Accessing HPS Devices from the FPGA

For Quartus® Prime 18.1

1 Introduction

This document describes how to connect a bus-mastering device in the FPGA to slave devices in the Hard Processor System (HPS) in Intel® SoC FPGA devices. This allows masters on the FPGA to use HPS resources such as USB, ethernet, SD* card, and more.

Contents:

2 HPS Devices Overview

2.1 Built-In Devices

Table 1 lists the devices that are built into the HPS. These devices provide memory-mapped interfaces which are mapped to addresses within the HPS interconnect's 32-bit (4GB) address space. Any master device connected to the interconnect (such as one that is instantiated in the FPGA) can read and write these interfaces at their respective addresses. For more details about these devices, refer to the document *Cyclone® V Hard Processor System Technical Reference Manual*.

Table 1. HPS Devices

Device	Interface	Base Address
SD/MMC Controller	sdmmc	0xFF704000
Quad SPI Flash Controller	qspiregs	0xFF705000
	qspidata	0xFFA00000
Ethernet Media Access Controller (EMAC)	emac0	0xFF700000
	emac1	0xFF702000
General Purpose I/O (GPIO) Controller	gpio0	0xFF708000
	gpio1	0xFF709000
	gpio2	0xFF70A000
NAND Flash Controller	nanddata	0xFF900000
	nandregs	0xFFB80000
USB OTG Controller	usb0	0xFFB00000
	usb1	0xFFB40000
CAN Controller	can0	0xFFC00000
	can1	0xFFC001FF
UART Controller	uart0	0xFFC02000
	uart1	0xFFC03000
I2C Controller	i2c0	0xFFC04000
	i2c1	0xFFC05000
	i2c2	0xFFC06000
	i2c3	0xFFC07000
Timer	sptimer0	0xFFC08000
	sptimer1	0xFFC09000
	osctimer0	0xFFD00000
	osctimer1	0xFFD01000
SDRAM Controller	sdr	0xFFC20000
DMA Controller	dmanonsecure	0xFFE00000
	dmasecure	0xFFE01000
SPI Controller	spis0	0xFFE02000
	spis1	0xFFE03000
	spim0	0xFFF00000
	spim1	0xFFF01000
On-Chip Memory	ocram	0xFFFF0000

2.2 External Devices and Peripheral Pin Multiplexing

In addition to built-in devices, the HPS may be connected to external devices through the HPS's peripheral pins. These pins are physical connections that are wired to other devices on the FPGA board. Peripheral pin multiplexers inside the HPS are then configured to route the signals from these pins to various endpoints. These multiplexers' select signals are set by writing to *Pin Mux Control* registers, which are mapped in HPS address space. To determine which peripheral pins have been connected to external devices, consult the board manufacturer's schematics for the board in question.

As an example of using the pin multiplexing, let's examine the peripheral pin connection to the ADXL345 accelerometer chip on the DE1-SoC board. The ADXL345 is operated through its I2C interface, and by consulting the DE1-SoC board's schematics we can see that the accelerometer's I2C wires are connected to the HPS peripheral pins *trace_d6* and *trace_d7*. To determine which registers are responsible for controling these pins' multiplexers, we consult the *Cyclone V HPS Memory Map* document. Figure 1 shows an excerpt of the memory map, which shows that registers *GENERALIO7* and *GENERALIO8* are responsible for pins *trace_d6* and *trace_d7* respectively.

GENERALIO4	0x490	32	RW	0x0	trace_d3 Mux Selection Register
GENERALIO5	0x494	32	RW	0x0	trace_d4 Mux Selection Register
GENERALIO6	0x498	32	RW	0x0	trace_d5 Mux Selection Register
GENERALIO7	0x49C	32	RW	0x0	trace_d6 Mux Selection Register
GENERALIO8	0x4A0	32	RW	0x0	trace_d7 Mux Selection Register
GENERALIO9	0x4A4	32	RW	0x0	spim0_clk Mux Selection Register
GENERALIO10	0x4A8	32	RW	0x0	spim0_mosi Mux Selection Register
GENERALIO11	0x4AC	32	RW	0x0	spim0_miso Mux Selection Register
GENERALIO12	0x4B0	32	RW	0x0	spim0_ss0 Mux Selection Register

Figure 1. Consulting the Cyclone V HPS Memory Map for the list of Pin Mux Control registers.

By clicking on one of the pin multiplexing registers, you can see a list of possible routings that can be made for the corresponding pin. Figure 2 shows the list for the *GENERALIO7* register (*trace_d6* pin). Note that the routing options for *trace_d7* is similar to *trace_d6*.

GENERALIO7 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d6. 0 : Pin is connected to GPIO/LoanIO number 55. 1 : Pin is connected to Peripheral signal I2C0.SDA. 2 : Pin is connected to Peripheral signal SPIS1.SS0. 3 : Pin is connected to Peripheral signal TRACE.D6.	RW	0x0

Figure 2. Consulting the Cyclone V HPS Memory Map for details of the *GENERALIO7* register.

The possible routings for *trace_d6* are described in more detail below:

1. GPIO/LoanIO number 55: A value of 0 routes the pin to the GPIO/LoanIO multiplexer, which in turn can route this signal to either the GPIO controller or to the FPGA fabric as a LoanIO wire.
2. I2C0.SDA: A value of 1 routes the pin to SDA port of the I2C controller *I2C0*.
3. SPIS1.SS0: A value of 2 routes the pin to the SS0 port of the SPI Slave controller *SPIS1*.
4. TRACE.D6: A value of 3 routes the pin to the D6 port of the Trace controller.

Figure 3 provides a high-level view of the ADXL345's signals, and the multiplexers involved in routing them. The typical routing configuration is to connect the ADXL345's I2C signals to the *I2C0* I2C controller. This allows a master to communicate with the ADXL345 chip via *I2C0*'s memory-mapped register interface. This means writing '1' to *GENERALIO7* and *GENERALIO8*, and '0' to *I2C0USEFPGA*.

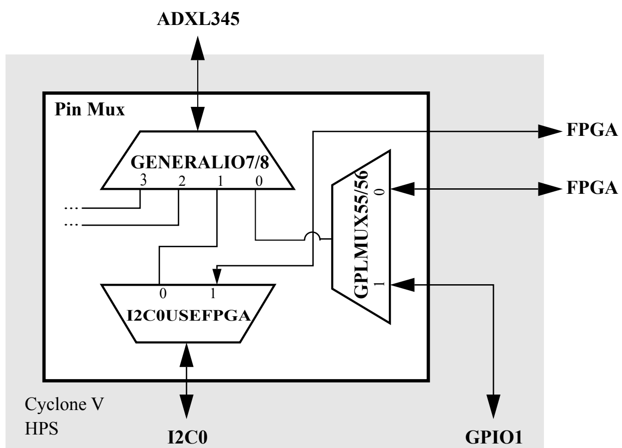

Figure 3. Routing the I2C signals from the accelerometer in a DE1-SoC board.

2.3 Allowing Non-Secure Access to Devices

The HPS interconnect contains a security feature that limits access to various devices so that only "secure" masters, such as the ARM* Cortex* A9 processor, can access them. Because masters in the FPGA are considered non-secure, a secure master must first configure the interconnect to allow non-secure access to a device before FPGA-side masters can access it. This is done by writing to *Security Register Group* registers, which are part of the *L3 GPV Registers*. To allow non-secure access to a device, a '1' must be written to the device's corresponding security bit. For example, to allow non-secure access to *I2C0*'s register interface, you must write a '1' to bit 2 of the *l4sp* register. Further details about the *Security Register Group* registers can be found in the *Cyclone V HPS Memory Map*, as shown in Figure 4.

Security Register Group

Register	Offset	Width	Access	Reset Value	Description
l4main	0x8	32	WO	0x0	L4 main peripherals security
l4sp	0xC	32	WO	0x0	L4 SP Peripherals Security
l4mp	0x10	32	WO	0x0	L4 MP Peripherals Security
l4osc1	0x14	32	WO	0x0	L4 OSC1 Peripherals Security
l4spim	0x18	32	WO	0x0	L4 SPIM Peripherals Security
stm	0x1C	32	WO	0x0	STM Peripheral Security
lwhps2fpgaregs	0x20	32	WO	0x0	LWHPS2FPGA AXI Bridge Registers Peripheral Security
usb1	0x28	32	WO	0x0	USB1 Registers Peripheral Security
nanddata	0x2C	32	WO	0x0	NAND Flash Controller Data Peripheral Security
usb0	0x80	32	WO	0x0	USB0 Registers Peripheral Security
nandregs	0x84	32	WO	0x0	NAND Flash Controller Registers Peripheral Security
qspidata	0x88	32	WO	0x0	QSPI Flash Controller Data Peripheral Security
fpgamgrdata	0x8C	32	WO	0x0	FPGA Manager Data Peripheral Security
hps2fpgaregs	0x90	32	WO	0x0	HPS2FPGA AXI Bridge Registers Peripheral Security
acp	0x94	32	WO	0x0	MPU ACP Peripheral Security
rom	0x98	32	WO	0x0	ROM Peripheral Security
ocram	0x9C	32	WO	0x0	On-chip RAM Peripheral Security
sdrdata	0xA0	32	WO	0x0	SDRAM Data Peripheral Security

Figure 4. The L3 GPV Security Registers, seen in the *Cyclone V HPS Memory Map*.

3 Accessing the HPS Interconnect from the FPGA

3.1 Connecting an FPGA Master to the HPS Interconnect

An AXI or Avalon® bus-mastering device inside the FPGA can be connected to the HPS interconnect through the FPGA-to-HPS bridge. This connection is made in the Platform Designer system integration tool, by connecting the master device's memory mapped master port to the *Hard Processor System* component's AXI_Slave port named *f2h_axi_slave*. Figure 5 shows an example of such connection in the Platform Designer GUI, where the master device is an instantiation of the Nios® II soft processor.

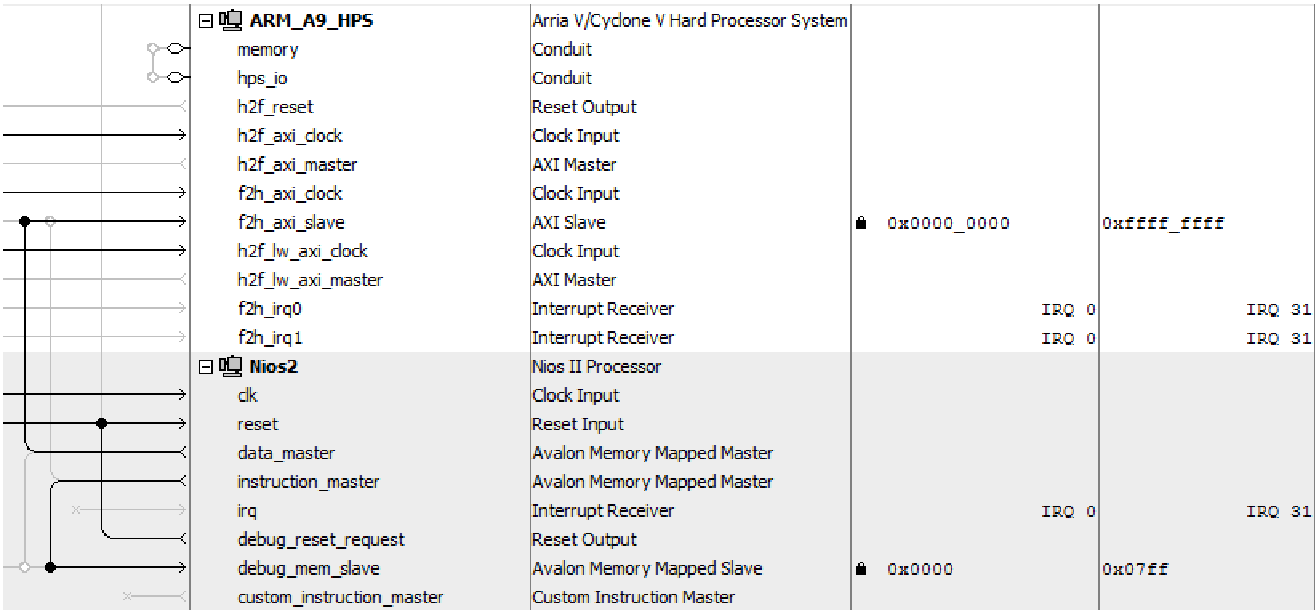

Figure 5. Connecting an FPGA-side master to the HPS interconnect.

3.2 Enabling the FPGA-to-HPS Bridge

Before FPGA-side masters can access the HPS interconnect, the FPGA-to-HPS bridge must first be enabled by deasserting its reset bit in the *brgmodrst*. The *brgmodrst* register is located at address `0xFFD0501C` in HPS address space. Since FPGA-side masters cannot access HPS address space until the bridge is enabled, the resets must be deasserted by a master inside the HPS. This is usually accomplished by running a baremetal program on the ARM Cortex A9 processor to write a 0 to bit 2 of the *brgmodrst* register. After deasserting the bridge's reset, the FPGA-side master has access to the full 4GB address space through the FPGA-to-HPS bridge.

3.3 The Address Span Extender

The HPS interconnect has an address space that spans 4GB, which takes up the entirety of a 32-bit master's address range. This scenario was shown in Figure 5, where the *f2h_axi_slave* connection took up the entire 32-bit (`0x00000000 - 0xffffffff`) address range of the Nios II processor. Such a connection would prevent the master from addressing any other memory-mapped device. As a workaround to this limitation, you can use a standard Platform Designer IP core called the *Address Span Extender*.

The *Address Span Extender* IP core provides a window into the address space of a slave. Figure 6 shows the use of the *Address Span Extender* to provide a 16MB window into the top portion of the HPS interconnect's memory range, from 0xFF000000 to 0xFFFFFFFF. This window provides the Nios II processor access to all of the HPS's built-in devices listed in Section 2.1, and leaves the rest of the address range free for addressing other memory-mapped devices. The size of the window, as well as the window's offset from the base address of the slave can be configured during the instantiation of the core. For further details regarding the *Address Span Extender*, refer to the *Platform Designer System Design Components* section of the *Quartus® Prime Handbook*.

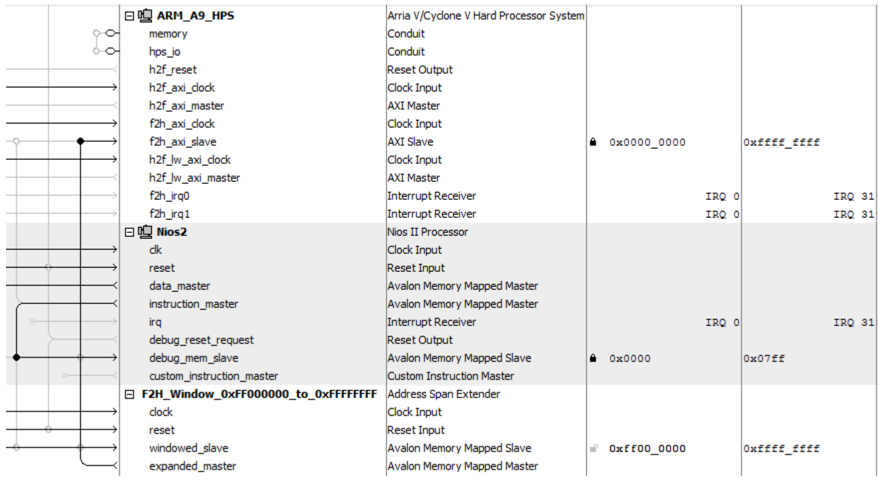

Figure 6. Connecting an FPGA-side master to the HPS interconnect via an address span extender.

4 Accessing HPS Peripheral Pins from the FPGA

This section describes how to connect HPS peripheral pins as input, output, or inout ports to user-defined HDL modules in the FPGA.

4.1 Using the LoanIO Interface in Platform Designer

In Figure 3 you can see that the pin multiplexing can be configured to route the ADXL345 I2C pins to the FPGA side, by setting *GENERALIO7/8* to '0' and *GPLMUX55/56* to '0'. When the multiplexing is configured in such a way, the pins can be accessed through the *LOANIO* port of the Hard Processor System component in Platform Designer. In order to use the *LOANIO* port, you must first configure the HPS component in the *Peripheral Pins* tab of the component wizard. Near the bottom of the tab, you will see a table of peripheral pins, as shown in Figure 7. In the table, you must export the required pins to the *LOANIO* interface by clicking on the *LOANIOXX* button in the corresponding row. Figure 7 shows this being done for the pins *trace_d6* and *trace_d7*.

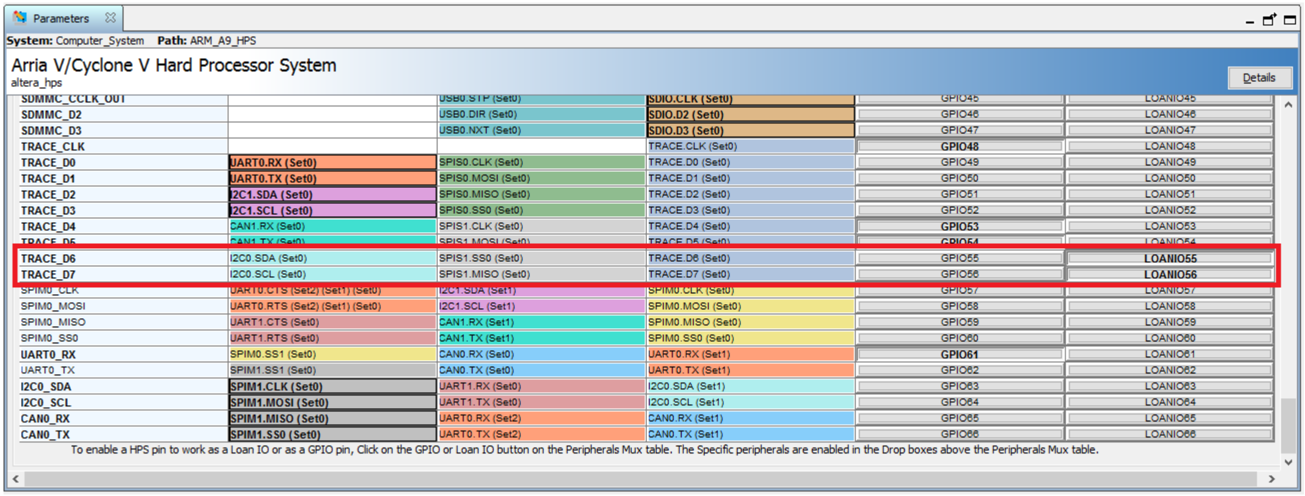

Figure 7. Configuring the HPS Platform Designer Component to connect HPS peripheral pins to the *LOANIO* port.

Once the HPS component is configured, the HPS component will now have a conduit named *h2f_loan_io*, as shown in Figure 8. To access this port in your HDL code, you must export it by double clicking the *Double-click to export* text to the right of the *h2f_loan_io* conduit. This will result in three additional ports in the top-level module generated by Platform Designer, as shown in Figure 9. These ports are as wide as the number of HPS peripheral pins that exist in the chip. In the case of the DE1-SoC board, the ports are 67 bits wide corresponding to the 67 HPS peripheral pins. In Figure 7, you can see that the two pins that we exported are sent to *LOANIO55* and *LOANIO56*, meaning that the two pins can be accessed at indices 55 and 56 in the three ports.

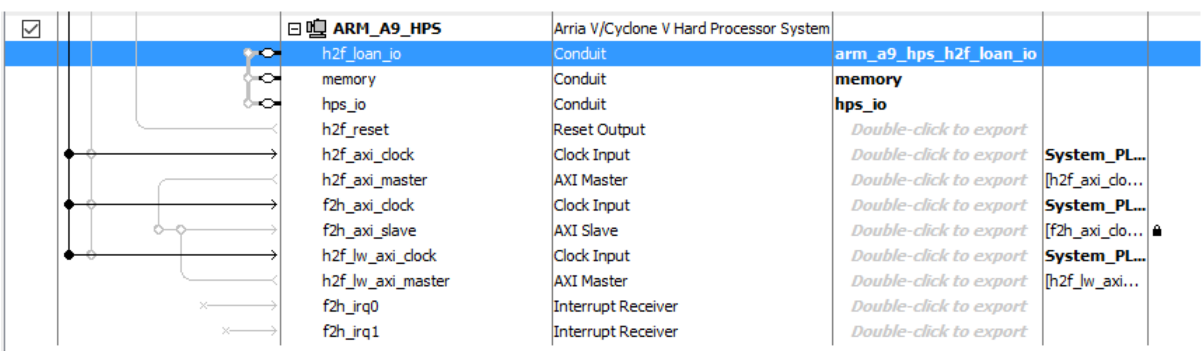

Figure 8. Exporting the HPS component's *LOANIO* port.

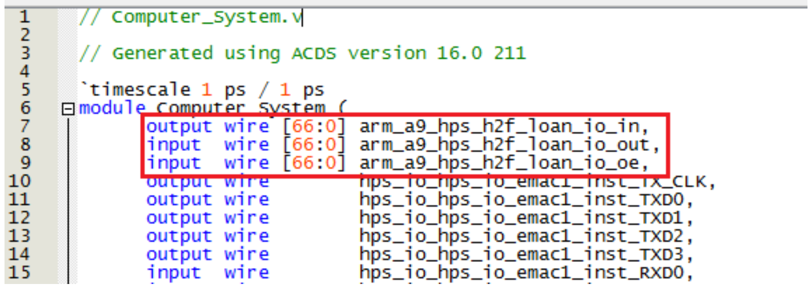

Figure 9. The exported *LOANIO* port.

4.2 Configuring Pin Multiplexing for LoanIO

Section 4.1 described the FPGA-side configuration for using the *LOANIO* port. HPS-side configuration for using the *LOANIO* port is done in a similar way as described in Section 2.2. The goal is to configure the pin multiplexers to route the pins to the *LOANIO* port. First, the *GENERALIO* multiplexer corresponding to the pin must be configured to '0' to route the pin to the *GPIO/LOANIO* interface. Then, the corresponding *GPLMUX* multiplexer must be configured to '0', which routes the pin to the *LOANIO* port.