
SmartFusion2 FPGA

Fabric DDR Controller Configuration



Table of Contents

Introduction	3
1 Fabric External Memory DDR Controller Configurator	4
Memory Settings	4
Fabric Interface Settings	5
2 FDDR Controller Configuration	7
Fabric DDR Control Registers	7
Importing DDR Configuration Files	9
Exporting DDR Configuration Files	10
Firmware	11
Fabric DDR Configuration Path	11
3 Port Description	13
A Product Support	19
Customer Service	19
Customer Technical Support Center	19
Technical Support	19
Website	19
Contacting the Customer Technical Support Center	19
ITAR Technical Support	20

Introduction

The SmartFusion2 FPGA has two embedded DDR controllers - one accessible via the MSS (MDDR) and the other intended for direct access from the FPGA Fabric (FDDR). The MDDR and FDDR both control off-chip DDR memories.

To fully configure the Fabric DDR controller you must:

1. Use the Fabric External Memory DDR Controller Configurator to configure the DDR Controller, select its datapath bus interface (AXI or AHBLite), and select the DDR clock frequency as well as the fabric datapath clock frequency.
2. Set the register values for the DDR controller registers to match your external DDR memory characteristics.
3. Instantiate the Fabric DDR as part of a user application and make datapath connections.
4. Connect the DDR controller's APB configuration interface as defined by the Peripheral Initialization solution.

1 – Fabric External Memory DDR Controller Configurator

The Fabric External Memory DDR (FDDR) Configurator is used to configure the overall datapath and the external DDR memory parameters for the Fabric DDR Controller.

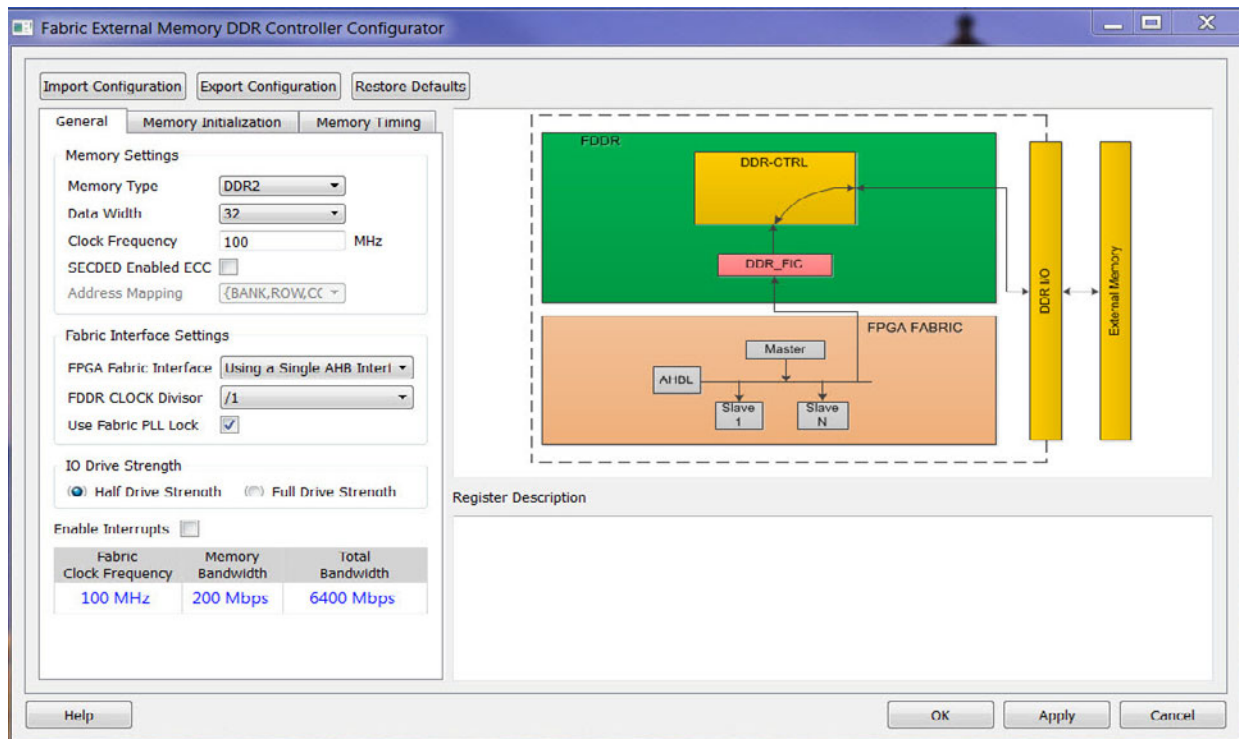


Figure 1-1 • FDDR Configurator Overview

Memory Settings

Use Memory Settings to configure your memory options in the MDDR.

- **Memory Type** - LPDDR, DDR2, or DDR3
- **Data Width** - 32-bit, 16-bit or 8-bit
- **Clock Frequency** - Any value (Decimal/Fractional) in the range of 20 MHz to 333 MHz
- **SECCED Enabled ECC** - ON or OFF
- **Address Mapping** - {ROW,BANK,COLUMN},{BANK,ROW,COLUMN}

Fabric Interface Settings

FPGA Fabric Interface - This is the data interface between the FDDR and the FPGA design. Because the FDDR is a memory controller, it is intended to be a slave on an AXI or AHB bus. The Master of the bus initiates bus transactions, which are in turn interpreted by the FDDR as memory transactions and communicated to the off-chip DDR Memory. FDDR fabric interface options are:

- **Using an AXI-64 Interface** - One master accesses the FDDR through a 64-bit AXI interface.
- **Using a Single AHB-32 Interface** - One master accesses the FDDR through a single 32-bit AHB interface.
- **Using Two AHB-32 Interfaces** - Two masters access the FDDR using two 32-bit AHB interfaces.

FPGA CLOCK Divisor - Specifies the frequency ratio between the DDR Controller clock (CLK_FDDR) and the clock controlling the fabric interface (CLK_FIC64). The CLK_FIC64 frequency should be equal to that of the AHB/AXI subsystem that is connected to the FDDR AHB/AXI bus interface. For example, if you have a DDR RAM running at 200 MHz and your Fabric/AXI Subsystem runs at 100 MHz, you must select a divisor of 2 (Figure 1-2).

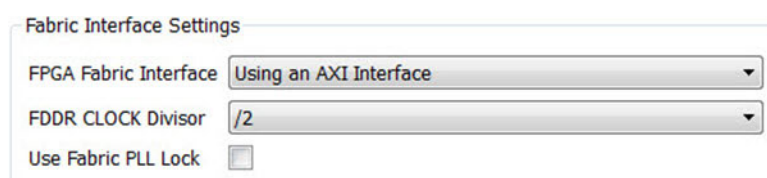


Figure 1-2 • Fabric Interface Settings - AXI Interface and FDDR Clock Divisor Agreement

Use Fabric PLL LOCK - If CLK_BASE is sourced from a Fabric CCC, you can connect the fabric CCC LOCK output to the FDDR FAB_PLL_LOCK input. CLK_BASE is not stable until the Fabric CCC locks. Therefore, Microsemi recommends that you hold the FDDR in reset (i.e., assert the CORE_RESET_N input) until CLK_BASE is stable. The LOCK output of the Fabric CCC indicates that the Fabric CCC output clocks are stable. By checking the Use FAB_PLL_LOCK option, you can expose the FAB_PLL_LOCK input port of the FDDR. You can then connect the LOCK output of the Fabric CCC to the FAB_PLL_LOCK input of the FDDR.

IO Drive Strength

Select one of the following drive strengths for your DDR I/O's:

- Half Drive Strength
- Full Drive Strength

Depending on your DDR Memory type and the I/O Strength you select, Libero SoC sets the DDR I/O Standard for your FDDR system as follows:

DDR Memory Type	Half Drive Strength	Full Drive Strength
DDR3	SSTL15I	SSTL15II
DDR2	SSTL18I	SSTL18II
LPDDR	LPDRI	LPDRII

Enable Interrupts

The FDDR is capable of raising interrupts when certain predefined conditions are satisfied. Check **Enable Interrupts** in the FDDR configurator if you would like to use these interrupts in your application. This exposes the interrupt signals on the FDDR instance. You can connect these interrupt signals as your design requires. The following Interrupt signals and their preconditions are available:

- **FIC_INT** - Generated when there is an error in the transaction between the Master and the FDDR
- **IO_CAL_INT** - Enables you to recalibrate DDR I/O's by writing to DDR controller registers via the APB configuration interface. When calibration is complete, this interrupt is raised. For details about I/O recalibration, refer to the Microsemi [SmartFusion2 Users Guide](#).
- **PLL_LOCK_INT** - Indicates that the FDDR FPLL has locked
- **PLL_LOCKLOST_INT** - Indicates that the FDDR FPLL has lost lock
- **FDDR_ECC_INT** - Indicates a single or two-bit error has been detected

Fabric Clock Frequency

Clock frequency calculation based on your current Clock frequency and CLOCK divisor, displayed in MHz.

$$\text{Fabric Clock Frequency (in MHz)} = \text{Clock Frequency} / \text{CLOCK divisor}$$

Memory Bandwidth

Memory bandwidth calculation based on your current Clock Frequency value in Mbps.

$$\text{Memory Bandwidth (in Mbps)} = 2 * \text{Clock Frequency}$$

Total Bandwidth

Total bandwidth calculation based on your current Clock Frequency, Data Width and CLOCK divisor, in Mbps.

$$\text{Total Bandwidth (in Mbps)} = (2 * \text{Clock Frequency} * \text{Data Width}) / \text{CLOCK Divisor}$$

2 – FDDR Controller Configuration

When you use the Fabric DDR Controller to access an external DDR Memory, the DDR Controller must be configured at runtime. This is done by writing configuration data to dedicated DDR controller configuration registers. This configuration data is dependent on the characteristics of the external DDR memory and your application. This section describes how to enter these configuration parameters in the FDDR controller configurator and how the configuration data is managed as part of the overall Peripheral Initialization solution. Refer to the [Peripheral Initialization User Guide](#) for detailed information about the Peripheral Initialization solution.

Fabric DDR Control Registers

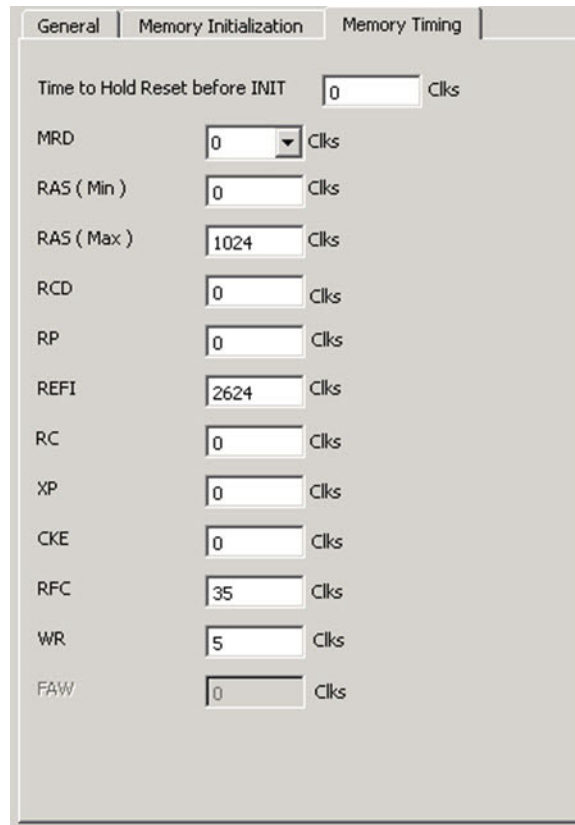
The Fabric DDR Controller has a set of registers that need to be configured at runtime. The configuration values for these registers represent different parameters (for example, DDR mode, PHY width, burst mode, ECC, etc.). For details about the DDR controller configuration registers, refer to the [Microsemi SmartFusion2 User's Guide](#).

Fabric DDR Registers Configuration

Use the Memory Initialization ([Figure 2-1](#)) and Memory Timing ([Figure 2-2](#)) tabs to enter parameters that correspond to your DDR Memory and application. Values you enter in these tabs are automatically translated to the appropriate register values. When you click a specific parameter, its corresponding register is described in the Register Description Window ([Figure 1-1 on page 4](#)).

General	Memory Initialization	Memory Timing
Burst Length	4	Bits
Burst Order	Sequential	
Timing Mode	1T	
CAS Latency	3	Clks
Self Refresh Enabled	NO	
Auto Refresh Burst Count	Single	Bursts
Powerdown Enabled	YES	
Stop the Clock	NO	
Deep Powerdown Enabled	NO	
Powerdown Entry Time	192	
Additive CAS Latency	0	Clks
CAS Write Latency	5	Clks
Zqinit	0	Clks
ZQCS	0	Clks
ZQCS Interval	0	Clks

Figure 2-1 • FDDR Configuration - Memory Initialization Tab



Parameter	Value (Clks)
Time to Hold Reset before INIT	0
MRD	0
RAS (Min)	0
RAS (Max)	1024
RCD	0
RP	0
REFI	2624
RC	0
XP	0
CKE	0
RFC	35
WR	5
FAW	0

Figure 2-2 • FDDR Configuration - Memory Timing Tab

Importing DDR Configuration Files

In addition to entering DDR Memory parameters using the Memory Initialization and Timing tabs, you can import DDR register values from a file. To do so, click the **Import Configuration** button and navigate to the text file containing DDR register names and values. [Figure 2-3](#) shows the import configuration syntax.

```
ddrc_dyn_soft_reset_CR      0x00 ;
ddrc_dyn_refresh_1_CR      0x27DE ;
ddrc_dyn_refresh_2_CR      0x030F ;
ddrc_dyn_powerdown_CR      0x02 ;
ddrc_dyn_debug_CR          0x00 ;
ddrc_ecc_data_mask_CR      0x0000 ;
ddrc_addr_map_col_1_CR      0x3333 ;
ddrc_addr_map_col_3_CR      0x3300 ;
ddrc_init_1_CR              0x0001 ;
ddrc_cke_rstn_cycles_CR1    0x0100 ;
ddrc_cke_rstn_cycles_CR2    0x0008 ;
ddrc_init_emr2_CR           0x0000 ;
ddrc_init_emr3_CR           0x0000 ;
ddrc_dram_bank_act_timing_CR 0x1947;
```

Figure 2-3 • DDR Register Configuration File Syntax

Note: If you choose to import register values rather than entering them using the GUI, you must specify all necessary register values. Refer to the [SmartFusion2 User Guide](#) for details.

Exporting DDR Configuration Files

You can also export the current register configuration data into a text file. This file will contain register values that you imported (if any) as well as those that were computed from GUI parameters you entered in this dialog box.

If you want to undo changes you have made to the DDR register configuration, you can do so with Restore Default. This deletes all register configuration data and you must either re-import or reenter this data. The data is reset to the hardware reset values.

Generated Data

Click **OK** to generate the configuration. Based on your input in the General, Memory Timing and Memory Initialization tabs, the FDDR Configurator computes values for all DDR configuration registers and exports these values into your firmware project and simulation files. The exported file syntax is shown in [Figure 2-4](#).

```
# Exported: 2013-Sep-02 05:07:16
# Libero DDR Configurator GUI Version = 2.0
# DDR Controller Type = DDR2
# Bus Width = 32-bits
# Memory Bandwidth = 200 Mbps
# Total Bandwidth = 6400 Mbps
#
# Validation Status:
# Target Device Manufacturer:
# Target Device:
#
# User Comments:
#
#
DDRC_ADDR_MAP_BANK_CR.REG_DDRC_ADDRMAP_BANK_B2      0xa
DDRC_ADDR_MAP_BANK_CR.REG_DDRC_ADDRMAP_BANK_B1      0xa
DDRC_ADDR_MAP_BANK_CR.REG_DDRC_ADDRMAP_BANK_B0      0xa
DDRC_ADDR_MAP_COL_1_CR.REG_DDRC_ADDRMAP_COL_B7      0x3
DDRC_ADDR_MAP_COL_1_CR.REG_DDRC_ADDRMAP_COL_B4      0x3
DDRC_ADDR_MAP_COL_1_CR.REG_DDRC_ADDRMAP_COL_B3      0x3
DDRC_ADDR_MAP_COL_1_CR.REG_DDRC_ADDRMAP_COL_B2      0x3
DDRC_ADDR_MAP_COL_2_CR.REG_DDRC_ADDRMAP_COL_B11     0xf
DDRC_ADDR_MAP_COL_2_CR.REG_DDRC_ADDRMAP_COL_B10     0xf
DDRC_ADDR_MAP_COL_2_CR.REG_DDRC_ADDRMAP_COL_B9      0xf
DDRC_ADDR_MAP_COL_2_CR.REG_DDRC_ADDRMAP_COL_B8      0x3
DDRC_ADDR_MAP_COL_3_CR.REG_DDRC_ADDRMAP_COL_B6      0x3
DDRC_ADDR_MAP_COL_3_CR.REG_DDRC_ADDRMAP_COL_B5      0x3
```

Figure 2-4 • Exported DDR Register Configuration File Syntax

Firmware

When you generate the SmartDesign, the following files are generated in the <project dir>/firmware/drivers_config/sys_config directory. These files are required for the CMSIS firmware core to compile properly and contain information regarding your current design, including peripheral configuration data and clock configuration information for the MSS. Do not edit these files manually, as they are recreated every time your root design is regenerated.

- sys_config.c
- sys_config.h
- sys_config_mddr_define.h - MDDR configuration data.
- sys_config_fddr_define.h - FDDR configuration data.
- sys_config_mss_clocks.h - MSS clocks configuration

Simulation

When you generate the SmartDesign associated with your MSS, the following simulation files are generated in the <project dir>/simulation directory:

- **test.bfm** - Top-level BFM file that is first executed during any simulation that exercises the SmartFusion2 MSS Cortex-M3 processor. It executes peripheral_init.bfm and user.bfm, in that order.
- **peripheral_init.bfm** - Contains the BFM procedure that emulates the CMSIS::SystemInit() function run on the Cortex-M3 before you enter the main() procedure. It copies the configuration data for any peripheral used in the design to the correct peripheral configuration registers and then waits for all the peripherals to be ready before asserting that the user can use these peripherals.
- **FDDR_init.bfm** - Contains BFM write commands that simulate writes of the Fabric DDR configuration register data you entered (using the Edit Registers dialog box) into the DDR Controller registers.
- **user.bfm** - Intended for user commands. You can simulate the datapath by adding your own BFM commands in this file. Commands in this file will be executed after peripheral_init.bfm has completed.

Using the files above, the configuration path is simulated automatically. You only need to edit the user.bfm file to simulate the datapath. Do not edit the test.bfm, peripheral_init.bfm, or MDDR_init.bfm files as these files are recreated every time your root design is regenerated.

Fabric DDR Configuration Path

The Peripheral Initialization solution requires that, in addition to specifying Fabric DDR configuration register values, you configure the APB configuration data path in the MSS (FIC_2). The SystemInit() function writes the data to the FDDR configuration registers via the FIC_2 APB interface.

Note: If you are using System Builder the configuration path is set and connected automatically.

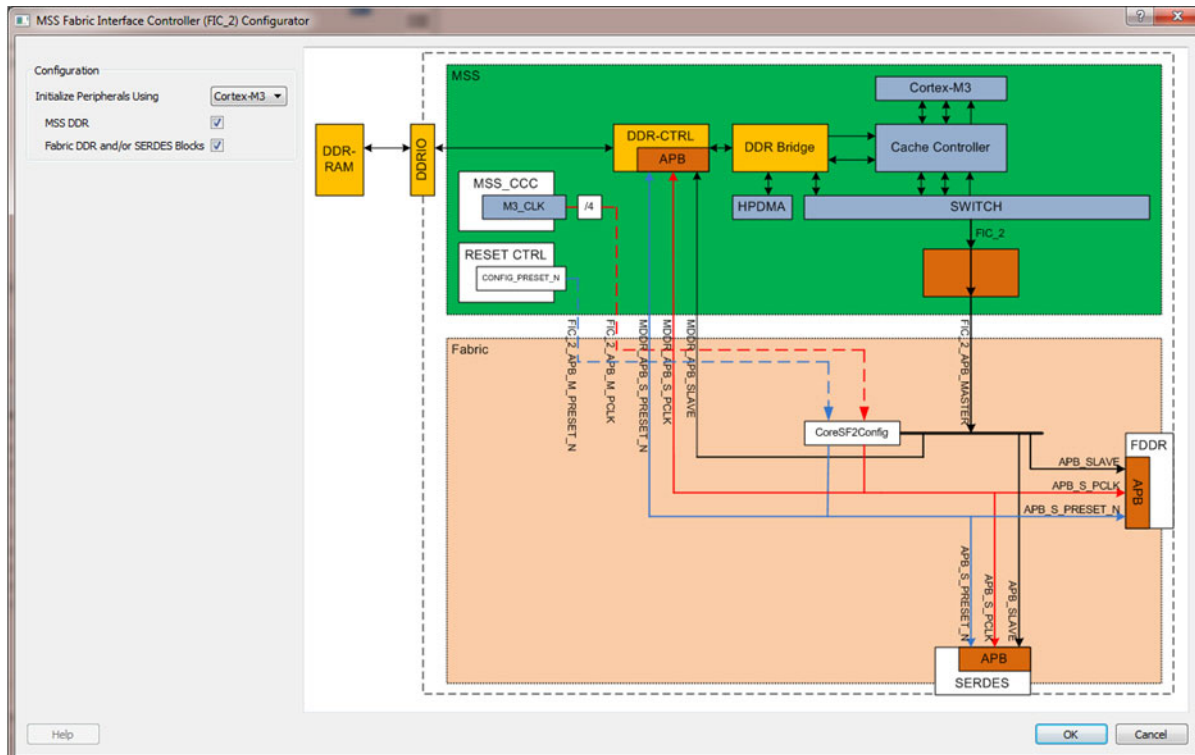


Figure 2-5 • FIC_2 Configurator Overview

To configure the FIC_2 interface:

1. Open the FIC_2 configurator dialog (Figure 2-5) from the MSS configurator.
2. Select the **Initialize peripherals using Cortex-M3** option.
3. Make sure that the MSS DDR is checked, as are the Fabric DDR/SERDES blocks if you are using them.
4. Click **OK** to save your settings. This exposes the FIC_2 configuration ports (Clock, Reset, and APB bus interfaces), as shown in Figure 2-6.
5. Generate the MSS. The FIC_2 ports (FIC_2_APB_MASTER, FIC_2_APB_M_PCLK and FIC_2_APB_M_RESET_N) are now exposed at the MSS interface and can be connected to CoreSF2Config and CoreSF2Reset as per the Peripheral Initialization solution specification.

For details on configuring and connecting the CoreSF2Config and CoreSF2Reset cores, refer to the [Peripheral Initialization Configuration Guide](#).

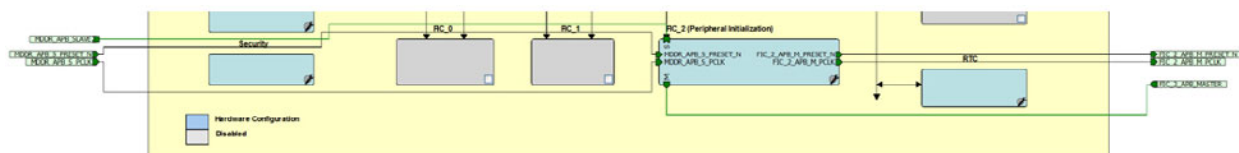


Figure 2-6 • FIC_2 Ports

3 – Port Description

FDDR Core Ports

Table 3-1 • FDDR Core Ports

Port Name	Direction	Description
CORE_RESET_N	IN	FDDR Controller Reset
CLK_BASE	IN	FDDR Fabric Interface Clock
FPLL_LOCK	OUT	FDDR PLL Lock output - high when FDDR PLL is locked
CLK_BASE_PLL_LOCK	IN	Fabric PLL Lock Input. This input is exposed only when the Use FAB_PLL_LOCK option is selected.

Interrupt Ports

This group of ports is exposed when you select the Enable Interrupts option.

Table 3-2 • Interrupt Ports

Port Name	Direction	Description
PLL_LOCK_INT	OUT	Asserts when FDDR PLL locks.
PLL_LOCKLOST_INT	OUT	Asserts when FDDR PLL lock is lost.
ECC_INT	OUT	Asserts when an ECC Event occurs.
IO_CALIB_INT	OUT	Asserts when I/O calibration is complete.
FIC_INT	OUT	Asserts when there is an error in the AHB/AXI protocol on the Fabric interface.

APB3 Configuration Interface

Table 3-3 • APB3 Configuration Interface

Port Name	Direction	Description
APB_S_PENABLE	IN	Slave Enable
APB_S_PSEL	IN	Slave Select
APB_S_PWRITE	IN	Write Enable
APB_S_PADDR[10:2]	IN	Address
APB_S_PWDATA[15:0]	IN	Write Data
APB_S_PREADY	OUT	Slave Ready
APB_S_PSLVERR	OUT	Slave Error
APB_S_PRDATA[15:0]	OUT	Read Data
APB_S_PRESET_N	IN	Slave Reset
APB_S_PCLK	IN	Clock

DDR PHY Interface

Table 3-4 • DDR PHY Interface

Port Name	Direction	Description
FDDR_CAS_N	OUT	DRAM CASN
FDDR_CKE	OUT	DRAM CKE
FDDR_CLK	OUT	Clock, P side
FDDR_CLK_N	OUT	Clock, N side
FDDR_CS_N	OUT	DRAM CSN
FDDR_ODT	OUT	DRAM ODT
FDDR_RAS_N	OUT	DRAM RASN
FDDR_RESET_N	OUT	DRAM Reset for DDR3
FDDR_WE_N	OUT	DRAM WEN
FDDR_ADDR[15:0]	OUT	Dram Address bits
FDDR_BA[2:0]	OUT	Dram Bank Address
FDDR_DM_RDQS[4:0]	INOUT	Dram Data Mask
FDDR_DQS[4:0]	INOUT	Dram Data Strobe Input/Output - P Side
FDDR_DQS_N[4:0]	INOUT	Dram Data Strobe Input/Output - N Side
FDDR_DQ[35:0]	INOUT	DRAM Data Input/Output
FDDR_FIFO_WE_IN[2:0]	IN	FIFO in signal
FDDR_FIFO_WE_OUT[2:0]	OUT	FIFO out signal
FDDR_DM_RDQS ([3:0]/[1:0]/[0])	INOUT	Dram Data Mask
FDDR_DQS ([3:0]/[1:0]/[0])	INOUT	Dram Data Strobe Input/Output - P Side
FDDR_DQS_N ([3:0]/[1:0]/[0])	INOUT	Dram Data Strobe Input/Output - N Side
FDDR_DQ ([31:0]/[15:0]/[7:0])	INOUT	DRAM Data Input/Output
FDDR_DQS_TMATCH_0_IN	IN	FIFO in signal
FDDR_DQS_TMATCH_0_OUT	OUT	FIFO out signal
FDDR_DQS_TMATCH_1_IN	IN	FIFO in signal (32-bit only)
FDDR_DQS_TMATCH_1_OUT	OUT	FIFO out signal (32-bit only)
FDDR_DM_RDQS_ECC	INOUT	Dram ECC Data Mask
FDDR_DQS_ECC	INOUT	Dram ECC Data Strobe Input/Output - P Side
FDDR_DQS_ECC_N	INOUT	Dram ECC Data Strobe Input/Output - N Side
FDDR_DQ_ECC ([3:0]/[1:0]/[0])	INOUT	DRAM ECC Data Input/Output
FDDR_DQS_TMATCH_ECC_IN	IN	ECC FIFO in signal
FDDR_DQS_TMATCH_ECC_OUT	OUT	ECC FIFO out signal (32-bit only)

Note: Port widths for some ports change depending on the selection of the PHY width. The notation "[a:0]/[b:0]/[c:0]" is used to denote such ports, where "[a:0]" refers to the port width when a 32-bit PHY width is selected, "[b:0]" corresponds to a 16-bit PHY width, and "[c:0]" corresponds to an 8-bit PHY width.

AXI Bus Interface

Table 3-5 • AXI Bus Interface

Port Name	Direction	Description
AXI_S_AWREADY	OUT	Write address ready
AXI_S_WREADY	OUT	Write address ready
AXI_S_BID[3:0]	OUT	Response ID
AXI_S_BRESP[1:0]	OUT	Write response
AXI_S_BVALID	OUT	Write response valid
AXI_S_ARREADY	OUT	Read address ready
AXI_S_RID[3:0]	OUT	Read ID Tag
AXI_S_RRESP[1:0]	OUT	Read Response
AXI_S_RDATA[63:0]	OUT	Read data
AXI_S_RLAST	OUT	Read Last - This signal indicates the last transfer in a read burst.
AXI_S_RVALID	OUT	Read address valid
AXI_S_AWID[3:0]	IN	Write Address ID
AXI_S_AWADDR[31:0]	IN	Write address
AXI_S_AWLEN[3:0]	IN	Burst length
AXI_S_AWSIZE[1:0]	IN	Burst size
AXI_S_AWBURST[1:0]	IN	Burst type
AXI_S_AWLOCK[1:0]	IN	Lock type - This signal provides additional information about the atomic characteristics of the transfer.
AXI_S_AWVALID	IN	Write address valid
AXI_S_WID[3:0]	IN	Write Data ID tag
AXI_S_WDATA[63:0]	IN	Write data
AXI_S_WSTRB[7:0]	IN	Write strobes
AXI_S_WLAST	IN	Write last
AXI_S_WVALID	IN	Write valid
AXI_S_BREADY	IN	Write ready
AXI_S_ARID[3:0]	IN	Read Address ID
AXI_S_ARADDR[31:0]	IN	Read address
AXI_S_ARLEN[3:0]	IN	Burst length
AXI_S_ARSIZE[1:0]	IN	Burst size
AXI_S_ARBURST[1:0]	IN	Burst type
AXI_S_ARLOCK[1:0]	IN	Lock Type
AXI_S_ARVALID	IN	Read address valid
AXI_S_RREADY	IN	Read address ready

Table 3-5 • AXI Bus Interface (continued)

Port Name	Direction	Description
AXI_S_CORE_RESET_N	IN	MDDR Global Reset
AXI_S_RMW	IN	Indicates whether all bytes of a 64-bit lane are valid for all beats of an AXI transfer. 0: Indicates that all bytes in all beats are valid in the burst and the controller should default to write commands. 1: Indicates that some bytes are invalid and the controller should default to RMW commands. This is classed as an AXI write address channel sideband signal and is valid with the AWVALID signal. Only used when ECC is enabled.

AHB0 Bus Interface

Table 3-6 • AHB0 Bus Interface

Port Name	Direction	Description
AHB0_S_HREADYOUT	OUT	AHBL slave ready - When high for a write indicates the slave is ready to accept data and when high for a read indicates that data is valid.
AHB0_S_HRESP	OUT	AHBL response status - When driven high at the end of a transaction indicates that the transaction has completed with errors. When driven low at the end of a transaction indicates that the transaction has completed successfully.
AHB0_S_HRDATA[31:0]	OUT	AHBL read data - Read data from the slave to the master
AHB0_S_HSEL	IN	AHBL slave select - When asserted, the slave is the currently selected AHBL slave on the AHB bus.
AHB0_S_HADDR[31:0]	IN	AHBL address - byte address on the AHBL interface
AHB0_S_HBURST[2:0]	IN	AHBL Burst Length
AHB0_S_HSIZE[1:0]	IN	AHBL transfer size - Indicates the size of the current transfer (8/16/32 byte transactions only)
AHB0_S_HTRANS[1:0]	IN	AHBL transfer type - Indicates the transfer type of the current transaction.
AHB0_S_HMASTLOCK	IN	AHBL lock - When asserted the current transfer is part of a locked transaction.
AHB0_S_HWRITE	IN	AHBL write - When high indicates that the current transaction is a write. When low indicates that the current transaction is a read.
AHB0_S_HREADY	IN	AHBL ready - When high, indicates that the slave is ready to accept a new transaction.
AHB0_S_HWDATA[31:0]	IN	AHBL write data - Write data from the master to the slave

AHB1 Bus Interface

Table 3-7 • AHB1 Bus Interface

Port Name	Direction	Description
AHB1_S_HREADYOUT	OUT	AHBL slave ready - When high for a write, indicates the slave is ready to accept data, and when high for a read, indicates that data is valid.
AHB1_S_HRESP	OUT	AHBL response status - When driven high at the end of a transaction indicates that the transaction has completed with errors. When driven low at the end of a transaction, indicates that the transaction has completed successfully.
AHB1_S_HRDATA[31:0]	OUT	AHBL read data - Read data from the slave to the master
AHB1_S_HSEL	IN	AHBL slave select - When asserted, the slave is the currently selected AHBL slave on the AHB bus.
AHB1_S_HADDR[31:0]	IN	AHBL address - byte address on the AHBL interface
AHB1_S_HBURST[2:0]	IN	AHBL Burst Length
AHB1_S_HSIZE[1:0]	IN	AHBL transfer size - Indicates the size of the current transfer (8/16/32 byte transactions only).
AHB1_S_HTRANS[1:0]	IN	AHBL transfer type - Indicates the transfer type of the current transaction.
AHB1_S_HMASTLOCK	IN	AHBL lock - When asserted, the current transfer is part of a locked transaction.
AHB1_S_HWRITE	IN	AHBL write - When high, indicates that the current transaction is a write. When low, indicates that the current transaction is a read.
AHB1_S_HREADY	IN	AHBL ready - When high, indicates that the slave is ready to accept a new transaction.
AHB1_S_HWDATA[31:0]	IN	AHBL write data - Write data from the master to the slave

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060

From the rest of the world, call 650.318.4460

Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to [My Cases](#).

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. [Sales office listings](#) can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within [My Cases](#), select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the [ITAR](#) web page.



Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at www.microsemi.com.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.