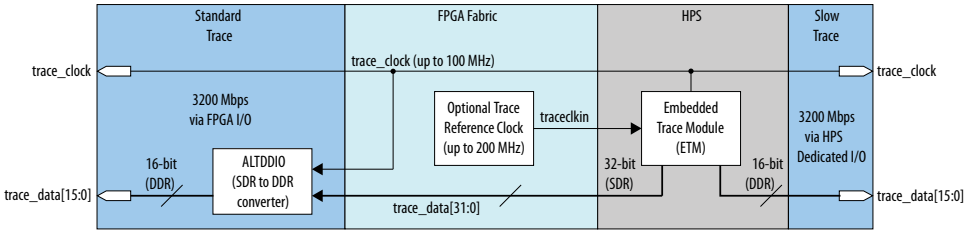
The HPS offers two trace interfaces either through HPS Dedicated I/O or FPGA I/O. The interface through HPS Dedicated I/O is a 16-bit DDR interface that you can use to trace low bandwidth traffic (such as the MPU operating at a low frequency).

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.

Intel recommends 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 12. Trace Diagram



2.8. Board Design Guidelines for Stratix 10 SoC FPGAs Revision History

Table 7. Board Design Guidelines for Stratix 10 SoC FPGAs Revision History

Document Version	Changes
2018.12.24	• Updated the "HPS SDRAM I/O Locations" section with restrictions on where the PLL reference clock and RZQ pin must be placed. • Added mandatory Intel Stratix 10 HPS EMIF pin placement rules. • Removed the following sections: — I/O Bank 2M, Lanes 0,1,2 (Addr/Cmd) — I/O Bank 2M, Lane 3 (ECC) — I/O Bank, 2N (Data) — I/O Bank, 2L (Data, 64/72-bit interfaces)
2018.09.24	• Changed the pin name from HPS_COLD_RESET to HPS_COLD_nRESET. • Updated the steps for configuring the HPS_COLD_nRESET to be on any open SDM I/O pin. • Added the GUIDELINE: Do not connect HPS_COLD_nRESET to other resets on the board in the "Pin Features and Connections for HPS Clocks, Reset and PoR" section.
2018.05.07	• Added a guideline for implementing an SGMII PHY interface using FPGA transceiver I/O for an S10 HPS EMAC instance, since the TSE MAC IP with 1000BASE-X PCS option no longer provides an option for transceiver I/O. • Added a guideline to reflect that the HPS EMIF reference clocks are stable pior to FPGA configuration. • Removed instances of LPDDR3
2018.03.01	• Replaced "SDM JTAG" with "FPGA JTAG through SDM dedicated pins" in the Summary of SoC-FPGA I/O Types table in the "Design Considerations for Connecting Device I/O to HPS Peripherals and Memory" section. • Corrected definition for HPS_COLD_nRESET, because it is not configured for "open drain". • Updated the "Boundary Scan for HPS" section with details about how to issue a boundary scan from the FPGA JTAG; and how to chain the JTAG.
2017.11.06	Initial release

3. Interfacing to the FPGA for Stratix 10 SoC FPGAs

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.

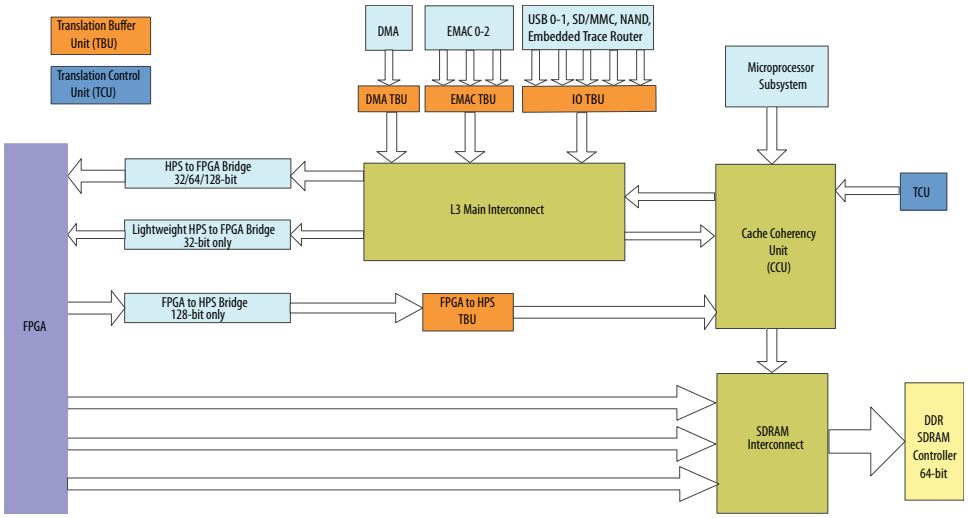
Design guidelines for the remainder of the FPGA portion of your design are provided in the *Stratix 10 Device Design Guidelines*.

3.1. Overview of HPS Memory-Mapped Interfaces

The HPS exposes three memory-mapped interfaces between the HPS and FPGA.

- HPS-to-FPGA bridge: 32-, 64-, or 128-bit wide Advanced Microcontroller Bus Architecture (AMBA*) Advanced eXtensible Interface (AXI*)-4
- Lightweight HPS-to-FPGA bridge: 32-bit wide AXI-4
- FPGA-to-HPS bridge: 128-bit wide ACE*-Lite
- FPGA-to-SDRAM AXI-4 port: three interfaces, 32, 64, or 128 bits wide, allowing the FPGA to directly access the HPS-connected SDRAM

Figure 13. Stratix 10 HPS Connectivity



3.1.1. HPS-to-FPGA Bridge

GUIDELINE: Use the HPS-to-FPGA 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 HPS-to-FPGA bridge is used for HPS boot, ensure that the FPGA portion of the SoC device is configured first.

The HPS-to-FPGA bridge is accessed if the MPU boots from the FPGA. Before the MPU boots from the FPGA, the FPGA portion of the SoC device must be configured, and the HPS-to-FPGA bridge must be remapped into addressable space. Otherwise, access to the HPS-to-FPGA bridge during the boot process results in a bus error. To satisfy these requirements, use the FPGA First boot and configuration scheme. The standard tool flow for boot firmware generation takes care of mapping the HPS-to-FPGA bridge into addressable memory space.

For more information about the FPGA First boot and configuration scheme and generating boot firmware for the Stratix 10 HPS, refer to the *Intel Stratix 10 SoC Boot User Guide*.

Related Information

Intel Stratix 10 SoC Boot User Guide

3.1.2. 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 Platform Designer can adapt the transactions to widths other than 32 bits within the interconnect generated in the FPGA portion.

3.1.3. FPGA-to-HPS Bridge

GUIDELINE: Use the FPGA-to-HPS bridge for cache coherent memory 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 a fixed 128-bit data path. Platform Designer can handle the data width adaptation in the generated interconnect for narrow masters.

GUIDELINE: The FPGA-to-HPS bridge supports cache coherent memory accesses with the ACE-Lite protocol.

FPGA masters must use the ACE-Lite cache signaling extensions for cache coherent accesses.

For more information about the ACE-Lite protocol extensions for cache coherent transactions, refer to the AMBA AXI and ACE Protocol Specification on the Arm* Developer website.

Related Information

AMBA AXI and ACE Protocol Specification

3.1.4. FPGA-to-SDRAM Ports

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

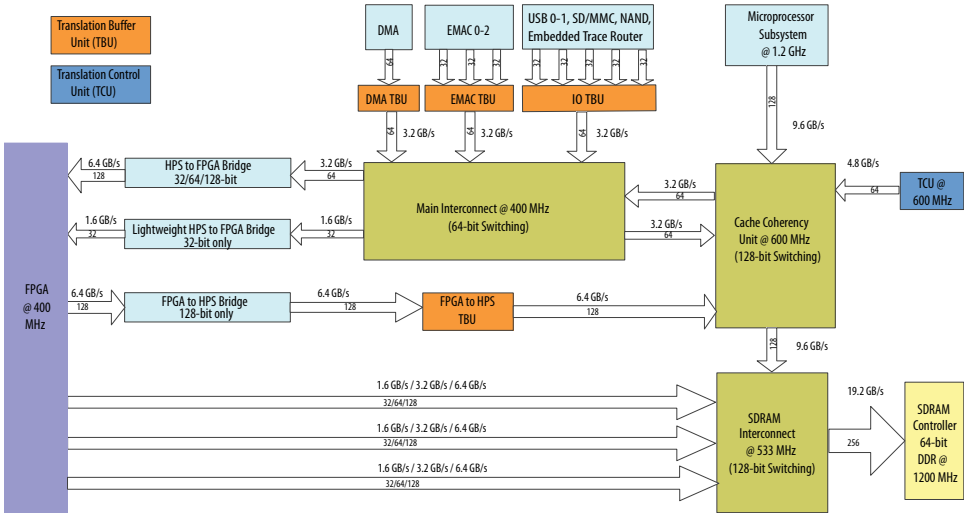
The FPGA-to-SDRAM ports allow masters implemented in the FPGA fabric to directly access HPS SDRAM without the transactions flowing through the CCU. There are three FPGA-to-SDRAM ports—FPGA-to-SDRAM0, FPGA-to-SDRAM1, FPGA-to-SDRAM2—supporting 32-, 64-, or 128-bit data paths. These interfaces connect only to the HPS SDRAM subsystem so Intel recommends 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 use the FPGA-to-HPS bridge with support for cache coherent accesses using the ACE-Lite protocol.

3.1.5. 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 400 MHz, the MPU operates at 1200 MHz, and the 64-bit external SDRAM operates at 1067 MHz.

Figure 14. Stratix 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

Interface	Transaction Use Case	Latency	Throughput
HPS-to-FPGA	MPU accessing memory in FPGA	Medium	Medium
HPS-to-FPGA	MPU accessing peripheral in FPGA	Medium	Very Low
Lightweight HPS-to-FPGA	MPU accessing register in FPGA	Low	Low
Lightweight HPS-to-FPGA	MPU accessing memory in FPGA	Low	Very Low
FPGA-to-HPS	FPGA master accessing non-cache coherent SDRAM	High	Medium
FPGA-to-HPS	FPGA master accessing HPS on-chip RAM	Low	High
FPGA-to-HPS	FPGA master accessing HPS peripheral	Low	Low
FPGA-to-HPS	FPGA master accessing coherent memory resulting in cache miss	High	Medium
FPGA-to-HPS	FPGA master accessing coherent memory resulting in cache hit	Low	Medium-High
FPGA-to-SDRAM	FPGA master accessing SDRAM through single FPGA-to-SDRAM port	Medium	High
FPGA-to-SDRAM	FPGA masters accessing SDRAM through multiple FPGA-to-SDRAM ports	Medium	Very High

Note: For the interfaces with no configuration recommended, refer to the corresponding interface sections: "HPS-to-FPGA Bridge", "Lightweight HPS-to-FPGA Bridge", and "FPGA-to-HPS Bridge".

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 masters in the FPGA.

The FPGA-to-HPS bridge is optimized for accessing non-SDRAM accesses (peripherals, on-chip RAM). 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 (for example, 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 Intel recommends 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 ACE-Lite cache extensions 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 through one of the FPGA-to-SDRAM bridge ports, achieving the highest throughput possible. Even though the HPS includes a DMA engine internally that can 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.

Related Information

- HPS-to-FPGA Bridge on page 36
- Lightweight HPS-to-FPGA Bridge on page 36
- FPGA-to-HPS Bridge on page 37

3.2. 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.

3.2.1. 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 lightweight HPS-to-FPGA bridge.

It is important to include both HPS-to-FPGA and lightweight HPS-to-FPGA bridge 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 bridge can be accessed in parallel if there are multiple HPS masters accessing the FPGA fabric at the same time so including both bridge can also improve the performance of the system.

3.2.2. 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 require 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.

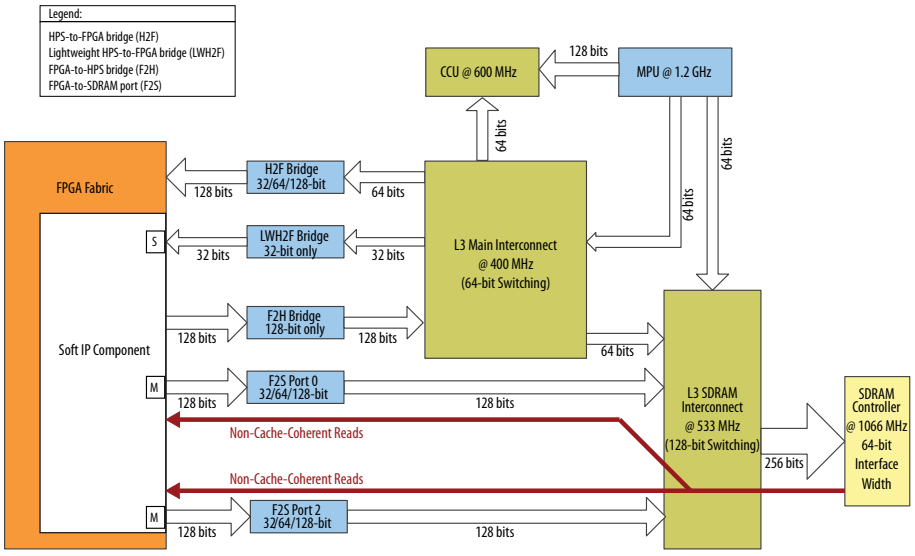
3.2.3. Examples of Cacheable and Non-Cacheable Data Accesses From the FPGA

3.2.3.1. 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 15. FPGA Reading Data from HPS FPGA-to-SDRAM Ports

This figure depicts an example of using two of the three F2S ports configured for 128 bits in width.



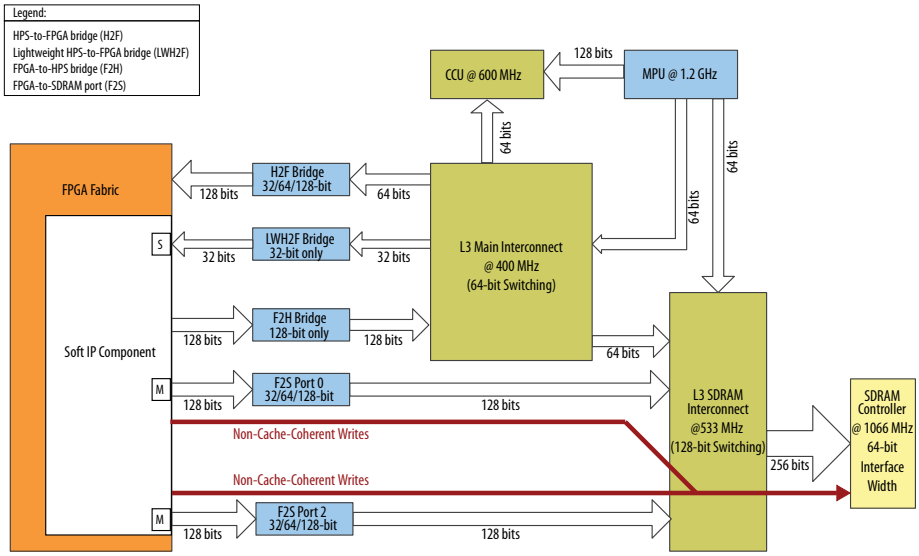
Since the Stratix 10 HPS supports up to three 128-bit ports into the SDRAM you can maximize the read throughput by implementing as many as three masters in the FPGA accessing data in the SDRAM through each port. If you decide to implement multiple paths into the SDRAM through the FPGA-to-SDRAM ports 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 others, then you can adjust the QoS settings for each port shaping the traffic patterns as needed by your application. recommends 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.(5)

3.2.3.2. 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, software may need to flush or invalidate cache lines before the transfer starts, 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 16. FPGA Writing Data to HPS FPGA-to-SDRAM Ports

This figure depicts an example of using two of the three F2S ports configured for 128 bits in width.



Note: Like in Example 1: FPGA Reading Data from HPS SDRAM Directly on page 41, where the FPGA reads data from the FPGA-to-SDRAM ports, you can maximize write throughput into the HPS SDRAM by using multiple 128-bit FPGA-to-SDRAM ports with at least one master in the FPGA connected to each port.

(5) Ensure that Avalon®-MM burst transactions into the HPS do not cross the 4 KB address boundary restriction specified by the AXI protocol.

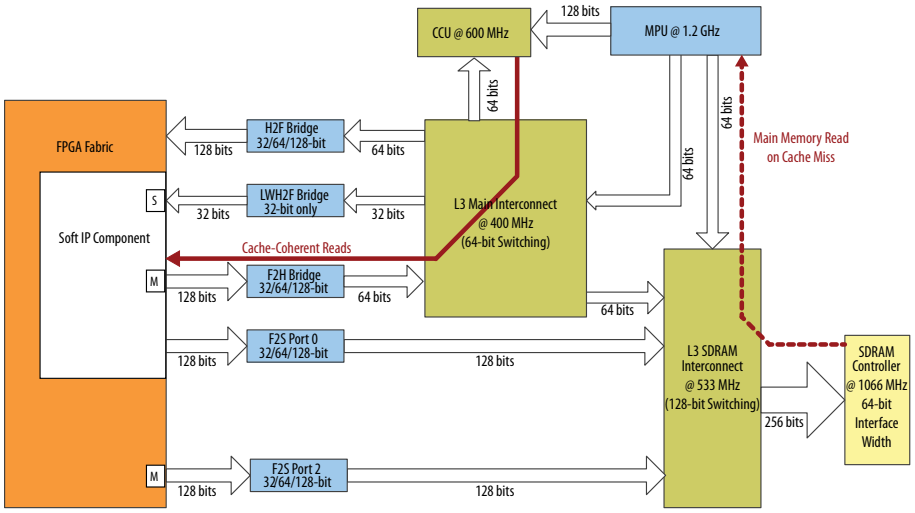
3.2.3.3. Example 3: FPGA Reading Cache Coherent Data from HPS

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. Otherwise, the L2 cache might thrash reading data from SDRAM for most 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 utilize the cache extension signaling of the ACE-Lite protocol. For more information about the ACE-Lite protocol signaling extensions for cache coherent accesses, refer to the "Related Information" section.

Figure 17. FPGA Reading Cache Coherent Data



GUIDELINE: Perform cacheable accesses aligned to 64 bytes targeting the FPGA-to-HPS bridge.

The CCU of the HPS is optimized for transactions that are the same size as the cache line (64 bytes). As a result you should attempt to align the data to 64 byte boundaries and ensure after data width adaptation the burst length into the 128-bit FPGA-to-HPS bridge port is four beats long. For example, a 64-bit FPGA master should align the data to be 64 byte aligned and perform full 64-bit accesses with a burst length of 8.

GUIDELINE: Access 64 bytes per cacheable transaction.

Ensure that each burst transaction accesses 64 bytes. Each transaction must start on a 64-byte boundary.

Table 8. Burst Lengths for 64-byte Alignment

FPGA Master Width (Bits)	Access Size (Bytes)	Burst Length
32	4	16
64	8	8
128	16	4

Related Information

AMBA AXI and ACE Protocol Specification

3.2.3.4. Example 4: FPGA Writing Cache Coherent Data to HPS

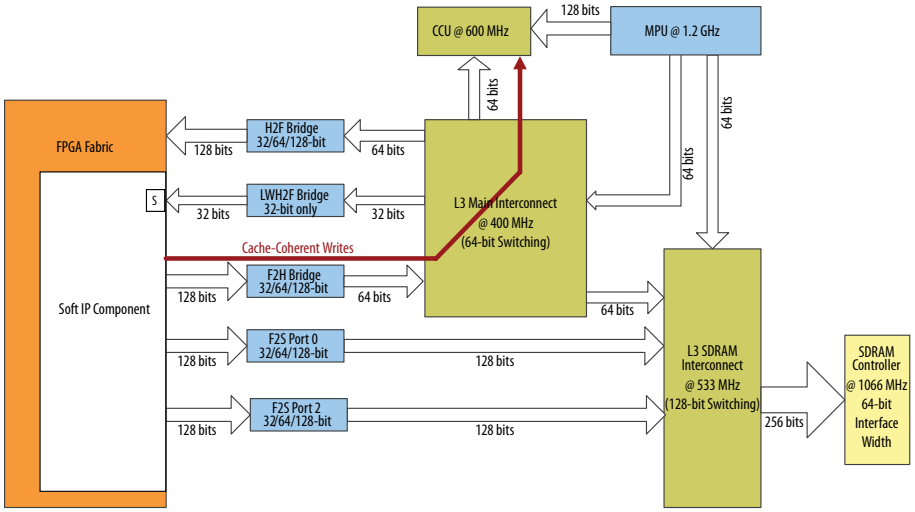
In this example the HPS MPU requires access to data that originates in the FPGA. The most efficient mechanism for sharing small blocks of data with the MPU is to have logic in the FPGA perform cacheable writes to the HPS. It is important that the amount of data to be written to the HPS be in the form of relatively small blocks because large block writes cause the L2 cache to thrash, causing the cache to write to SDRAM for most 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 utilize the cache extension signaling of the ACE-Lite protocol. See the Related Information for details on the ACE-Lite protocol signaling extensions for cache coherent accesses.

Figure 18. 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 CCU 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 CCU 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, 64 byte cache transactions result in the best performance. For more information about 64 byte cache transactions, refer to *GUIDELINE: Access 64 bytes per cacheable transaction.* in the "Example 3: FPGA Reading Cache Coherent Data from HPS" section.

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

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

Related Information

- AMBA AXI and ACE Protocol Specification
- Example 3: FPGA Reading Cache Coherent Data from HPS on page 43

3.3. 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.

For more information, refer to the "Golden Hardware Reference Design (GHRD)" section.

Related Information

Golden Hardware Reference Design (GHRD) on page 55

3.4. Timing Closure for FPGA Accelerators

The HPS bridge and FPGA-to-SDRAM ports exposed to the FPGA are synchronous; and clock crossing is performed within the interface itself. As a result, you must only ensure that both the FPGA-facing logic and your user design close timing in Timing Analyzer. Interrupts are considered asynchronous by the HPS, and as a result the HPS logic resynchronizes them to the internal HPS clock domain so there is no need to close timing for them.

3.5. Information on How to Configure and Use the Bridges

By default, the SSBL only brings all the bridges out of reset. It does not automatically configure or enable the bridges. You must specifically configure and enable all the bridges according to your own design. This can be accomplished by creating a "`u-boot.scr`" script file that is executed by the SSBL, where the SSBL modifies any registers necessary to configure the bridges. At this point the bridges are configured and enabled, and cannot be changed by the SSBL, even during any future FPGA configurations.

3.6. Interfacing to the FPGA for Stratix 10 SoC FPGAs Revision History

Table 9. Interfacing to the FPGA for Stratix 10 SoC FPGAs Revision History

Document Version	Changes
2018.09.24	Added information about how to configure and use the bridges during FPGA Configuration First and HPS Boot First modes.
2017.11.06	Initial release

4. System Considerations for Stratix 10 SoC FPGAs

4.1. Timing Considerations

The following PHY interfaces can be selected when configuring your system:

- HPS EMAC PHY Interfaces
  - Reduced Media Independent Interface (RMII)
  - Reduced Gigabit Media Independent Interface (RGMII)
- PHY Interfaces connected through FPGA I/O
  - GMII/MII
    - RGMII—Using the GMII-to-RGMII Adapter
    - RMII—Using the MII-to-RMII Adapter
    - Serial Gigabit Media Independent Interface (SGMII)—Using the GMII-to-SGMII Adapter
  - Management Data Input/Output (MDIO)

For more information about the timing considerations for each of these PHY interfaces, refer to their corresponding sections under the "Design Guidelines for HPS Interfaces" section.

Related Information

Design Guidelines for HPS Interfaces on page 9

4.1.1. Timing Closure for FPGA Accelerators

The HPS bridges and FPGA-to-SDRAM ports exposed to the FPGA are synchronous and clock crossing is performed within the interface itself. As a result, you must only ensure that both the FPGA-facing logic and your user design close timing in Timing Analyzer. Interrupts are considered asynchronous by the HPS, and as a result the HPS logic resynchronizes them to the internal HPS clock domain so there is no need to close timing for them.

Conduits carry signals that do not fit into any standard interface supported by Platform Designer. Examples of these conduits are HPS peripheral external interfaces routed into the FPGA fabric or the HPS DMA peripheral request interfaces.

4.1.1.1. HPS First and FPGA First Boot Considerations

The Stratix 10 SoC device supports two boot and configuration modes. When designing your system, you must choose one of the following boot modes for your application: HPS First and FPGA First.

Note: Faster HPS boot times are possible using HPS First boot mode.

Guideline: Engineering Sample Device Restrictions

Engineering Sample (ES) devices only support the HPS First boot mode with dual flash. Any boards built with ES devices must be designed to support HPS First boot mode. It is possible to design your board for both HPS First and FPGA First boot with either dual or single flash, if you plan to use a different boot scheme in production.

Guideline: HPS First Boot Mode Utilizes Early I/O Release

Follow the guidelines in this document to properly design your board and the SoC device pin out for the HPS SDRAM interface for Early I/O Release.

For more information about the supported boot modes, refer to the *Intel Stratix 10 SoC Boot User Guide* and the "Boot and Configuration" section in the *Intel Stratix 10 Hard Processor System Technical Reference Manual*.

Related Information

- Intel Stratix 10 Hard Processor System Technical Reference Manual
- Intel Stratix 10 SoC Boot User Guide

4.1.2. USB Interface Design Guidelines

The Intel Stratix 10 HPS can connect its embedded USB MACs directly to industry-standard USB 2.0 ULPI PHYs using the 1.8 V dedicated HPS I/O. 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.

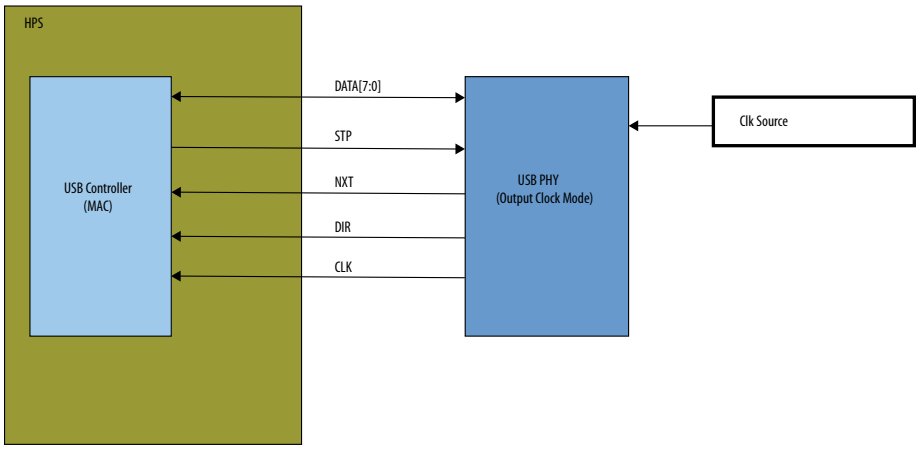
GUIDELINE: Intel recommends that you design the board to support both USB PHY modes where the device supplies the clock versus where an external clock is the source.

The interface between the ULPI MAC and PHY on the Stratix 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, Intel recommends 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.

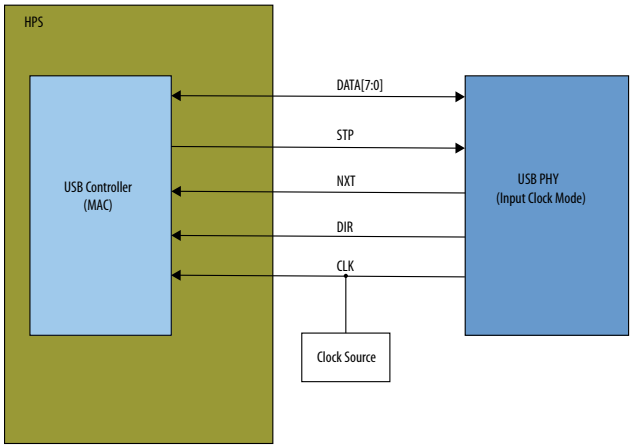
- Output Mode—In output clock mode, the clock is generated by the USB PHY. All signals are synchronized to this clock.

Figure 19. Output Mode



- Input Mode—In input clock mode, the PHY receives a clock from an external source. All signals are synchronized to the clock. In this mode, the clock can be generated by a PLL in the FPGA or by an external source.

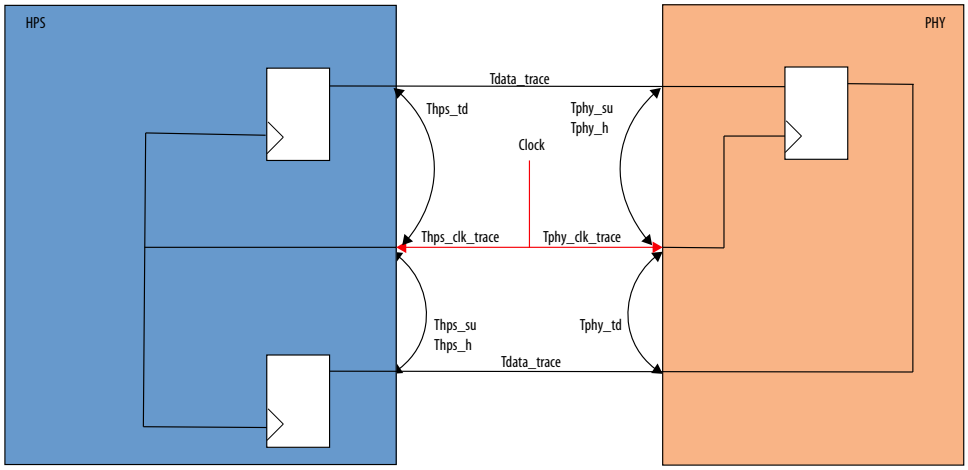
Figure 20. Input Mode



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

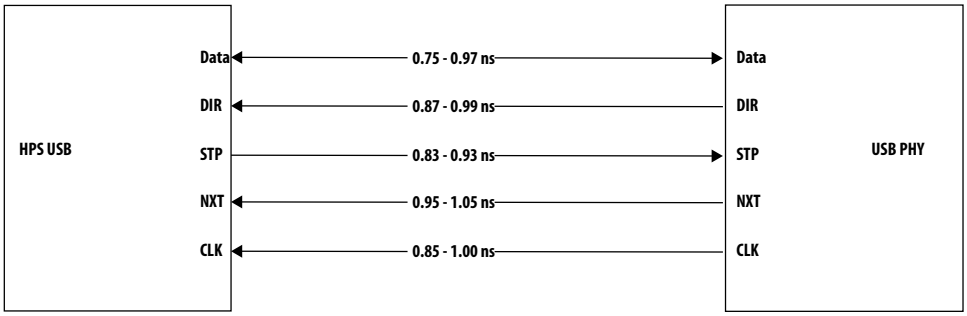
At 60 MHz, the period is 16.67 ns and, in that time, for example, the clock must travel from the external PHY to the MAC and then the data and control signals must travel from the MAC to the PHY. Because there is a round-trip delay, the maximum length of the CLK and ULPI signals are important. Based 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.

Figure 21. Trace Length



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.

Figure 22. System Diagram and Board Spec



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 Intel recommends 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 using external diodes and current limiters such as those used on the Intel development kits for Stratix 10 SoC.

4.2. Maximizing Performance

The memory-mapped connectivity between the HPS and the FPGA fabric is a crucial tool to maximize the performance of your design.

For more information about recommended topologies to optimize your system performance, refer to the guidelines in the "Interfacing to the FPGA" section.

Related Information

Interfacing to the FPGA for Stratix 10 SoC FPGAs on page 35

4.3. System Level 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 a directory-based CCU fabric using the ACE-Lite interface.

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 through the CCU. The mechanism you use depends on the size of the buffer of memory the master is accessing.

For more information, refer to the "Interfacing to the FPGA" section.

GUIDELINE: Ensure that data accessed through the CCU fits in the 1 MB L2 cache to avoid thrashing overhead.

Since the L2 cache is 1 MB in size, if a master in the system frequently accesses buffers whose total size exceeds 1 MB, 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 to

the CCU through the ACE-Lite protocol supported by the FPGA-to-HPS bridge ensure 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.

Related Information

Interfacing to the FPGA for Stratix 10 SoC FPGAs on page 35

4.4. System Considerations for Stratix 10 SoC FPGAs Revision History

Table 10. System Considerations for Stratix 10 SoC FPGAs Revision History

Document Version	Changes
2018.09.24	Maintenance release
2017.11.06	Initial release

5. Embedded Software Design Guidelines for Intel Stratix 10 SoC FPGAs

5.1. Overview

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

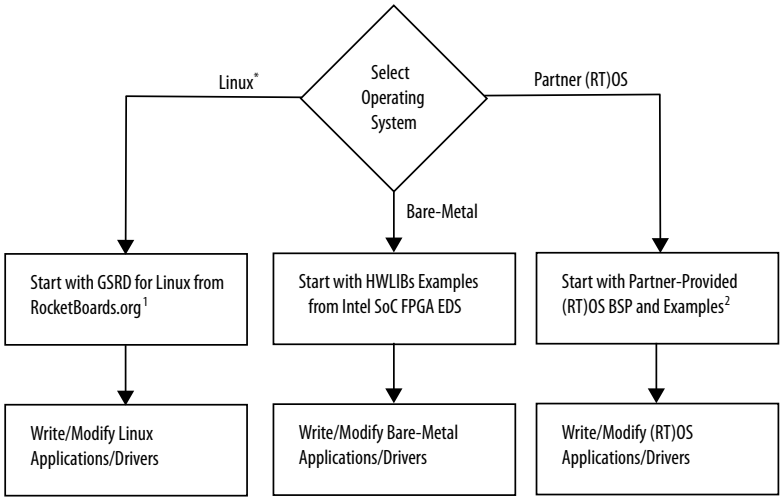
You must follow the provided recommendations to select the components of your software platform that suit the performance, support and time-to-market requirements of your end application.

5.2. Assembling the Components of Your Software Development Platform

To successfully build your software development platform, Intel recommends that you start with a baseline project, a known good configuration of an HPS system. Then you can modify the baseline project to suit your end application.

Figure 23 on page 55 presents the recommended procedure to determine the software development platform components.

Figure 23. Assembling Software Development Platform



(1) Golden System Reference Design (GSRD) provides a set of essential software and hardware system components that can be used as a starting point for various custom user designs. For more information, refer to RocketBoards.org.
Note: RocketBoards.org is an open source community for Linux application developers, firmware engineers, and hardware engineers.

(2) Some, but not all, partner-provided Board Support Package (BSP)s are based on the Golden Hardware Reference Design (GHRD).
Note: The GHRD is part of GSRD.

The flow consists of the following steps:

1. Select one of the following operating systems:
   - Bare-Metal
   - Linux* or partner operating system
   - Real-time operation system
2. Either or both write and update either or both applications and drivers

Related Information

RocketBoards.org Web Page
For more information about the GSRD, enter "GSRD" in the search field.

5.3. Golden Hardware Reference Design (GHRD)

The GHRD, part of the Golden System Reference Design (GSRD), is an Intel Quartus Prime project that contains a full HPS design for the Intel Stratix 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 Intel Stratix 10 SoC if you choose 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 software projects depend on a hardware design.

The GHRD is regression tested with every major release of the Intel Quartus Prime Pro Edition 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.

GUIDELINE: 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 either:

- GSRD for Linux web page—This location contains the latest version which is the best known configuration.
- SoC EDS Installation folder—`<SoC EDS Installation directory>\examples\hardware\s10_soc_devkit_ghrd`—This location contains the version supported by the corresponding SoC EDS version, used as a basis for the provided Intel hardware libraries (HWLIBs) design examples in the SoC EDS. This may not be the latest configuration.

Related Information

GSRD for Linux page

5.4. Selecting an Operating System for Your Application

5.4.1. Using Linux or RTOS

There are several operating systems that support the Intel Stratix 10 SoC, including Linux OS.

For more information, refer to the OS SoC Partner ecosystem web page.

There are many factors that go into the selection of an operating system for SoC FPGAs including:

- Features of the operating system
- Licensing terms
- Availability of collaborative software projects and frameworks based on the operating system
- Available device drivers and reference software
- In-house legacy code and familiarity with the operating system
- Real time requirements of your system
- Functional safety and other certifications required for your application

To select an appropriate operating system for your application, 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' websites are a good source of information you can use to help make your selection.

Intel supports the Yocto Project compatible, Ångström 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 devices.

There are several 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 several 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. You must take this into account when selecting an operating system.

5.4.2. Developing a Bare-Metal Application

The HPS can be used in a bare-metal configuration (without an operating system) 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. 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 Microprocessor Unit (MPU) subsystem.

For example:

- A typical bare-metal application uses only a single core. You must develop runtime capabilities to manage all four cores 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 makes more efficient use of the resources in your MPU subsystem.

5.4.3. 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
- UEFI—open-source BSD license
- Arm Trusted Firmware (ATF)—open-source BSD license

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

The Quad Core Arm Cortex-A53 MPCore* in the Intel Stratix 10 HPS can support both Symmetrical Multi Processing (SMP) and Asymmetrical Multi-processing (AMP) operating modes.

In SMP mode, a single operating system instance controls all four cores. The SMP configuration is supported by a wide variety of operating system 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 optimized performance and ease of use. For instance, SMP-enabled operating systems offer the option of setting processor affinity. This means that each task or thread can be assigned to run on a specific core. This feature allows you to better control the workload distribution for each Arm Cortex-A53 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 operating system or RTOS meets your performance and real-time requirements.

In the AMP configuration, up to four different operating systems could run on the four Cortex-A53 cores, which allows more valid combinations. You could also combine AMP and SMP allowing you to have two cores running an SMP and the other two cores running an AMP.

Special Considerations

- Use AMP only if you are familiar with the techniques to manage and schedule processes, handle inter-process communication, synchronize between events, and manage secure processes between the two instances of the operating systems.
- OS providers do not generally offer support for using their operating system in an AMP mode, so a special support agreement is typically needed in this case.
- If you use AMP, it is best to use the virtualization feature of the Cortex-A53 because the Cortex-A53 includes native hardware supports for virtualization solving most of the AMP resource sharing issues.

5.5. Assembling Your Software Development Platform for Linux

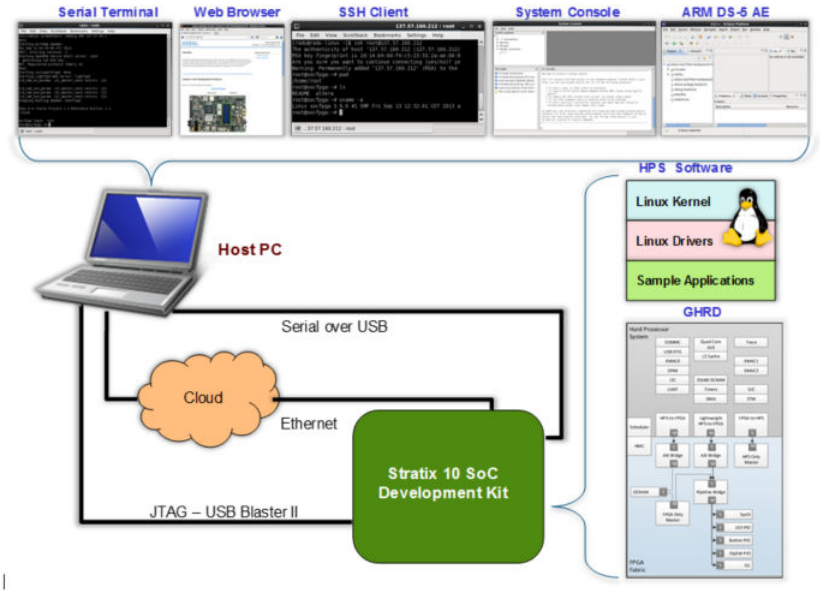
This section presents design guidelines to be used when you have selected Linux as the operating system for your end application.

5.5.1. Golden System Reference Design (GSRD) for Linux

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

- GHRD - A Intel Quartus Prime Pro Edition Prime project
- Reference U-Boot based bootloader
- Reference Linux BSP
- Sample Linux Applications

Figure 24. 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 Linux software development platform, Intel recommends that you use the latest 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.

Related Information

RocketBoards.org Web Page
For more information about the GSRD, enter "GSRD" in the search field.

5.5.2. Source Code Management Considerations

The GSRD build process relies on several git trees that are available online, including:

Table 11. Git Tree Link

Git Tree	Link
Intel SoC FPGA Linux Kernel	https://github.com/altera-opensource/linux-socfpga
Intel SoC FPGA Linux designs	https://github.com/altera-opensource/linux-refdesigns
Intel SoC FPGA Angstrom recipes	https://github.com/altera-opensource/angstrom-socfpga

Note: Intel provides Linux enablement, upstreams to mainline and collaborates with the Linux community. Intel provides two kernel versions, the latest stable kernel (N) and latest LTSI kernel (M) and drops support for previous Linux kernel versions (N-1, M-1). At any point in time the (N, N-1, M, M-1) versions are available from the kernel repository. Older kernel versions are removed.

GUIDELINE: Manage your own Git repositories and do not assume the contents of the repositories available on the intel-opensource site remains available. Managing Git repositories can be achieved in many ways, such as using a Git service provider. Some benefits of managing your own Git repositories include build reproducibility, source code management and leveraging the distributed model enabled by Git.

The GSRD uses the Angstrom `rootfilesystem`, built using Yocto recipes. The recipes pull in various open source package sources, and build them into the `rootfilesystem`. Because some of these recipes are generic, and do not refer to a specific version, the end result may be different from one build to another.

GUIDELINE: If you rebuild the Angstrom `rootfilesystem` and require repeatability, you must keep a copy of the Yocto downloads folder that was used for the build.

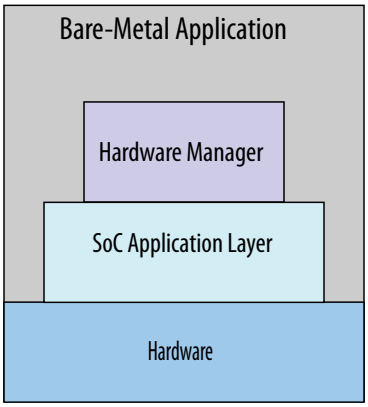
5.6. Assembling your Software Development Platform for a Bare-Metal Application

The 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 25. HWLIBs Overview



Note: Not all hardware is covered by SoCAL and HWMgr, therefore writing custom code may be necessary depending on application.

Software applications that use HWLibs must have runtime provisions to manage the resources of the MPU subsystem. These provisions are typically what operating systems provide.

GUIDELINE: Use HWLIBs only if you are familiar with developing runtime 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.

For more information about HWLIBs, refer to:

- *Intel SoC FPGA Embedded Design Suite User Guide*
- Getting Started with HWLIBs Bare-metal Development web page on the Wiki

Related Information

- Intel SoC FPGA Embedded Design Suite
- SoCEDSGettingStarted
  This wiki page contains a list of getting started guides to help you get started with the Intel SoC FPGA Embedded Development Suite (SoC EDS).

5.7. Assembling your Software Development Platform for Partner OS or RTOS

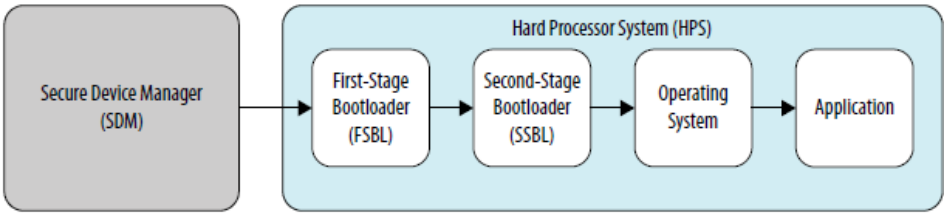
Partner OS providers offer board support packages and commercial support for the Intel SoC FPGA devices. Typically the support includes example getting started projects and associated documentation.

For more information about how to assemble the software development platform when targeting a partner OS or RTOS, refer to the partner documentation and support services.

5.8. Choosing the Bootloader Software

The typical Stratix 10 SoC HPS boot flow is depicted in the figure below:

Figure 26. Typical Intel Stratix 10 SoC Boot Flow



The boot loader software is one of the most important components of your software development platform. The bootloader initializes the system and then loads and passes control to the next boot image which is either an operating system or a bare-metal application.

The Intel Stratix 10 SoC boot loader software is split into two different stages:

- First Stage Bootloader (FSBL) – Loaded by the SDM from the FPGA configuration bitstream into the HPS side on-chip memory:
  - Provides essential initial hardware settings to configure the HPS
  - Software features to control the flash and peripheral components of the HPS
  - Utilities to enable early debugging and troubleshooting
- Second Stage Bootloader (SSBL) – Loaded by FSBL into the DDRAM and potentially having significantly more capabilities than FSBL, such as: network access, command line interface and scripting support.

Intel provides several bootloader options:

- **U-Boot Bootloader:** Inherits several features available from the open source community and is popular with Linux OS users. U-Boot bootloader is governed by GPL licensing. It is available as a part of SoC EDS and used by GSRD for Linux.
- **UEFI Bootloader:** Feature rich and popular with RTOS users and is governed by an open-source BSD style license.
- **ATF (ARM Trusted Firmware) Bootloader**: Used by the UEFI and provides just the first stage bootloader. It uses a BSD-style license, and could be used to directly load a bare-metal application instead of a SSBL.

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

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 DDRAM memory, flash boot source and peripheral interfaces.

GUIDELINE: Given the amount of 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.

5.9. 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 operating system.

Note: Familiarize yourself with the available tools for development, compilation and debug. A list of supported tools is available at the Intel FPGAs Ecosystem web page.

Related Information

Intel FPGAs Ecosystem web page

5.9.1. Selecting Software Build Tools

You must decide which software development tools to use, along with its version:

- Compiler
- Assembler
- Linker
- Archiver

The Arm Development Studio 5* Intel SoC FPGA Edition includes the Arm Bare-metal Compiler 6, an Arm V8-A software build tool.

The U-Boot is compiled with Linaro* GCC compiler from the Linaro Releases web page.

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

Related Information

Linaro Releases

5.9.2. Selecting Software Debug Tools

You must decide which software debug tools to use.

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 Intel SoC FPGA device. You can achieve a JTAG connection through:

- An embedded Intel FPGA Download Cable II like what is available in the Intel Stratix 10 SoC Development Kit.
- External JTAG hardware similar to what may be required when using the Lauterbach T32 tools.

5.9.3. 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 (for example, SDRAM) or the embedded trace buffer, then stopping the system, downloading the trace information through JTAG, and 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 Arm DS-5 Intel SoC FPGA Edition 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 Arm DSTREAM is required.

Lauterbach T32 also requires external hardware for real-time tracing.

5.10. Boot And Configuration Considerations

The Intel Stratix 10 SoC HPS does not have a boot ROM. Instead the SDM has a BootROM which loads the initial FPGA configuration bitstream. This bitstream also contains the HPS First Stage Bootloader (FSBL) binary.

For more information, refer to the *Intel Stratix 10 SoC FPGA Boot User Guide*.

Related Information

Intel Stratix 10 SoC FPGA Boot User Guide

5.10.1. Configuration Sources

The initial FPGA configuration and the HPS FSBL are part of the initial configuration bitstream, which can be obtained from several sources:

- Avalon-ST Data Source: An external Avalon-ST master provides the bitstream.
- JTAG Interface: An external JTAG master (usually driven by a host tool) provides the bitstream.
- SDM Flash: A flash device connected on SDM side provides the bitstream.

The following flash device types can be connected to SDM:

Table 12. Flash Type Support Status

Flash Type	Support Status
QSPI	Currently supported in the Intel Quartus Prime Pro Edition 18.1 release
SD/eMMC	Will be supported in future Intel Quartus Prime Pro Edition releases

5.10.2. Configuration Flash

The following QSPI devices are validated for Intel Stratix 10 SoC configuration:

Table 13. QSPI Devices

Vendor	Part Number	Capacity
Micron*	MT25QU128	128 Mb
Micron	MT25QU256	256 Mb
Micron	MT25QU512	512 Mb
Micron	MT25QU01G	1 Gb
Micron	MT25QU02G	2 Gb
Macronix*	MX25U128	128 Mb
Macronix	MX25U256	256 Mb
Macronix	MX25U512	512 Mb
Macronix	MX66U512	512 Mb
Macronix	MX66U1G	1 Gb
Macronix	MX66U2G	2 Gb

GUIDELINE: When configuring FPGA from flash, select a compatible QSPI device.

GUIDELINE: Select the QSPI device that fits your design. Using a larger device allows for increases in the design bitstream size.

5.10.3. Configuration Clock

GUIDELINE: In the Intel Quartus Prime Pro Edition GUI, select the configuration clock speed to match the capabilities of the QSPI flash device that you selected.

5.10.4. Selecting HPS Boot Options

You must select a configuration and boot mode from the "HPS Boot Source" sub-window located on the "FPGA Interfaces" tab in Intel Quartus Prime Pro Edition.

- **FPGA Configuration First**: The SDM configures the FPGA core and all the periphery I/O before loading the FSBL into the HPS on-chip RAM and releasing the HPS from reset. If any errors exist during initial configuration, the HPS is not released from reset.
- **HPS First**: The SDM only configures the I/O required for the HPS SDRAM, and then loads the FSBL into the HPS on-chip RAM before releasing the HPS from reset. The FPGA core, as well as the other unused I/O, remain unconfigured. The HPS configures the rest of the FPGA.

5.10.5. HPS Boot Sources

The HPS FSBL is included with the initial FPGA configuration bitstream; and the HPS SSBL can be in several places:

Table 14. HPS SSBL Support Status

HPS SSBL Location	Support Status
SDM QSPI	Currently supported in the Intel Quartus Prime Pro Edition 18.1 release
HPS SD/eMMC	Currently supported in the Intel Quartus Prime Pro Edition 18.1 release
HPS NAND	Will be supported in future Intel Quartus Prime Pro Edition releases

GUIDELINE: Intel recommends to place the HPS SSBL on the HPS SD/eMMC flash.

5.11. System Reset Considerations

After any one of the four Watchdog timers expire and generates a system reset request to the SDM, the SDM then performs one of three types of system resets:

- HPS Cold reset
- HPS Warm reset
- HPS Cold and trigger remote update

Note: One of these three options can be chosen from within the Intel Quartus Prime Pro Edition tool.

In the Intel Quartus Prime Pro Edition tool, you must select the "HPS Clocks and resets" tab, then the "Resets" tab, then click on the "Enable watchdog reset" check box, and then choose one of three choices from the pull-down menu for the "How SDM handles HPS watchdog reset" label:

- **HPS Cold reset**
  - **Impact on HPS**—The SDM holds the processor in reset. The SDM loads the FSBL from the same bitstream that was loaded into the device prior to the cold reset into the HPS on-chip memory. When successfully completed, the SDM releases the HPS reset causing the processor to start executing code from the reset exception address.
  - **Impact on FPGA**—The FPGA core fabric is untouched during the reset. After exiting reset, software determines whether to reconfigure the FPGA portion.
- **HPS Warm reset**
  - **Impact on HPS**—The SDM holds the processor in reset. The FSBL remains in the on-chip RAM during a warm reset. The SDM takes the processor out of reset, and the processor runs the FSBL in on-chip RAM.
  - **Impact on FPGA**—The FPGA portion is left alone during the reset. After exiting reset, software determines whether to reconfigure the FPGA portion.
- **HPS Cold reset and trigger a remote Update**
  - **Impact on HPS**—The SDM holds the processor in reset. The SDM loads the FSBL from the next valid *.pof image or factory image into the HPS on-chip memory. The *.pof contains the data to configure the FPGA portion of the SoC and the FSBL payload. When successfully completed, the SDM releases the HPS from reset and the processor begins executing code from the reset exception address.
  - **Impact on FPGA**—The FPGA portion is first erased, then reconfigured with the next valid Core RBF or Factory Core RBF. There must always be a valid factory RBF present.

5.12. Flash Considerations

5.12.1. Flash Programming Method

The flash connected to SDM is programmed using the Intel Quartus Prime Programmer tool, that is part of both Intel Quartus Prime Pro Edition and Intel SoC FPGA Embedded Development Suite (SoC EDS).

GUIDELINE: Use Intel Quartus Prime Pro Edition Programmer to write to SDM flash.

It is your responsibility to program the flash connected to HPS. Several options are possible:

- Use a bus switch to route the flash signals to an external master that does the programming.
- Use software running on HPS to do the programming. For example U-Boot can be loaded with an Arm debugger or System Console, then used to program the flash.

GUIDELINE: Plan for the HPS flash programming method early in the project lifecycle, as it may impact board design or require additional tool support.

5.12.2. Using a Single flash for Both FPGA Configuration and HPS Mass Storage

The QSPI device connected to the SDM can also be accessed directly by the HPS. However, there is a significant speed penalty when doing so. It is up to you to decide whether the speed penalty is acceptable for the end application.

For reference, here are some performance numbers:

- Maximum HPS eMMC read speed: 50Mbytes/s
- Maximum HPS SD read speed: 25Mbytes/s
- Maximum HPS read speed from SDM QSPI: 4Mbytes/s

GUIDELINE: For best performance, Intel recommends to use a flash device connected to HPS for mass storage by HPS.

5.13. Embedded Software Debugging and Trace

The HPS Debug Access Port (DAP) can be accessed through dedicated HPS pins configured as JTAG, or it can be accessed through FPGA JTAG interface pins.

The option to access the HPS JTAG interface through the FPGA JTAG pins is available in the Intel Quartus Prime Pro Edition project.

At power up, the FPGA appears as the first device in the JTAG chain. Once the FPGA is configured with an image for which the HPS JTAG interface is made available to the FPGA JTAG pins, the HPS appears as the first interface in the JTAG chain; and the FPGA appears as the second interface. This requires different connection settings for the FPGA tools, like the Intel Quartus Prime Pro Edition Programmer when it is used at power up and after FPGA configuration.

GUIDELINE: You must have an available JTAG connection to the board that can be used for development as well as to debug and diagnose field issues.

The HPS offers two trace interfaces either through HPS Dedicated I/O or FPGA I/O. The interface through HPS Dedicated 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).

To improve the trace bandwidth, you can use the standard trace interface which is a 32-bit single data rate interface to the FPGA.

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.

5.14. Embedded Software Design Guidelines for Intel Stratix 10 SoC FPGAs Revision History

Table 15. Embedded Software Design Guidelines for Stratix 10 SoC FPGAs Revision History

Document Version	Changes
2018.12.24	Added a link to the *Intel Stratix 10 SoC FPGA Boot User Guide*.
2018.09.24	Added a new section titled: "System Reset Considerations" that covers the three types of system resets that the SDM performs.
2018.05.07	The following content was added: • Assembling Your Software Development Platform for: — Linux — Bare-Metal Application — PartnerOS or RTOS • Choosing the Bootloader Software • Selecting Software Tools for Development, Debug and Trace • Boot and Configuration Considerations • Flash Considerations • Embedded Software Debugging and Trace
2017.11.06	Initial release

6. Recommended Resources for Stratix 10 SoC FPGAs

6.1. Device Documentation

- Intel Stratix 10 Hard Processor System Technical Reference Manual
- Intel Stratix 10 Device Datasheet
- Documentation: Pin-Out Files for FPGA Devices
- Hard Processor System Pin Information for Intel Stratix 10 Devices (version 2017.10.09)
- Intel Stratix 10 Device Family Pin Connection Guidelines
- Intel Stratix 10 Device Design Guidelines
- Intel Stratix 10 High-Performance Design Handbook
- External Memory Interfaces Intel Stratix 10 FPGA IP User Guide
- Intel Stratix 10 External Memory Interface Pin Information for Devices (version 2018.1.23)
- AN 692: Power Sequencing Considerations for Intel Cyclone® 10 GX, Intel Arria® 10, and Intel Stratix 10 Devices
- Differences Among Intel SoC Device Families
- Intel Stratix 10 SoC FPGA Boot User Guide
- Early Power Estimator for Intel Stratix 10 FPGAs User Guide

Contact your local Intel FPGA support representative for the following:

- *Intel Stratix 10 SX ES Device Errata*

6.2. Tools and Software Web Pages

- Arm DS-5 Intel SoC FPGA Edition Toolkit
- Intel SoC FPGA Embedded Development Suite (SoC EDS)
- Intel Quartus Prime Software Suite
- Mentor Graphics* Embedded for Intel
- External Memory Interface (EMIF) Spec Estimator
- RocketBoards.org - Linux on SoC FPGAs

6.3. Recommended Resources for Stratix 10 SoC FPGAs Revision History

Table 16. Recommended Resources for Stratix 10 SoC FPGAs Revision History

Document Version	Changes
2018.12.24	Added a link to the *Intel Stratix 10 SoC FPGA Boot User Guide*.
2018.09.24	Maintenance release
2017.11.06	Initial release