RTG4 DDR Memory Controller

Configuration User Guide

Microsemi
a Microchip company

Table of Contents

Introduction

The RTG4 FPGA has two DDR memory controller blocks located on the East and West side of the chip identified as:

- East FDDR
- West FDDR

The DDR controllers control off-chip DDR memories.

To fully configure the RTG4 DDR memory controller you must:

1. Use the RTG4 DDR Memory Controller Configurator to configure the DDR Controller, select its datapath bus interface (AXI or AHB), 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 DDR controller 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.

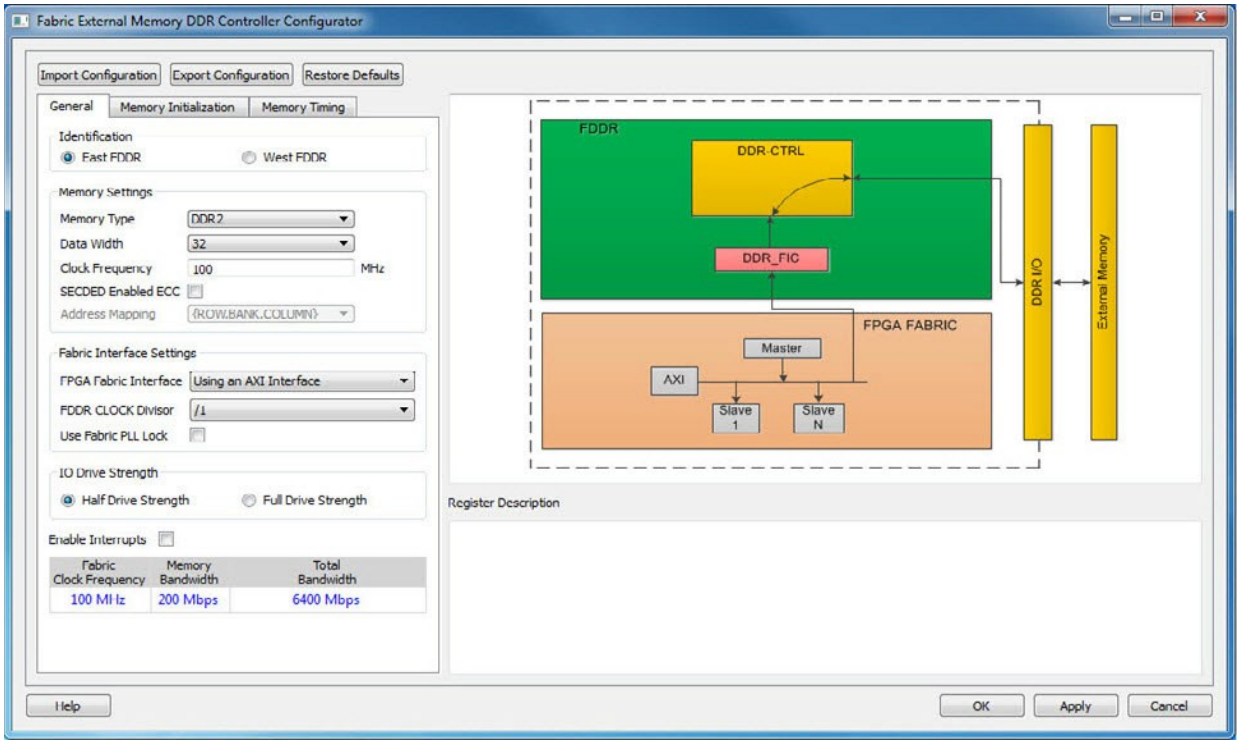

***Figure 1-1* • FDDR Configurator Overview**

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 333MHz
- **SECDED Enabled ECC** - ON or OFF
  Single Error Correction Double Error Detection (SECDED) ECC feature of DDR/LPDDR.

Fabric Interface Settings

FPGA Fabric Interface - This is the data interface between the FDDR and the FPGA design. Since 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 can 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).

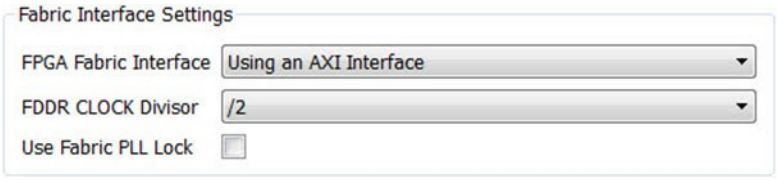

***Figure 1-2* • Fabric Interface Settings - AXI Interface and FDDR Clock Divisor Agreement**

Use Fabric PLL LOCK - In the case where 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 (DDR2 and DDR3 only)

Select one of the following drive strengths for your DDR I/Os:

- 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

IO Standard (LPDDR only)

Select one of the following options:

- LVCMOS18 (Lowest Power) for LVCMOS 1.8V IO standard.
- LPDDRI Note: Before you choose this standard, make sure that your board supports this standard.

IO Calibration

Choose one of the following options:

- On
- Off

Calibration ON and OFF provide different values for PCODE and NCODE registers. The I/O calibration block calibrates the I/O drivers to an external resistor. The impedance control is used to identify the digital values PCODE<5:0> and NCODE<5:0>. These values are fed to the pull-up/pull-down reference network to match the impedance with an external resistor. Once it matches the PCODE and NCODE registers, they are latched and sent to the drivers. Users turn on or turn off this feature as per their board requirements.

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 RTG4 User's 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 (CLK_BASE) calculation based on your current DDR Controller Clock (CLK_FDDR) frequency and the FDDR CLOCK divisor, displayed in MHz.

Fabric Clock (CLK_BASE) Frequency (in MHz) = CLK_FDDR Clock Frequency / FDDR CLOCK divisor

Memory Bandwidth

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

Memory Bandwidth (in Mbps) = 2 * Clock Frequency

Total Bandwidth

Total bandwidth calculation based on your current Fabric Clock Frequency (CLK_BASE), DDR, Data Width and FDDR CLOCK divisor, in Mbps.

Total Bandwidth (in Mbps) = (2 * Fabric Clock Frequency * DDR Data Width) / FDDR 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 to build the initialization circuitry for the FDDR Controller after the FDDR controller is configured.

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 RTG4 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. Consult your DDR Memory vendor's datasheet for values to enter in these two tabs.

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).

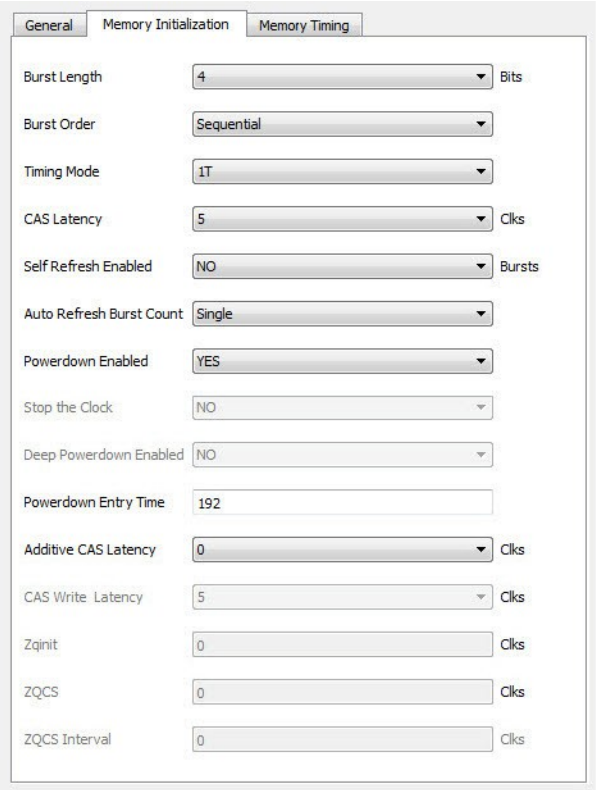

Figure 2-1 **• FDDR Configuration - Memory Initialization Tab**

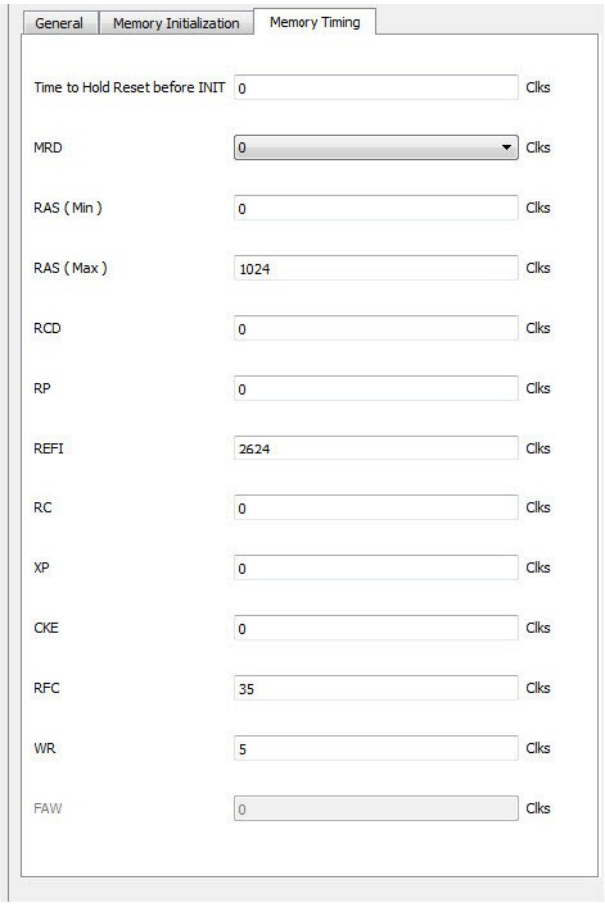

***Figure 2-2* • FDDR Configuration - Memory Timing Tab**

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. Consult 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.

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**

Fabric DDR Initialization

The Fabric DDR Initialization solution requires that, in addition to specifying Fabric DDR configuration register values, you need to build the configuration and initialization circuitry in SmartDesign for your Fabric DDR Controller. The following section describes how to build the initialization circuitry in a SmartDesign component 'FDDR_INIT'.

The FDDR_INIT component consists of

- CoreABC soft IP core
- CoreAPB3 bus Soft IP Core

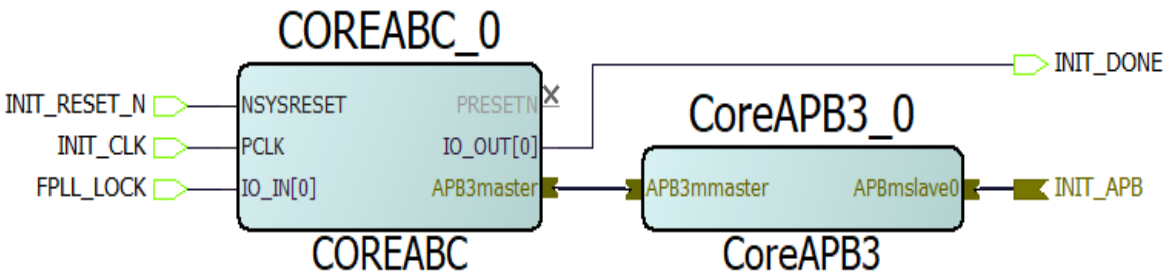

Figure 2-5 **• FDDR_INIT SmartDesign Component**

Configuring and generating the FDDR component

1. From the Catalog, right-click RTG4 DDR Memory Controller and choose **Configure Core**.
2. Click **OK** to exit the Configurator when done. Your FDDR component is generated. Libero generates the CoreABC program in the *<project_location>/component/work/<FDDR_component_name>/<FDDR_component_name>_0/fddr_init_abc.txt* file.
3. Mentioned below are some of the various steps that the CoreABC program does:
   – Perform the FPLL calibration/workaround, which includes polling for the FPLL_LOCK output from FDDR
   – Configure the DDR registers
   – Wait for controller ready and memory settling time, and then assert INIT_DONE

Building the FDDR Initialization Circuitry with FDDR_INIT Component

After you have configured the RTG4 DDR Memory Controller, you need to build the initialization circuitry for the FDDR.

1. Create a SmartDesign component with name FDDR_INIT.
2. From the Catalog, drag and drop CoreABC (v3.6.100 or later) to the SmartDesign component FDDR_INIT.
3. Specify a name to the CoreABC component and click OK to open its configurator.
4. Configure the CoreABC component as shown in Figure 2-6 below. Make sure it has the following selections in the 'Parameters' tab:
   – The data bus width is 32.
   – The maximum number of instructions is at least 256.
   – Number of I/O Inputs is 1.
   – Number of I/O Flags is 1.
   – Number of I/O Outputs is 1.
   – Instruction Store is Hard (FPGA Logic Elements).
   – Use AND and IOWRT operations as optional instructions.
5. Copy the CoreABC program generated for your FDDR from the *fddr_init_abc.txt* file created under the *<project_location>/component/work/../ <user_FDDR_name>_0/* folder and paste to the CoreABC Program tab. See Figure 2-7. The program code loads the DDR Controller Registers with the values you have configured for your DDR Controller and starts the initialization sequence.
   Depending on the number of instructions generated in the *fddr_init_abc.txt* file, you may have to increase the maximum number of Instructions.
6. Click **OK** to exit the configurator when done and generate the CoreABC component.
7. From the Catalog, drag and drop CoreAPB3 (v4.1.100 or later) to the SmartDesign component FDDR_INIT.
8. Specify a name to the CoreAPB3 component and click OK to open its configurator.
9. Configure the CoreAPB3 component as shown in Figure 2-8 below. Make sure it has the following selections:
   – APB Master Data Bus Width is 32.
   – Number of address bits driven by master is 20.
   – Enabled APB Slave Slots is Slot 0. All other Save Slots are disabled.
10. Click **OK** to exit the configurator when done and generate the CoreAB3 component.
11. Make the necessary port tie-offs, promotions and connections between the CoreABC and CoreAPB3 components in the FDDR_INIT SmartDesign canvas as follows. Refer to Figure 2-5 above to see the connections and port tie-offs.
12. When completed, click **Generate** button in the SmartDesign canvas to generate FDDR_INIT component.

Table 2-1 • Port tie-offs, promotions and connections in FDDR_INIT

Port/Bus Interface (BIF) Name/Block	Action
NSYSRESET/CoreABC	Promote to top and rename to INIT_RESET_N
PCLK/CoreABC	Promote to top and rename to INIT_CLK
IO_IN[0]/CoreABC	Promote IO_IN[0] to top and rename to FPLL_LOCK. This will be used to poll for FPLL_LOCK output of the FDDR block.
PRESETN/CoreABC	Mark as unused
IO_OUT[0]/CoreABC	Promote to top and rename to INIT_DONE
APB3master BIF/CoreABC	Connect to APB3mmaster BIF of CoreAPB3
APBmslave0/CoreAPB3	Promote to top and rename to INIT_APB

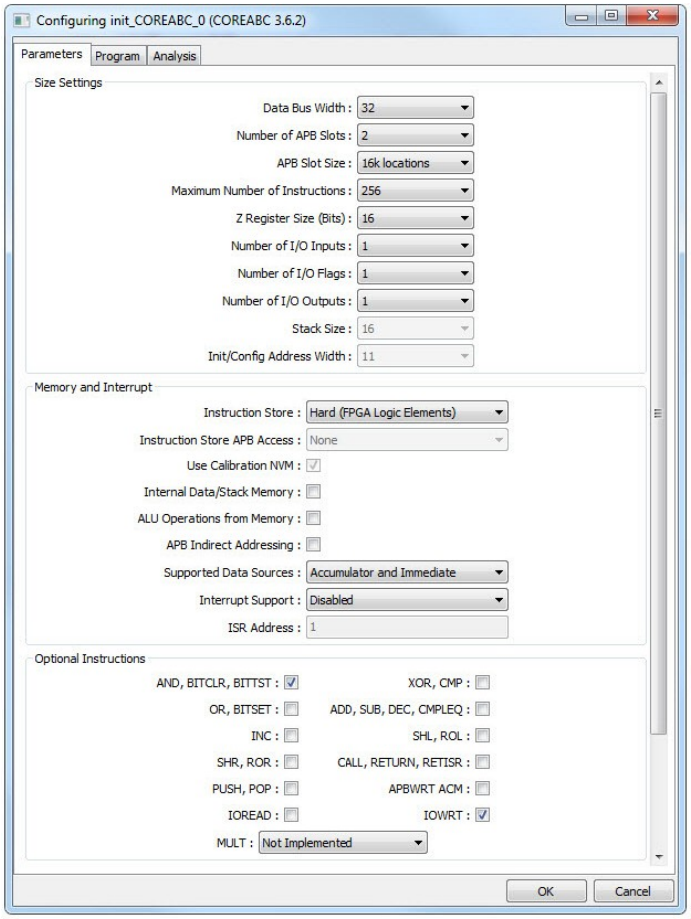

***Figure 2-6* • CoreABC Configurator**

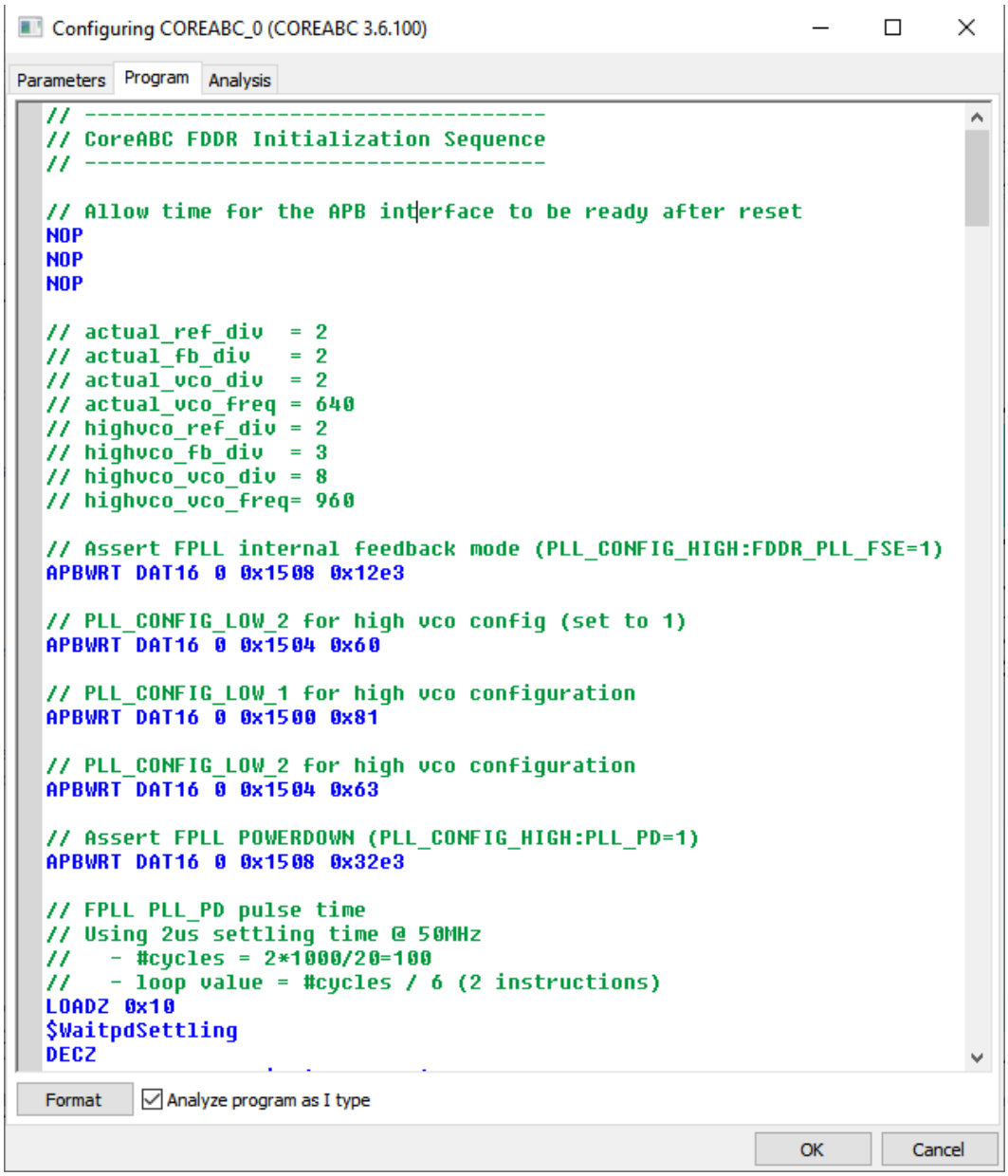

***Figure 2-7* • CoreABC Program Code for FDDR**

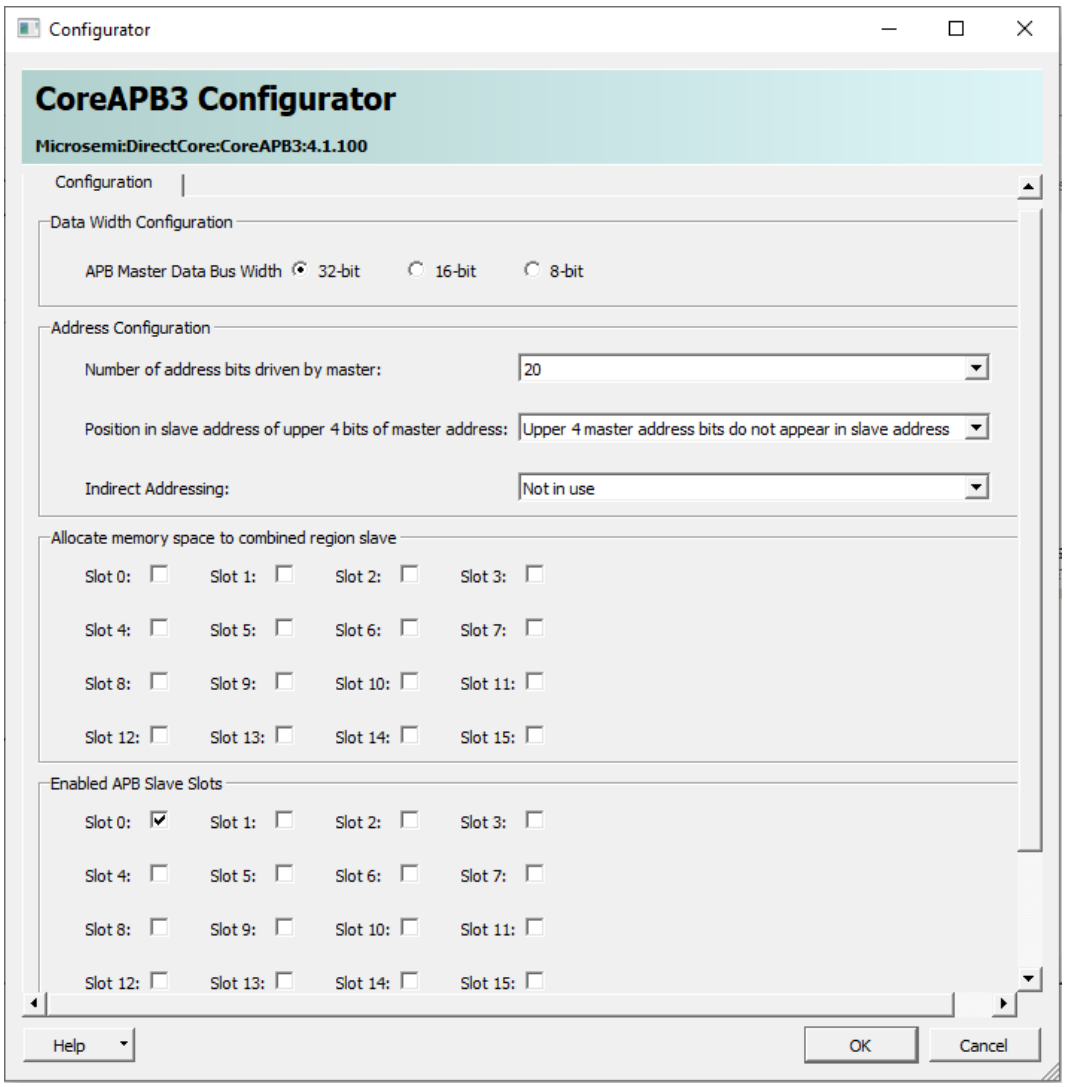

***Figure 2-8* • CoreAPB3 Configuration**

DDR Memory Settling Time

The RTG4 DDR memory controller block is hard-coded with a DDR memory settling time of 200 us, assuming the clock period of INIT_CLK is 20 ns (frequency 50 MHz). Microsemi recommends that the initialization frequency be kept at 50 MHz.

Figure 2-9 shows the DDR Memory Settling Time at 200 us.

```
// Memory Settling time
// Using 200us settling time @ 50MHz
//    - #cycles = 200*1000/20=10000
//    - loop value = #cycles / 6 (2 instructions)
//    - loop value = 10000/6 = 0x682
LOADZ 0x682
$WaitSettling
```

***Figure 2-9* • DDR Memory Settling Time - 200 us**

Consult your DDR Memory vendor's datasheet for the correct memory settling time to use. An incorrect memory settling time may result in the failure of the DDR memory to initialize during operation.

If a different memory settling time is required for your DDR memory or you choose to use a different INIT_CLK frequency than the recommended 50 MHz, you must edit the program code in the Program tab of CoreABC to change the load value of the register used to compute the settling time. Figure 2-10 shows an example of the modified code to support DDR Memory settling time of 400 us.

```
// Memory Settling time
// Using 400us settling time @ 50MHz
//    - #cycles = 400*1000/20=20000
//    - loop value = #cycles / 6 (2 instructions)
//    - loop value = 20000/6 = 0xd05
LOADZ 0xd05
$WaitSettling
```

Figure 2-10 • **DDR Memory Settling Time - 400 us**

Interfacing FDDR with the Initialization Logic

1. Drag and drop the generated FDDR component from the Design Hierarchy window into the top level SmartDesign Canvas.
2. Drag and drop the SmartDesign component FDDR_INIT from the Design Hierarchy window into the same SmartDesign canvas where you have instantiated the FDDR core.

To interface the FDDR to the initialization logic block FDDR_INIT, make the necessary interconnections as shown in the following table.

Table 2-2 • **FDDR to FDDR_INIT Interconnections**

From Port/BIF name/Block Name	To Port/BIF Name/Block Name
APB_SLAVE/FDDR	INIT_APB/FDDR_INIT
APB_S_PCLK/FDDR	INIT_CLK/FDDR_INIT
APB_S_PRESET_N/FDDR	INIT_RESET_N/FDDR_INIT
FPLL_LOCK/FDDR	FPLL_LOCK/FDDR_INIT
AXI_SLAVE/FDDR	Slave (mirrored) BIF of AXI Bus when the FDDR is configured as an AXI Slave
Connected to user Fabric Logic	INIT_DONE

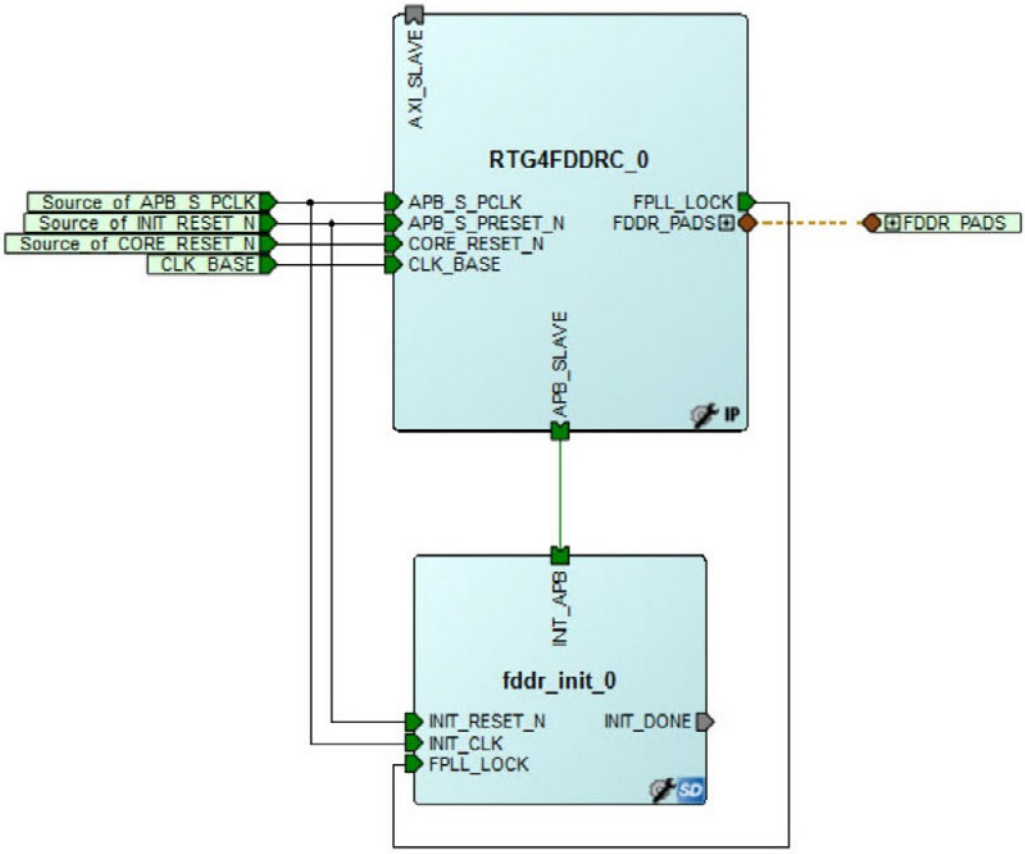

***Figure 2-11* • FDDR Subsystem Initialization Circuitry**

When completed, click the **Generate** button in SmartDesign to generate the FDDR subsystem. Configuration and initialization of your FDDR subsystem is complete.

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
AXI_S_CORE_RESET_N	IN	FDDR Global Reset

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 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and security, aerospace, and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs, and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

5-02-00589-2/04.20