Contents

1. Introduction to the Intel® Agilex™ Hard Processor System Component ....................... 3
   1.1. Cortex*-A53 MPCore* Processor .............................................................................. 3
   1.2. CoreSight* Debug Components............................................................................... 4
   1.3. Interconnect ........................................................................................................ 4
   1.4. FPGA Bridges ........................................................................................................ 5
   1.5. Memory Controllers ............................................................................................... 5
   1.6. Support Peripherals ............................................................................................... 5
       1.6.1. Interface Peripherals ................................................................................. 6
       1.6.2. On-Chip Memories .................................................................................... 7
   1.7. Introduction to the HPS Component Revision History.................................................. 7

2. Configuring the Intel Agilex Hard Processor System Component ................................. 8
   2.1. Parameterizing the HPS Component ....................................................................... 8
   2.2. FPGA Interfaces .................................................................................................. 10
       2.2.1. General Interfaces .................................................................................. 11
       2.2.2. HPS-FPGA AXI Bridges.............................................................................. 15
       2.2.3. HPS Boot Source ...................................................................................... 16
       2.2.4. DMA Controller Interface ........................................................................... 17
       2.2.5. Interrupts .............................................................................................. 18
   2.3. HPS Clocks and Resets ........................................................................................ 20
       2.3.1. Input Clocks ............................................................................................ 21
       2.3.2. Internal Clocks and Output Clocks .............................................................. 25
       2.3.3. Resets .................................................................................................... 27
   2.4. HPS EMIF ............................................................................................................ 27
   2.5. I/O Delays .......................................................................................................... 29
   2.6. Pin MUX and Peripherals ...................................................................................... 31
       2.6.1. Pin Mux GUI ............................................................................................ 31
       2.6.2. Pin Mux Report ......................................................................................... 34
       2.6.3. EMAC ptp Interface .................................................................................. 34
   2.7. Generating and Compiling the HPS Component ....................................................... 34
   2.8. Using the Address Span Extender Component ......................................................... 35
   2.9. Configuring the HPS Component Revision History ..................................................... 36
1. Introduction to the Intel® Agilex™ Hard Processor System Component

The hard processor system (HPS) component is a wrapper that interfaces logic in your design to the:

- HPS hard logic
- Simulation models
- Bus functional models (BFMs)
- Software handoff files

The HPS component instantiates the HPS hard logic in your design and enables other soft components to interface with the HPS hard logic. The HPS component has a small footprint in the FPGA fabric, as the component only serves to enable soft logic to hard logic connection in the HPS.

After you connect the soft logic to the HPS, you can use Platform Designer to ensure:

- Interoperability by adapting Avalon® Memory-Mapped (Avalon-MM) interfaces to AXI*
- Handling of data width mismatches and clock domain transfer crossings

You are able to interface your Intel®, customer, or third party FPGA core IP to the HPS without the creation of integration logic. This reference manual details the interfaces exposed and configured by the options in the component.

For more information about the HPS system architecture and features, refer to the "Introduction to the Hard Processor" chapter in the Intel Agilex™ Hard Processor System Technical Reference Manual.

Related Information

- Introduction to the Hard Processor System
  For more information, refer to this chapter in the Intel Agilex Hard Processor System Technical Reference Manual
- Intel Agilex Device Data Sheet

1.1. Cortex*-A53 MPCore* Processor

The HPS includes an Arm* Cortex*-A53 MPCore* Processor that is composed of four Armv8-A architecture central processing units (CPUs).
1.2. CoreSight* Debug Components

The following lists the Arm CoreSight* debug components:

- Debug Access Port (DAP)
- System Trace Macrocell (STM)
- Embedded Trace FIFO (ETF)
- AMBA* Trace Bus Replicator
- Embedded Trace Router (ETR)
- Trace Port Interface Unit (TPIU)
- Embedded Cross Trigger (ECT)

1.3. Interconnect

The HPS system interconnect supports the following features:

- Configurable Arm TrustZone*-compliant firewall and security support.
  - For each peripheral, implements secure or non-secure access.
  - Allows configuration of individual transactions as secure or non-secure at the initiating master.
- Multi-tiered bus structure to separate high bandwidth masters from lower bandwidth peripherals and control and status ports.
- Quality of service (QoS) with three programmable levels of service on a per master basis.
- On-chip debugging and tracing capabilities. The system interconnect is based on the Arteris* FlexNoC* network-on-chip (NoC) interconnect technology.
1.4. FPGA Bridges

The FPGA bridges provide a variety of communication channels between the HPS and the FPGA fabric. The HPS is highly integrated with the FPGA fabric, resulting in thousands of connecting signals. Some of the HPS—FPGA interfaces include:

- FPGA-to-SoC bridge
- SoC-to-FPGA bridge
- Lightweight SoC-to-FPGA bridge

Related Information
HPS-FPGA Bridges
For more information, refer to this chapter in the *Intel Agilex Hard Processor System Technical Reference Manual*.

1.5. Memory Controllers

The HPS includes two memory controller peripherals.
- NAND Flash Controller
- SD/MMC Controller

Related Information
- NAND Flash Controller
  For more information, refer to this chapter in the *Intel Agilex Hard Processor System Technical Reference Manual*.
- SD/MMC Controller
  For more information, refer to this chapter in the *Intel Agilex Hard Processor System Technical Reference Manual*.

1.6. Support Peripherals

- Clock Manager
- Reset Manager
- System Manager
- Timer
- Watchdog Timer
- Direct Memory Access (DMA) Controller
- Error Checking and Correction Controller

Related Information
- Clock Manager
  For more information about the support peripherals, refer to its corresponding chapter in the *Intel Agilex Hard Processor System Technical Reference Manual*.
- Reset Manager
  For more information about the support peripherals, refer to its corresponding chapter in the *Intel Agilex Hard Processor System Technical Reference Manual*. 
• System Manager
  For more information about the support peripherals, refer to its corresponding chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

• Timer
  For more information about the support peripherals, refer to its corresponding chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

• Watchdog Timer
  For more information about the support peripherals, refer to its corresponding chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

• DMA Controller
  For more information about the support peripherals, refer to its corresponding chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

• Error Checking and Correction Controller
  For more information about the support peripherals, refer to its corresponding chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

1.6.1. Interface Peripherals

• Ethernet Media Access Controllers (EMAC)
• USB 2.0 On-The-Go (OTG) Controllers
• I\(^2\)C Controllers
• UARTs
• SPI Master Controllers
• SPI Slave Controllers
• GPIO Interfaces

Related Information

• Ethernet Media Access Controller

• USB 2.0 OTG Controller

• SPI Controller

• I\(^2\)C Controller

• UART Controller

• General-Purpose I/O Interface
1.6.2. On-Chip Memories

On-Chip RAM is the only on-chip memory.

Related Information
On-Chip Memory
For more information, refer to this chapter in the Intel Agilex Hard Processor System Technical Reference Manual

1.7. Introduction to the HPS Component Revision History

Table 1. Document Revision History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.09.30</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
2. Configuring the Intel Agilex Hard Processor System Component

This chapter describes the parameters available and the interfaces enabled by those parameters in the hard processor system (HPS) component parameter editor. The parameter editor opens when you add or edit an Intel Agilex HPS component in Platform Designer.

2.1. Parameterizing the HPS Component

1. Install the current version of the Intel Quartus® Prime Pro Edition design software, along with Intel Agilex device support.

   Instructions on how to install the current version are located at: https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html

2. Open the Intel Quartus Prime software.

   Figure 1.  Intel Quartus Prime Starting Screen

4. Select an existing Intel Quartus Prime project and Platform Designer system or create new files. Ensure that Intel Agilex device is selected in the **Device Family** dropdown, and an appropriate device is selected in the **Device Part** dropdown.

5. In the **IP Catalog** tab, under Library, select **Processors and Peripherals ➤ Hard Processor Systems ➤ Hard Processor System Intel Agilex FPGA IP**.
2.2. FPGA Interfaces

The **FPGA Interfaces** tab allows you to specify options for the primary interfaces between the FPGA and the HPS. The following groups in this tab are:
2. Configuring the Intel Agilex Hard Processor System Component

2.2. General Interfaces

2.2.1. Enable MPU Standby and Event Interfaces

Microprocessor Unit (MPU) standby signals are notification signals to the FPGA fabric that the MPU is in standby. Event signals wake up the Cortex-A53 processors from a wait-for-event (WFE) state. Turning on the Enable MPU Standby and Event Interfaces option enables the h2f_mpu_events conduit, which is comprised of the following signals:
• **h2f_mpu_eventi**—Input for FPGA to signal events to all processors. This FPGA-to-HPS signal is used to wake up a processor that is in a WFE state. Asserting this signal has the same effect as executing the SEV instruction in the Cortex-A53. You must deassert the signal until the FPGA fabric configures. You must assert the signal high for at least two MPU clock cycles for the processor to recognize any of the Cortex-A53 cores.

• **h2f_mpu_evento**—Output from any MPU core into the FPGA fabric. This HPS-to-FPGA signal is asserted when an SEV instruction is executed by one of the Cortex-A53 processors. This signal is output as a multiple cycle pulse so logic in the FPGA should use a rising edge detector circuit to detect the occurrence of the event.

• **h2f_mpu_standbywfe[3:0]**—Output per processor that indicates if the processor is in WFE standby mode. When high, the processor is in WFE standby mode.

• **h2f_mpu_standbywfi[3:0]**—Output per processor that indicates if the processor is in the wait-for-interrupt (WFI) standby mode. When the logic level is high, the processor is in WFI standby mode.

Figure 6. **Platform Designer Enable MPU Signals**

2.2.1.2. Enable General Purpose Signals

Turning on the **Enable General Purpose Signals** option enables the **h2f_gp** conduit, which is comprised of a pair of 32-bit uni-directional general-purpose interfaces between the HPS System Manager and the FPGA fabric. These signal names are **h2f_gp_in** and **h2f_gp_out**, and are inputs to the HPS and outputs from the HPS, respectively.
2.2.1.3. Enable Debug APB Interface

The debug Advanced Peripheral Bus (APB)* interface allows debug components in the FPGA fabric to access debug components in the HPS.


Turning on this option enables the following interfaces and signals:

Table 2. APB Interfaces and Signals

<table>
<thead>
<tr>
<th>Interface Name</th>
<th>Interface Type</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>h2f_debug_apb_clock</td>
<td>Clock Input</td>
<td>h2f_dbg_apb_clk</td>
</tr>
<tr>
<td>h2f_debug_apb_reset</td>
<td>Reset Output</td>
<td>h2f_dbg_apb_rst_n</td>
</tr>
<tr>
<td>h2f_debug_apb</td>
<td>APB Master</td>
<td>h2f_dbg_apb_PADDR[14..0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PADDR31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PENABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PRDATA[31..0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PREADY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PSEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PSLVERB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PWDATA[31..0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dbg_apb_PWRITE</td>
</tr>
<tr>
<td>h2f_debug_apb_sideband</td>
<td>Conduit</td>
<td>h2f_debug_apb_PCLKEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_debug_apb_DBG_APB_DISABLE</td>
</tr>
</tbody>
</table>

Related Information

CoreSight Debug and Trace
For more information about the interfaces described in this section, refer to the “CoreSight Debug and Trace” chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

2.2.1.4. Enable System Trace Macrocell (STM) Hardware Events

The system trace microcell hardware events interface allows logic in the FPGA to insert messages into the trace stream.


Turning on the Enable System Trace Macrocell Hardware Events option enables the h2f_cs conduit and the f2h_stm_hwevents conduit, which is comprised of the single bus f2h_stm_hwevents[43...0].

Related Information

CoreSight Debug and Trace
For more information about the interfaces described in this section, refer to the “CoreSight Debug and Trace” chapter in the Intel Agilex Hard Processor System Technical Reference Manual.
2.2.1.5. Enable FPGA Cross Trigger Interface

The cross trigger interface (CTI) allows trigger sources and sinks in FPGA logic to interface with the embedded cross trigger (ECT).

For more information about the FPGA Cross Trigger interface, refer to the “CoreSight Debug and Trace” chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

If this interface must be connected to a Signal Tap II instance in the FPGA fabric, then it must be left disabled in Platform Designer. Turning on the Enable FPGA Cross Trigger Interface option enables the h2f_cti conduit, which is comprised of the following signals:

- h2f_cti_trig_in [7..0]
- h2f_cti_trig_out_ack [7..0]
- h2f_cti_trig_out [7..0]
- h2f_cti_trig_in_ack [7..0]

Related Information

CoreSight Debug and Trace
For more information about the interfaces described in this section, refer to the “CoreSight Debug and Trace” chapter in the Intel Agilex Hard Processor System Technical Reference Manual.

2.2.1.6. Enable DDR Arm Trace Bus (ATB)

Turning on the Enable DDR Arm Trace Bus option enables the ddr_atb_clock clock input and ddr_atb_reset reset input interfaces.

Related Information

CoreSight Debug and Trace
For more information about the interfaces described in this section, refer to the “CoreSight Debug and Trace” chapter in the Intel Agilex Hard Processor System Technical Reference Manual.
2.2.2. HPS-FPGA AXI Bridges

2.2.2.1. FPGA-to-HPS Slave Interface

The FPGA-to-HPS slave interface allows FPGA masters to issue transactions to the HPS. You can use the:

- **Interface specification** drop-down to configure this master interface to AXI-4 or ACE-lite.
- **Enable/Data Width** drop-down to configure this master interface's data widths to 128-, 256-, or 512-bit.
- **Interface address width** is configurable from 40 bits down to 20 bits, which allows the FPGA fabric to access the majority of the HPS address space. To facilitate masters in the FPGA logic with a smaller address width than the bridge in accessing the HPS address space, you can use the Intel Address Span Extender component.
- **Interface destination** configures routing for transactions. This interface supports routing to the Cache Coherency Unit (CCU) directly, SDRAM directly (bypasses the CCU), or a Custom configuration controlled by the FPGA AXI-master.

For more information, refer to the "Using the Address Span Component Extender" chapter.

When this bridge is enabled, the interfaces `f2h_axi_slave`, `f2h_axi_clock`, and `f2h_axi_reset` are made available.

This interface allows the FPGA to access the majority of the HPS slaves. When configured as an ACE-lite slave, this interface provides a coherent memory interface. Other interface standards in the FPGA fabric, such as connecting to Avalon Memory
Mapped (Avalon-MM) interfaces, can be supported through the use of soft logic adapters. The Platform Designer system integration tool automatically generates adapter logic to connect AXI to Avalon-MM interfaces.

Note: The `hps_emif` conduit is enabled when either the AXI or ACE-Lite bridge is selected.


Related Information
- Using the Address Span Extender Component on page 35
- HPS-FPGA Bridges

2.2.2.2. HPS to FPGA AXI-4 Master Interface

The HPS-to-FPGA AXI-4 Master interface allows HPS masters to issue transactions to the FPGA fabric. You can use the:

- **Enable/Data Width** dropdown to configure this master interface's data widths to 32-, 64-, or 128-bit.
- **Bridge address width** is configurable from 32 bits down to 20 bits. When this bridge is enabled, the interfaces `h2f_axi_master`, `h2f_axi_clock`, and `h2f_axi_reset` are made available.

This bridge accepts a clock input from the FPGA fabric and performs clock domain crossing internally. The exposed AXI interface operates on the same clock domain as the clock supplied by the FPGA fabric. Other interface standards in the FPGA fabric, such as connecting to Avalon-MM interfaces, can be supported through the use of soft logic adapters. The Platform Designer system integration tool automatically generates adapter logic to connect AXI to Avalon-MM interfaces.

2.2.2.3. Lightweight HPS to FPGA Master Interface

The lightweight HPS-to-FPGA interface, a low-bandwidth control interface, allows HPS masters to issue transactions to the FPGA fabric. The **Enable/Data Width** dropdown is thus limited to a fixed 32-bit data width. The **Bridge address width** is configurable to either 21 bits or 20 bits. When this bridge is enabled, the interfaces `h2f_lw_axi_master`, `h2f_lw_axi_clock`, and `h2f_lw_axi_reset` are made available.

This bridge accepts a clock input from the FPGA fabric and performs clock domain crossing internally. The exposed AXI interface operates on the same clock domain as the clock supplied by the FPGA fabric. Other interface standards in the FPGA fabric, such as connecting to Avalon-MM interfaces, can be supported through the use of soft logic adapters. The Platform Designer system integration tool automatically generates adapter logic to connect AXI to Avalon-MM interfaces.

2.2.3. HPS Boot Source

The **HPS SSBL Location** dropdown allows you to choose one of three sources for the HPS Second Stage Bootloader:
• Use the boot flash as used by the SDM
• Use HPS SD/MMC flash
• Use HPS NAND flash

**Figure 8. Platform Designer HPS Boot Source**

2.2.4. DMA Controller Interface

The DMA controller interface allows soft IP in the FPGA fabric to communicate with the DMA controller in the HPS. You can configure up to eight separate interface channels by clicking on the dropdown in the **Enabled** column for the corresponding channel row. Each DMA peripheral request interface conduit $f2h\text{\_}dma<n>$ contains the following three signals, where $<n>$ corresponds to a specific request interface enabled in Platform Designer:

- $f2h\text{\_}dma<n>\_req$—This signal is used to request burst transfer using the DMA
- $f2h\text{\_}dma<n>\_single$—This signal is used to request single word transfer using the DMA
- $f2h\text{\_}dma<n>\_ack$—This signal indicates the DMA acknowledgment upon requests from the FPGA

*Note:* FPGA DMA interfaces 6 and 7 are multiplexed with the **EMAC2 I2C DMA** interface.
2.2.5. Interrupts

The Interrupts section is divided into two subsections, FPGA-to-HPS and HPS-to-FPGA.

Figure 9.  Platform Designer DMA Peripheral Request

Figure 10.  Platform Designer Interrupts
2. Configuring the Intel Agilex Hard Processor System Component

2.2.5.1. FPGA-to-HPS

Turning on the **Enable FPGA-to-HPS Interrupts** option configures the HPS component to provide 64 general purpose FPGA-to-HPS interrupts, allowing soft IP in the FPGA fabric to trigger interrupts to the MPU’s generic interrupt controller (GIC). The interrupts are implemented through the following 32-bit interfaces:

- `f2h_irq0`—FPGA-to-HPS interrupts 0 through 31
- `f2h_irq1`—FPGA-to-HPS interrupts 32 through 63

The FPGA-to-HPS interrupts are asynchronous on the FPGA interface. Inside the HPS, the interrupts are synchronized to the MPU’s internal peripheral clock (`mpu_periph_clk`).

2.2.5.2. HPS-to-FPGA

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Description</th>
<th>Interface Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Clock Peripheral Interrupts</td>
<td>Enables interface for HPS clock manager and MPU wake-up interrupt signals to the FPGA</td>
<td>h2f_clkmgr_interrupt</td>
</tr>
<tr>
<td>Enable DMA Interrupts</td>
<td>Enables interface for HPS DMA channels interrupt and DMA abort interrupt to the FPGA</td>
<td>h2f_dma_interrupt0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_interrupt7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_dma_abort_interrupt</td>
</tr>
<tr>
<td>Enable EMAC Interrupts</td>
<td>Enables interface for HPS Ethernet MAC controller interrupt to the FPGA. EMAC must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_emac0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_emac1_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_emac2_interrupt</td>
</tr>
<tr>
<td>Enable GPIO Interrupts</td>
<td>Enables interface for the HPS general purpose IO (GPIO) interrupt to the FPGA</td>
<td>h2f_gpio0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_gpio1_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_gpio2_interrupt</td>
</tr>
<tr>
<td>Enable I2C-EMAC Interrupts</td>
<td>Enable the HPS peripheral interrupt for I2CEMAC to be driven into the FPGA fabric</td>
<td>h2f_i2c_emac0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_i2c_emac1_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_i2c_emac2_interrupt</td>
</tr>
<tr>
<td>Enable I2C Peripherals Interrupts</td>
<td>Enable the HPS peripheral interrupt for I2C0 to be driven into the FPGA fabric. The I2C must be enabled in the Pin Mux Tab before enabling interrupt.</td>
<td>h2f_i2c0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_i2c1_interrupt</td>
</tr>
<tr>
<td>Enable L4 Timer Interrupts</td>
<td>Enables the HPS peripheral interrupt for L4TIMER to be driven into the FPGA fabric.</td>
<td>h2f_timer_l4sp_0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_timer_l4sp_1_interrupt</td>
</tr>
</tbody>
</table>

*continued...*
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Description</th>
<th>Interface Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable NAND Interrupts</td>
<td>Enables interface for the HPS NAND controller interrupt to the FPGA. The NAND IP Block must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_nand_interrupt</td>
</tr>
<tr>
<td>Enable SYS Timer Interrupts</td>
<td>Enables the HPS peripheral interrupt for SYSTIMER to be driven into the FPGA fabric.</td>
<td>h2f_timer_sys_0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_timer_sys_1_interrupt</td>
</tr>
<tr>
<td>Enable SD/MMC Interrupts</td>
<td>Enables interface for the HPS SD/MMC controller interrupt to the FPGA. The SD/MMC IP Block must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_sdmmc_interrupt</td>
</tr>
<tr>
<td>Enable SPI Master Interrupts</td>
<td>Enables interface for the HPS SPI master controller interrupt to the FPGA. The SPI Master IP Block must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_spim0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_spim1_interrupt</td>
</tr>
<tr>
<td>Enable SPI Slave Interrupts</td>
<td>Enables interface for the HPS SPI slave controller interrupt to the FPGA. The SPI IP Block must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_spis0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_spis1_interrupt</td>
</tr>
<tr>
<td>Enable ECC/Parity_L1 Interrupts</td>
<td>Enables the HPS peripheral interrupt for ECC single and double bit error and L1 parity error to be driven into the FPGA fabric.</td>
<td>h2f_ecc_serr_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_ecc_derr_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_parity_l1_interrupt</td>
</tr>
<tr>
<td>Enable UART Interrupts</td>
<td>Enables interface for the HPS UART controller interrupt to the FPGA. The UART IP Block must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_uart0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_uart1_interrupt</td>
</tr>
<tr>
<td>Enable USB Interrupts</td>
<td>Enables interface for the HPS USB controller interrupt to the FPGA. The USB IP Block must be enabled in Pin Mux Tab before enabling interrupt.</td>
<td>h2f_usb0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s2f_usb1_interrupt</td>
</tr>
<tr>
<td>Enable Watchdog Interrupts</td>
<td>Enables interface for the HPS watchdog interrupt to the FPGA</td>
<td>h2f_wdog0_interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h2f_wdog1_interrupt</td>
</tr>
</tbody>
</table>

### 2.3. HPS Clocks and Resets

**HPS Clocks and Reset** is the second of five tabs in the HPS component and is made up of three tabs: Input Clocks, Internal Clocks and Output Clocks, and Resets.
2.3.1. Input Clocks

The **Input Clocks** tab is comprised of three subsections: **External Clock Source**, **FPGA-to-HPS Clocks Source**, and **Peripheral FPGA Clocks**.

Figure 11. Platform Designer HPS Clocks and Resets Tab

[Diagram of Platform Designer HPS Clocks and Resets Tab]

Figure 12. Platform Designer Input Clocks Sub-Tab

[Diagram of Platform Designer Input Clocks Sub-Tab]
2.3.1.1. External Clock Source

The EOSC clock frequency field is used to specify the frequency of the input clock to the hps_osc_clk pin that drives the main HPS PLL.

Figure 13. Platform Designer External Clock Source Sub-Window

[Platform Designer External Clock Source Sub-Window Image]

Related Information
Intel Agilex Device Data Sheet

2.3.1.2. FPGA-to-HPS Clocks Source

Turning on the Enable FPGA-to-HPS free clock option enables the f2h_free_clk clock input. This is an alternative input to the main HPS PLL driven from the FPGA fabric instead of the dedicated hps_osc_clk pin. Turning on the Enable FPGA-to-HPS free clock option is subject to the same requirements as that pin.
Figure 14.  Platform Designer FPGA-to-HPS Clocks Source Sub-Window

For more information about the requirements for this clock, refer to the *Intel Agilex Device Datasheet*.

**Related Information**

Intel Agilex Device Data Sheet
2.3.1.3. Peripheral FPGA Clocks

Figure 15. Platform Designer Peripheral FPGA Clocks Sub-Window

The table below provides a description for each of the parameters in the "Peripheral FPGA Clocks" sub-window.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMAC 0 (emac0_md_clk clock frequency)</td>
<td>If EMAC 0 peripheral is routed to FPGA, use the input field to specify EMAC 0 MDIO clock frequency</td>
</tr>
<tr>
<td>EMAC 0 (emac0_gtx_clk clock frequency)</td>
<td>If EMAC 0 peripheral is routed to FPGA, use the input field to specify EMAC 0 transmit clock frequency</td>
</tr>
<tr>
<td>EMAC 1 (emac1_md_clk clock frequency)</td>
<td>If EMAC 1 peripheral is routed to FPGA, use the input field to specify EMAC 1 MDIO clock frequency</td>
</tr>
<tr>
<td>EMAC 1 (emac1_gtx_clk clock frequency)</td>
<td>If EMAC 1 peripheral is routed to FPGA, use the input field to specify EMAC 1 transmit clock frequency</td>
</tr>
<tr>
<td>EMAC 2 (emac2_md_clk clock frequency)</td>
<td>If EMAC 2 peripheral is routed to FPGA, use the input field to specify EMAC 2 MDIO clock frequency</td>
</tr>
<tr>
<td>EMAC 2 (emac2_gtx_clk clock frequency)</td>
<td>If EMAC 2 peripheral is routed to FPGA, use the input field to specify EMAC 2 transmit clock frequency</td>
</tr>
<tr>
<td>SD/MMC (sdmmc_cclk)</td>
<td>If this peripheral pin multiplexing is configured to route to FPGA fabric, use the input field to specify the SD/MMC sdmmc_cclk clock frequency</td>
</tr>
<tr>
<td>SPI M 0 (spim0_sclk_out clock frequency)</td>
<td>If SPI master 0 peripheral is routed to FPGA, use the input field to specify SPI master 0 output clock frequency</td>
</tr>
<tr>
<td>SPI M 1 (spim1_sclk_out clock frequency)</td>
<td>If SPI master 1 peripheral is routed to FPGA, use the input field to specify SPI master 1 output clock frequency</td>
</tr>
</tbody>
</table>

continued...
### 2.3.2. Internal Clocks and Output Clocks

The **Internal Clocks and Output Clocks** tab is comprised of five subsections: **Main PLL Output Clocks – Desired Frequencies**, **HPS to FPGA User Clocks**, **HPS Peripheral Clocks – Desired Frequencies**, **Clock Sources**, and **PLL Report**.

#### Figure 16. Platform Designer Internal Clocks and Output Clocks Sub-Tab

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C0 (i2c0_clk clock frequency)</td>
<td>If I²C 0 peripheral is routed to FPGA, use the input field to specify I²C 0 output clock frequency</td>
</tr>
<tr>
<td>I²C1 (i2c1_clk clock frequency)</td>
<td>If I²C 1 peripheral is routed to FPGA, use the input field to specify I²C 1 output clock frequency</td>
</tr>
<tr>
<td>I²CEMAC0 (i2cemac0_clk)</td>
<td>If this peripheral pin multiplexing is configured to route to the FPGA fabric, use the input field to specify the I²CEMAC0 i2cemac0_clk clock frequency</td>
</tr>
<tr>
<td>I²CEMAC1 (i2cemac1_clk)</td>
<td>If this peripheral pin multiplexing is configured to route to the FPGA fabric, use the input field to specify the I²CEMAC1 i2cemac1_clk clock frequency</td>
</tr>
<tr>
<td>I²CEMAC2 (i2cemac2_clk)</td>
<td>If this peripheral pin multiplexing is configured to route to the FPGA fabric, use the input field to specify the I²CEMAC2 i2cemac2_clk clock frequency</td>
</tr>
</tbody>
</table>

#### 2.3.2.1. Main PLL Output Clocks – Desired Frequencies

This section allows you to control the MPU clock frequency. The **Default MPU clock frequency** field displays the default maximum frequency for the MPU based on the device speed grade selected. You may check the **Override default MPU clock frequency** box to manually enter a slower frequency than the default MPU clock frequency, in the **Custom MPU clock frequency** field.
For more information about the maximum MPU clock frequencies, refer to the Intel Agilex Device Data Sheet.

**Related Information**
Intel Agilex Device Data Sheet

### 2.3.2.2. HPS to FPGA User Clocks

Turning on the Enable HPS-to-FPGA User0 clock or Enable HPS-to-FPGA User1 clock option enables one of two available HPS PLL outputs into the FPGA. You can connect a user clock to logic that you instantiate in the FPGA. When you enable a HPS-to-FPGA user clock, the clock frequency field displays the default maximum frequency for the user clock based on the device speed grade selected. User clocks can be manually overridden and driven from peripheral PLL or Main PLL.

### 2.3.2.3. HPS Peripheral Clocks – Desired Frequencies

The clock frequencies you provide in this section are reported in a Synopsys* Design Constraints File (.sdc) generated by Platform Designer. The .sdc file is referenced in the system .qip file when the system is generated. The grayed out boxes show the frequencies of the various clocks and can only be changed by changing the L3 source clock frequency or by changing the respective clock divider.

*Note:* GUI interface for this feature will change for Intel Quartus Prime Pro Edition version 19.3.

**Related Information**
Intel Agilex Device Data Sheet

### 2.3.2.4. Clock Sources

The drop-downs in this section control multiplexers in the HPS clock manager to select the source for the corresponding PLL or clock. Some of the drop-downs are enabled only when the corresponding peripherals are enabled. The FPGA to HPS Free clock is available as an option in these drop-downs when it is enabled on the Input Clocks tab.

*Note:* If you intend to use the FPGA to HPS free clock as the input to the hps_osc_clk pin, you must select that option for the Main PLL reference clock source and Peripheral PLL reference clock source.

You also have the option to override selected clock sources, by enabling the Override selected clock sources option.

### 2.3.2.5. PLL Report

This section is an informational section displaying the calculated parameters for the HPS PLLs and frequencies for the main and peripheral clocks.
2.3.3. Resets

Figure 17. Platform Designer Reset Sub-Tab

• Turning on the **Enable HPS warm reset handshake signals** option enables an additional pair of reset handshake signals allowing soft logic to notify the HPS when it is safe to initiate a warm reset in the FPGA fabric. Turning on this option exposes the `h2f_warm_reset_handshake` conduit, which is comprised of the signals `h2f_pending_rst_req_n` and `f2h_pending_rst_ack_n`.

• Turning on the **Enable HPS-to-FPGA cold reset output** option exposes the `h2f_coldreset` reset output interface. This signal is asserted when the HPS undergoes a cold reset.

• Turning on the **Enable watchdog reset** option exposes the `h2f_watchdog_rst` reset output interface and is asserted when the HPS watchdog timers are triggered.

• The **How SDM handles HPS watchdog reset** dropdown provides an input to the compiled bitstream that directs the SDM to treat the HPS watchdog reset assertion as an **HPS Cold reset**, **HPS warm reset**, or **HPS Cold reset and trigger a remote update**.

2.4. HPS EMIF

The HPS supports one DDR4 interface.

**Note:**

• In Intel Quartus Prime Pro Edition version 19.2, the HPS EMIF conduit can only be disabled if both the FPGA-to-HPS interface is unused, and the SDRAM tab enable box is unchecked.

• In Intel Quartus Prime Pro Edition version 19.3, the SDRAM tab will be removed, and the HPS EMIF conduit will be enabled or disabled through the FPGA-to-HPS interface used or unused.
### Figure 18. Platform Designer Displaying the hps_emif Conduit for FPGA Interfaces Tab

![Platform Designer Displaying hps_emif Conduit for FPGA Interfaces Tab]

### Figure 19. Platform Designer Displaying hps_emif in System View

![Platform Designer Displaying hps_emif in System View]
2. Configuring the Intel Agilex Hard Processor System Component

2.5. I/O Delays

The I/O Delays tab is the fourth of five tabs in the HPS component that allows you to add an optional delay chain to the input or output of any of the 48 HPS dedicated I/O pins. Each dropdown allows you to select between the following options for the corresponding I/O pin:

- Zero_chain_dly—input or output signal bypasses the delay chain
- Chain_dly—input or output signal goes through the minimum delay chain path
- One_chain_dly to thirty_chain_dly—input or output signal goes through between one to thirty chain delays, in addition to the minimum delay chain path

Related Information

- **HPS EMIF Design Considerations**
  For more information, refer to this section in the *Intel Agilex SoC Device Design Guidelines*.

- **External Memory Interface**
  For more information, refer to this section in the *Intel FPGAs and Programmable Devices support web page*.

- **External Memory Interfaces Intel Agilex FPGA IP User Guide**
  For more information about External Memory Interface for HPS Intel Agilex FPGA IP
Figure 21. Platform Designer I/O Delays

For more information about the delay timings, refer to the Intel Agilex Device Datasheet.

Related Information
Intel Agilex Device Data Sheet
2.6. Pin MUX and Peripherals

The Pin MUX and Peripherals tab contains three sub-windows: Pin Mux GUI, Pin Mux Report, and EMAC ptp interface.

Figure 23. Platform Designer Pin Mux and Peripherals

2.6.1. Pin Mux GUI

The Pin MUX GUI section contains two tabs: Auto-Place IP and Advanced.

2.6.1.1. Auto-Place IP

The Auto-Place IP tab contains a list of HPS peripherals that can be enabled and either routed to the HPS I/Os or to the FPGA.
You can enable the following types of peripherals:

- NAND Flash Controller
- SD/MMC Controller
- Ethernet Media Access Controller
- USB 2.0 OTG Controller
- I²C Controller
- UART Controller
- SPI Master
- SPI Slave
- CoreSight Debug and Trace
- GPIO

For more information about each of these HPS peripherals, refer to the Intel Agilex Hard Processor System Technical Reference Manual.

You can enable one or more instances of each peripheral type by using the dropdown menu next to each peripheral. When enabled, some peripherals also have mode settings specific to their functions. Once you have selected a peripheral, you must click the Apply Selections button in order to enable the selected peripherals. Clicking the Apply Selections button triggers the HPS component to do a best-effort automatic placement of the enabled peripheral signals to the HPS I/Os. This overrides any settings already chosen in the Advanced tab. The results of this placement becomes visible in the I/O Selections section on the right side of the Auto-Place IP tab. Any messages, such as failures to place a peripheral, appears in the message box in the I/O Selections section.
If the NAND, SD/MMC, or TRACE peripherals are enabled, there are further options to specify the desired bit width of the interface routed to the HPS I/Os. If any of the EMACs are enabled, the corresponding **Interface** and **PHY Options** dropdowns becomes available to specify the desired EMAC parameters.

**Related Information**

### 2.6.1.2. Advanced

The **Advanced** tab is divided into two sub-tabs, **Advanced IP Placement** and **Advanced FPGA Placement**.

**Advanced IP Placement**

The **Advanced IP Placement** tab allows you to be more specific about the placement of each peripheral pin in the HPS dedicated I/O quadrant space. Each location has a pulldown selection menu where you can select which peripheral I/O to be routed to the pin location. Each pulldown menu corresponds to the inputs available to the pinmux at that location. Changes to a dropdown only become effective when the **Apply Selections** button is pressed. Changes in the **Advanced IP Placement** tab carry over to the **Auto-Place IP** tab. The **Pin Mux Report** and **EMAC ptp interface** sections are identical to those in the **Auto-Place IP** tab.

**Advanced FPGA Placement**

The **Advanced FPGA Placement** tab allows you to route specific peripherals to the FPGA, if those peripherals were enabled and allocated to the FPGA in the **Auto-Place IP** tab. Similar options for the SD/MMC, NAND, and TRACE bit-width allow you to specify how wide the interfaces should be when routed to the FPGA. Changes to a dropdown only become effective when the **Apply Selections** button is pressed.
Changes in the **Advanced FPGA Placement** tab carry over to the **Auto-Place IP** tab. The **Pin Mux Report** and **EMAC ptp interface** sections are identical to those in the **Auto-Place IP** tab.

**Figure 26. Platform Designer Advanced FPGA Placement Sub-Tab**

2.6.2. Pin Mux Report

The **Pin Mux Report** section details which physical pins of the device map to each HPS I/O location.

2.6.3. EMAC ptp Interface

In the **EMAC ptp interface** section, there are options to turn on for each EMAC to enable the Precision Time Protocol (ptp) FPGA interface. These options are only applicable when an EMAC is routed to the HPS pins. When enabled, the signals `emac<n>_ptp_pps_o`, `emac<n>_ptp_aux_tx_trig_i`, `emac<n>_ptp_tstmp_data`, `emac<n>_ptp_tstmp_en`, as well as the `emac_ppt_ref_clock` clock input interface, are made available. When an EMAC is routed to the FPGA pins, these signals are automatically included in the `emac<n>` conduit.

2.7. Generating and Compiling the HPS Component

The process of generating and compiling an HPS design is very similar to the process for any other Platform Designer project. Perform the following steps:

1. Generate the design with Platform Designer. The generated files include an `.sdc` file containing clock timing constraints. If simulation is enabled, simulation files are also generated.

2. Add `<qsys_system_name>.qip` to the Intel Quartus Prime project. `<qsys_system_name>.qip` is the Intel Quartus Prime IP File for the HPS component, generated by Platform Designer.
Note: Platform Designer generates pin assignments in the .qip file.

**Figure 27. Platform Designer Displaying the Pin Assignments**

3. Perform analysis and synthesis with the Intel Quartus Prime software.
4. Compile the design with the Intel Quartus Prime software.
5. Optionally back-annotate the SDRAM pin assignments, to eliminate pin assignment warnings the next time you compile the design.

### 2.8. Using the Address Span Extender Component

The FPGA-to-SoC bridge memory-mapped interface can be configured to expose their entire address spaces to the FPGA fabric, 132GB and 128GB, respectively. The address span extender component provides a memory-mapped window into the address space that it masters. Using the address span extender, an FPGA master with a smaller address span can access the entire address space exposed by the FPGA bridge.

You can use the address span extender between a soft logic master and an FPGA-to-SoC bridge. This component reduces the number of address bits required for a master to address a memory-mapped slave interface located in the HPS.

In the example shown in the figure below, the bridges in the HPS component are configured for 32-bit wide addresses (4GB address span).
You can also use the address span extender in the HPS-to-FPGA direction, for slave interfaces in the FPGA. In this case, the HPS-to-FPGA bridge exposes a limited, variable address space in the FPGA, which can be paged in using the address span extender.

For example, suppose that the HPS-to-FPGA bridge has a 1-GB span, and the HPS needs to access three independent 1-GB memories in the FPGA portion of the device. To achieve this, the HPS programs the address span extender to access one SDRAM (1-GB) in the FPGA at a time. This technique is commonly called paging or windowing.


**Related Information**


### 2.9. Configuring the HPS Component Revision History

#### Table 5. Document Revision History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.09.30</td>
<td>Initial release</td>
</tr>
</tbody>
</table>