Timing Optimization for AXI3 DDR Interfaces Using SmartFusion2/IGLOO2 - Libero SoC v11.7

Table of Contents

Purpose

This application note describes the optimization techniques used for meeting timing closure on SmartFusion®2 and IGLOO®2 designs that use non-1:1 double data rate (DDR) to AXI clock ratios (2:1, 3:1, and 4:1). It provides reference designs for SmartFusion2 Advanced Development Kit board and IGLOO2 Evaluation Kit board.

Introduction

SmartFusion2 and IGLOO2 devices have two high-speed application-specific integrated circuit (ASIC) memory controllers, that is, microcontroller or memory subsystem (MSS) DDR (MDDR) and fabric DDR (FDDR). Microcontroller subsystem DDR present in SmartFusion2 devices, Memory Subsystem DDR in IGLOO2 devices and FDDR in both SmartFusion2 and IGLOO2 devices are used for interfacing with external memories DDR2, DDR3, and low power DDR1 (LPDDR1) SDRAM memories. The MDDR and FDDR subsystems are used to access high-speed DDR memories for high-speed data transfer and code execution.

The DDR memory connected to the MDDR subsystem can be accessed by the MSS master in SmartFusion2 devices and by High-Performance Memory Subsystem (HPMS) master in IGLOO2

devices. Another way to access the DDR memory in both SmartFusion2 and IGLOO2 devices is by using any master logic implemented in the FPGA fabric master. The DDR memory connected to the FDDR subsystem can only be accessed by an FPGA fabric master.

FPGA fabric master communicates with the MDDR and FDDR subsystems through advanced extensible interface (AXI) or advanced high-performance bus (AHB) interfaces. MDDR or FDDR subsystems operated in AXI mode provide the highest throughput interface to the external memory device.

When the MDDR or FDDR FIC64 interface is configured in AXI mode and is operating at a ratio of 2:1 or higher, depending on the design, timing violations may occur intermittently. For cases where design timing is not met, Microsemi recommends implementing optimization techniques explained in this application note to achieve timing closure. Timing closure optimization methods discussed in this application note apply only to the FDDR and MDDR FIC64 interfaces configured in AXI mode and running with a ratio of 2:1 or higher. They do not apply to DDR memory in AHB or AXI mode running with a 1:1 ratio.

The following figure shows the MDDR data path for AXI/AHB interfaces.

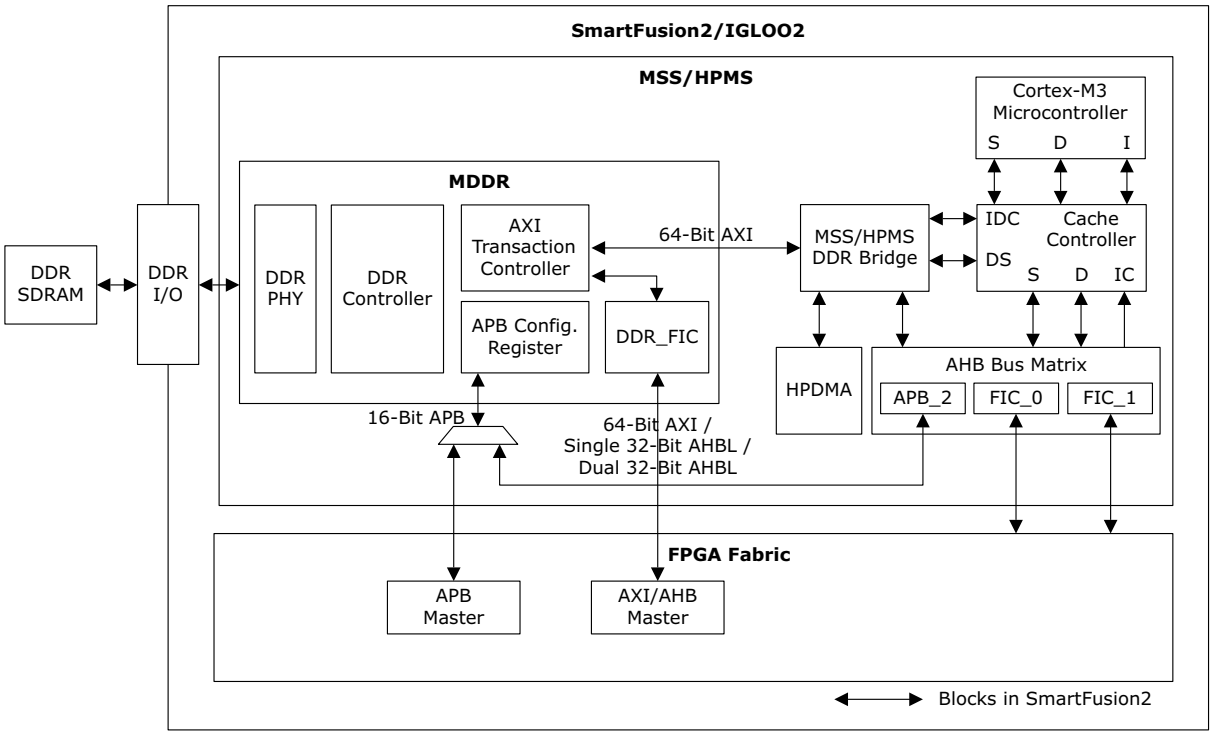

***Figure 1* • MDDR Data Path for AXI/AHB Interfaces**

References

For more information on the reference design for the SmartFusion2 Advanced Development Kit board and IGLOO2 Evaluation Kit board, refer to:

- *DG0568: Interfacing SmartFusion2 SoC FPGA with External LPDDR Memory through MDDR Controller Demo Guide*
- *UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide*
- *DG0534: Interfacing IGLOO2 FPGA with External LPDDR Memory through MDDR Controller*
- *AC424: IGLOO2 - Optimizing DDR Controller for Improved Efficiency Application Note*
- *AC409: Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks in the SmartFusion2 Devices*
- *UG0557: SmartFusion2 SoC FPGA Advanced Development Kit User Guide*

Design Requirements

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Advanced Development Kit: • 12 V adapter (included with the kit) • FlashPro4 programmer (included with the kit)	Rev B or later
IGLOO2 Evaluation Kit: • FlashPro4 programmer • 12 V adapter • USB A to mini-B cable	Rev C or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero® System-on-Chip (SoC)	v11.7
SoftConsole	v3.4 SP1
Host PC drivers	USB to UART drivers

Timing Optimization Techniques

This section describes timing optimization techniques for the following DDR to AXI clock ratios:

- 2:1 Ratio
- 3:1 Ratio
- 4:1 Ratio

2:1 Ratio

When AXI mode is used with the MDDR or FDDR subsystem operating at 2:1 DDR to AXI clock ratio, timing closure can be achieved by inserting a flip-flop on the AWVALID, ARVALID, and WVALID signal paths. A two-input AND gate with inputs from the fabric AXI master VALID signals and the FDDR/MDDR READY signals is fed to the flip-flop. This technique uses a negative edge-triggered flip-flop clocked using the AXI clock (DDR_FIC_SUBSYSTEM_CLK) on the VALID signal paths.

The optimization method can reside between an existing AXI master and the DDR fabric interface control (DDR_FIC) AXI slave interface, and no changes are required to the AXI master design. As the AXI VALID signals are delayed by half AXI clock cycle, AXI data lines going into the DDR_FIC get an additional half AXI clock cycle time to become stable before the active edge of the latching clock.

Figure 2 shows the block diagram of the optimization technique for the 2:1 DDR to AXI clock ratio.

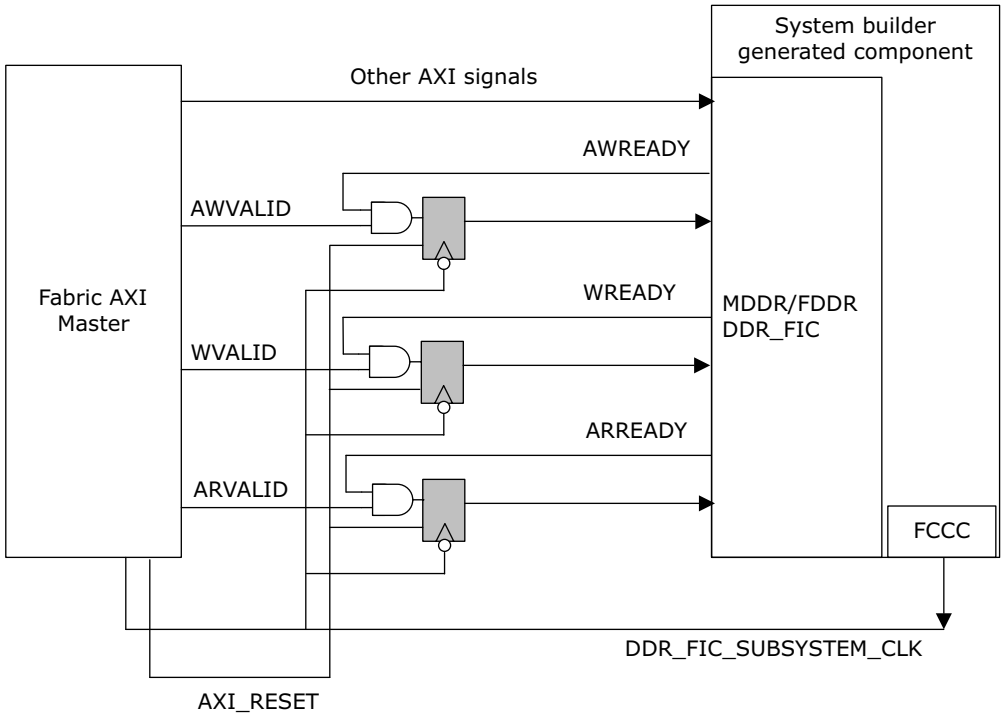

***Figure 2* • AXI Timing Optimization Logic for 2:1 Ratio**

Figure 3 shows the AXI transaction timing diagram with the optimization logic for the 2:1 ratio. The AXI data signals must meet two DDR clock or one AXI clock cycle setup time.

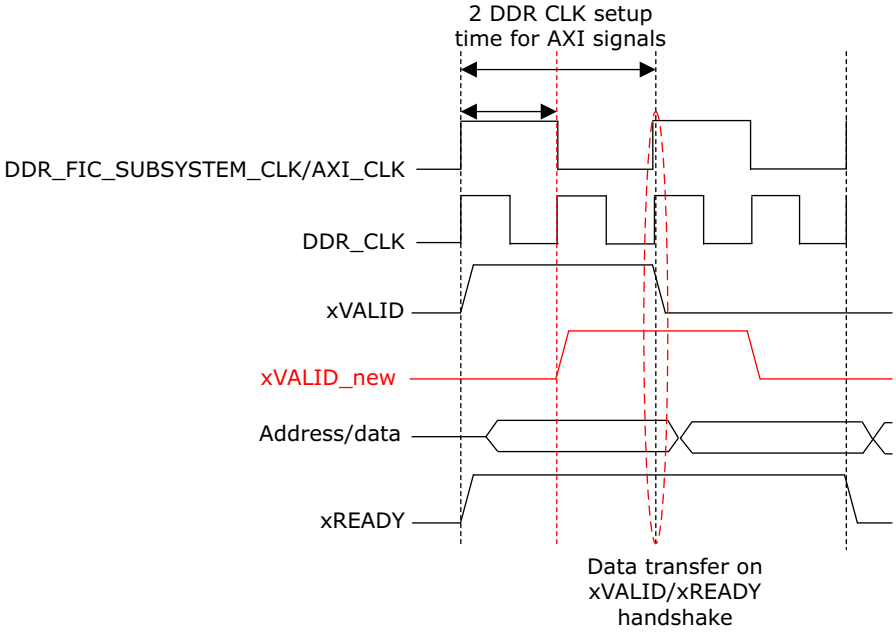

***Figure 3* • Timing Diagram for 2:1 Ratio**

When implementing the 2:1 ratio timing optimization technique, the following SDC constraints need to be added to the timing constraint file (.sdc), which is provided as part of the design files. See "Appendix A: Design files" on page 25 for more information.

For FDDR

```
# The following constraints provide a relaxation constraint on the signals of 1.5 AXI clock period. The users must adjust the ddr_clock_frequency to match their application.
# Apply new max delay for 2:1 clock ratio (new valid paths need to meet one DDR clock cycle setup time and other paths need to meet one AXI clock setup time)
# DDR_AXI FF setup time (Libero SoC 11.7) = DDR_AXI FF setup time (Libero SoC 11.5) + (n - 1) * DDR clock period == DDR_AXI FF setup time (Libero SoC 11.5) + 1* DDR clock period
set ddr_clock_frequency 333
set delay1 [ expr 3000/$ddr_clock_frequency ]
set delay2 [ expr 2000/$ddr_clock_frequency ]
set_max_delay $delay1 -to [ get_pins { \
*/INST_FDDR_IP:F_ARADDR* */INST_FDDR_IP:F_ARBURST* */INST_FDDR_IP:F_ARID* */INST_FDDR_IP:F_ARLEN*\
*/INST_FDDR_IP:F_ARLOCK* */INST_FDDR_IP:F_ARSIZE* */INST_FDDR_IP:F_AWADDR* */INST_FDDR_IP:F_AWBURST* \
*/INST_FDDR_IP:F_AWID* */INST_FDDR_IP:F_AWLEN* */INST_FDDR_IP:F_AWLOCK* */INST_FDDR_IP:F_AWSIZE* \
*/INST_FDDR_IP:F_WDATA* */INST_FDDR_IP:F_WID* */INST_FDDR_IP:F_WLAST */INST_FDDR_IP:F_WSTRB* \
*/INST_FDDR_IP:F_BREADY */INST_FDDR_IP:F_RMW_AXI */INST_FDDR_IP:F_RREADY\
} ]

/* The following constraints provide a relaxation constraint on the signals of 1 AXI clock period. */

set_max_delay $delay2 -to [ get_pins { \
*/INST_FDDR_IP:F_ARVALID* \
*/INST_FDDR_IP:F_AWVALID* \
*/INST_FDDR_IP:F_WVALID \
} ]
```

For MDDR

```
# The following constraints provide a relaxation constraint on the signals of 1.5 AXI clock periods. The users must adjust the ddr_clock_frequency to match their application.
# Apply new max delay for 2:1 clock ratio (new valid paths need to meet one DDR clock cycle setup time and other paths need to meet one AXI clock setup time)
# DDR_AXI FF setup time (Libero SoC 11.7) = DDR_AXI FF setup time (Libero SoC 11.5) + (n - 1) * DDR clock period == DDR_AXI FF setup time (Libero SoC 11.5) + 1* DDR clock period
set ddr_clock_frequency 333
set delay1 [ expr 3000/$ddr_clock_frequency ]
set delay2 [ expr 2000/$ddr_clock_frequency ]
set_max_delay $delay1 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARADDR* */INST_MSS_*_IP:F_ARBURST* */INST_MSS_*_IP:F_ARID* */INST_MSS_*_IP:F_ARLEN*\
*/INST_MSS_*_IP:F_ARLOCK* */INST_MSS_*_IP:F_ARSIZE* */INST_MSS_*_IP:F_AWADDR* */INST_MSS_*_IP:F_AWBURST* \
```

```
*/INST_MSS_*_IP:F_AWID*        */INST_MSS_*_IP:F_AWLEN*        */INST_MSS_*_IP:F_AWLOCK*
*/INST_MSS_*_IP:F_AWSIZE* \
*/INST_MSS_*_IP:F_WDATA*          */INST_MSS_*_IP:F_WID*          */INST_MSS_*_IP:F_WLAST
*/INST_MSS_*_IP:F_WSTRB* \
*/INST_MSS_*_IP:F_BREADY */INST_MSS_*_IP:F_RMW_AXI */INST_MSS_*_IP:F_RREADY\
} ]

/* The following constraints provide a relaxation constraint on the signals of 1 AXI clock period. */

set_max_delay $delay2 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARVALID* \
*/INST_MSS_*_IP:F_AWVALID* \
*/INST_MSS_*_IP:F_WVALID \
} ]
```

3:1 Ratio

When AXI mode is used with the MDDR or FDDR subsystem operating at 3:1 DDR to AXI clock ratio, timing closure can be achieved by inserting two flip-flops on the AWVALID, ARVALID, and WVALID signal paths. A two-input AND gate with inputs from the fabric AXI master VALID signals and the FDDR/MDDR READY signals is fed to the two-stage pipeline flip-flops.

If the design uses a DDR to AXI clock ratio greater than 2:1, increasing pipeline stages helps increase the timing margin without changing the clock frequency. For the 3:1 DDR to AXI clock ratio, two positive edge-triggered flip-flops are used in pipeline. These flip-flops are clocked using DDR clock (DDR_FIC_SUBSYSTEM_CLK*3) on the VALID signal paths. The DDR clock is derived using user PLL (Fabric CCC) as shown in Figure 4 on page 7.

The optimization method can reside between an existing AXI master and the DDR_FIC AXI slave interface and no changes are required to the AXI master design. As the AXI VALID signals are delayed by two DDR clock cycles AXI data lines going into the DDR_FIC get an additional two DDR clock cycle time period to become stable before the active edge of the latching clock.

Figure 4 shows the block diagram of the technique for 3:1 clock ratio.

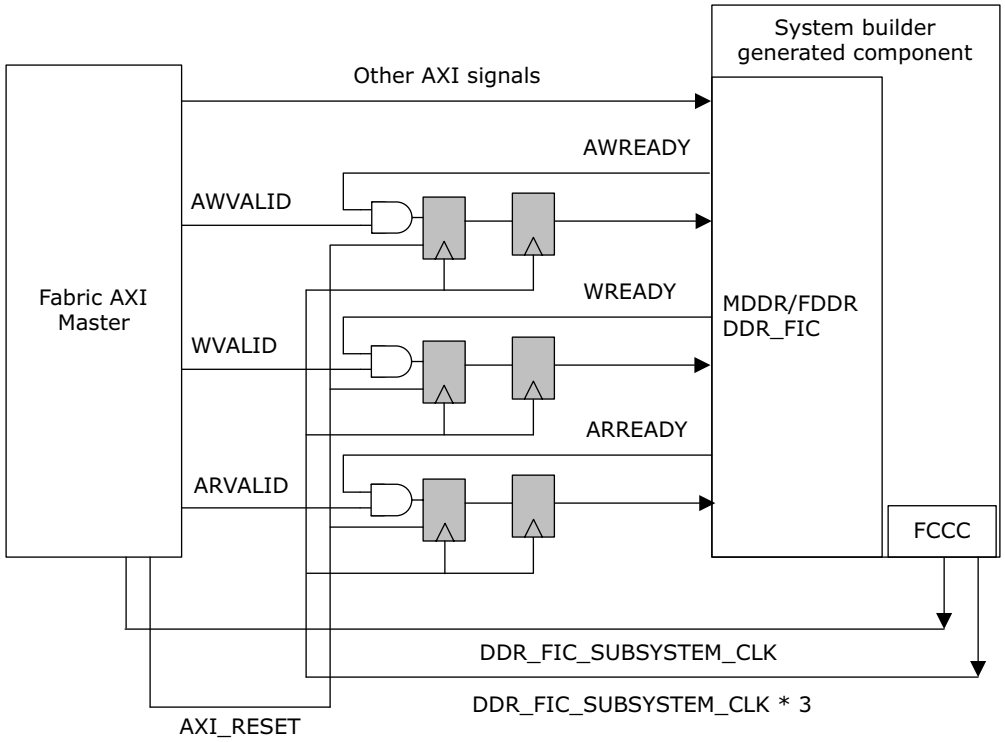

***Figure 4* • AXI Timing Optimization Logic for 3:1 Ratio**

The following figure shows the AXI transaction timing diagram with the optimization logic for 3:1 ratio. The AXI data signals must meet three DDR clock or one AXI clock cycle setup time.

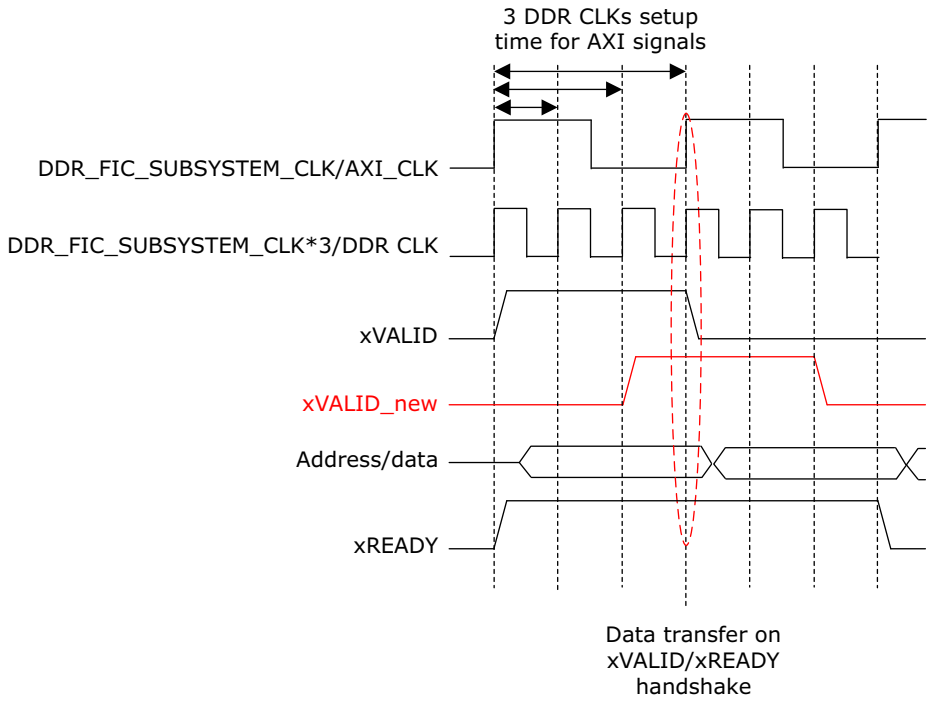

Figure 5 • **Timing Diagram for 3:1 Ratio**

When implementing the 3:1 ratio timing optimization technique, the following SDC constraints need to be added to the timing constraint file (.sdc), which is provided as part of the design files. See "Appendix A: Design files" on page 25 for more information.

For FDDR

```
# The users must adjust the ddr_clock_frequency to match their application.
set ddr_clock_frequency 333
# Apply new max delay for a 3:1 clock ratio (new valid paths need to meet one DDR clock cycle setup time and other paths need to meet one AXI clock setup time)
# DDR_AXI FF setup time (Libero SoC 11.7) = DDR_AXI FF setup time (Libero SoC 11.5) + (n - 1) * DDR clock period == DDR_AXI FF setup time (Libero SoC 11.5) + 2* DDR clock period
set delay1 [ expr 5000/$ddr_clock_frequency ]
set delay2 [ expr 3000/$ddr_clock_frequency ]
set_max_delay $delay1 -to [ get_pins { \
*/INST_FDDR_IP:F_ARADDR*  */INST_FDDR_IP:F_ARBURST*  */INST_FDDR_IP:F_ARID* */INST_FDDR_IP:F_ARLEN*\
*/INST_FDDR_IP:F_ARLOCK*  */INST_FDDR_IP:F_ARSIZE*  */INST_FDDR_IP:F_AWADDR* */INST_FDDR_IP:F_AWBURST* \
*/INST_FDDR_IP:F_AWID*  */INST_FDDR_IP:F_AWLEN*  */INST_FDDR_IP:F_AWLOCK* */INST_FDDR_IP:F_AWSIZE* \
*/INST_FDDR_IP:F_WDATA*  */INST_FDDR_IP:F_WID*  */INST_FDDR_IP:F_WLAST */INST_FDDR_IP:F_WSTRB* \
*/INST_FDDR_IP:F_BREADY */INST_FDDR_IP:F_RMW_AXI */INST_FDDR_IP:F_RREADY \
} ]
```

```
set_max_delay $delay2 -to [ get_pins { \
*/INST_FDDR_IP:F_ARVALID* \
*/INST_FDDR_IP:F_AWVALID* \
*/INST_FDDR_IP:F_WVALID \
} ]
```

For MDDR

```
# The users must adjust the ddr_clock_frequency to match their application.
set ddr_clock_frequency 333
# Apply new max delay for a 3:1 clock ratio (new valid paths need to meet one DDR clock cycle setup time and other paths need to meet 1 AXI clock cycle setup time)
# DDR_AXI FF setup time (Libero SoC 11.7) = DDR_AXI FF setup time (Libero SoC 11.5) + (n - 1) * DDR clock period == DDR_AXI FF setup time (Libero SoC 11.5) + 2* DDR clock period
set delay1 [ expr 5000/$ddr_clock_frequency ]
set delay2 [ expr 3000/$ddr_clock_frequency ]
set_max_delay $delay1 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARADDR*  */INST_MSS_*_IP:F_ARBURST*  */INST_MSS_*_IP:F_ARID* */INST_MSS_*_IP:F_ARLEN*\
*/INST_MSS_*_IP:F_ARLOCK*  */INST_MSS_*_IP:F_ARSIZE*  */INST_MSS_*_IP:F_AWADDR* */INST_MSS_*_IP:F_AWBURST* \
*/INST_MSS_*_IP:F_AWID*  */INST_MSS_*_IP:F_AWLEN*  */INST_MSS_*_IP:F_AWLOCK* */INST_MSS_*_IP:F_AWSIZE* \
*/INST_MSS_*_IP:F_WDATA*  */INST_MSS_*_IP:F_WID*  */INST_MSS_*_IP:F_WLAST */INST_MSS_*_IP:F_WSTRB* \
*/INST_MSS_*_IP:F_BREADY */INST_MSS_*_IP:F_RMW_AXI */INST_MSS_*_IP:F_RREADY \
} ]

set_max_delay $delay2 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARVALID* \
*/INST_MSS_*_IP:F_AWVALID* \
*/INST_MSS_*_IP:F_WVALID \
} ]
```

4:1 Ratio

When AXI mode is used with MDDR or FDDR subsystem operating at 4:1 DDR to AXI clock ratio, timing closure can be achieved by inserting three flip-flops on the AWVALID, ARVALID, and WVALID signal paths. A two-input AND gate with inputs from the fabric AXI master VALID signals and the FDDR/MDDR READY signals is fed to the three-stage pipeline flip-flops.

If the design uses DDR to AXI clock ratio greater than 2:1, increasing the pipeline stages helps to increase timing margin without changing the clock frequency. For 4:1 DDR to AXI clock ratio, three positive edge triggered flip-flops are used in pipeline. These flip-flops are clocked using DDR clock (DDR_FIC_SUBSYSTEM_CLK*4) on the VALID signal paths. DDR clock is derived using user PLL (Fabric CCC) as shown in Figure 6 on page 10.

The optimization method can reside between an existing AXI master and the DDR_FIC AXI slave interface and no changes are required to the AXI master design. As the AXI VALID signals are delayed by three DDR clock cycles AXI data lines going into the DDR_FIC get an additional three DDR clock cycle time to become stable before the active edge of the latching clock.

Figure 6 shows the block diagram of the optimization technique for the 4:1 clock ratio.

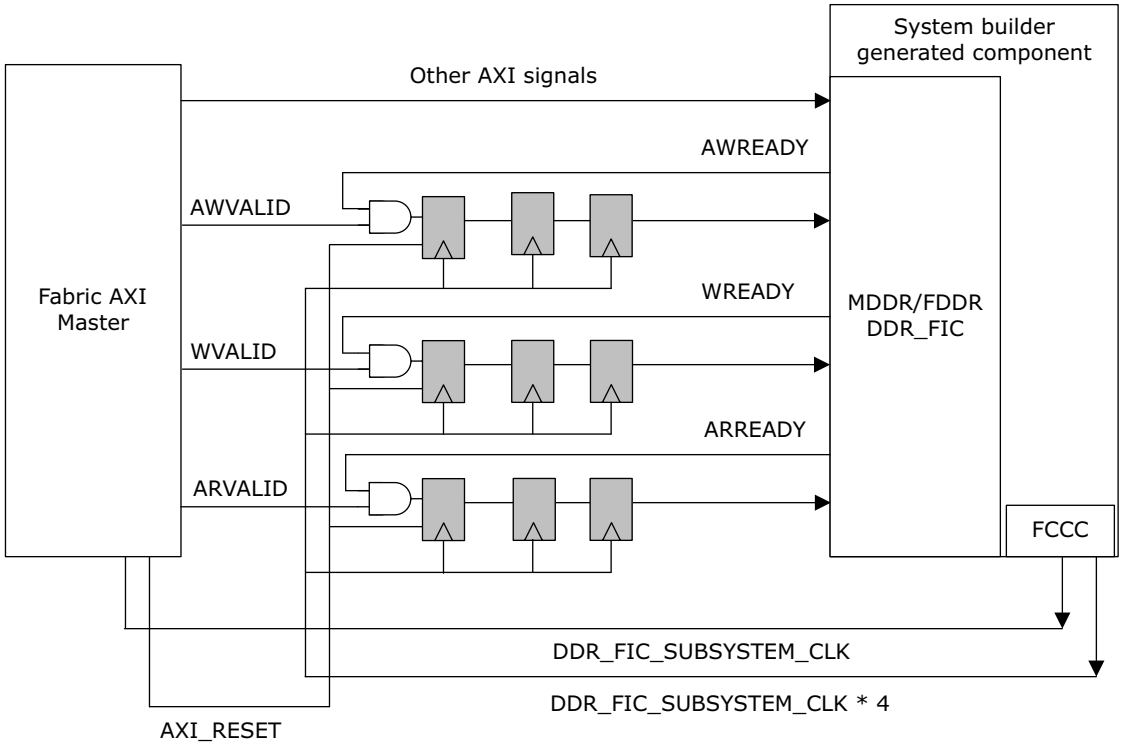

***Figure 6* • AXI Timing Optimization Logic for 4:1 Ratio**

Figure 7 shows the AXI transaction timing diagram with optimization logic for 4:1 ratio. The AXI data signal must meet four DDR clock or one AXI clock cycle setup time.

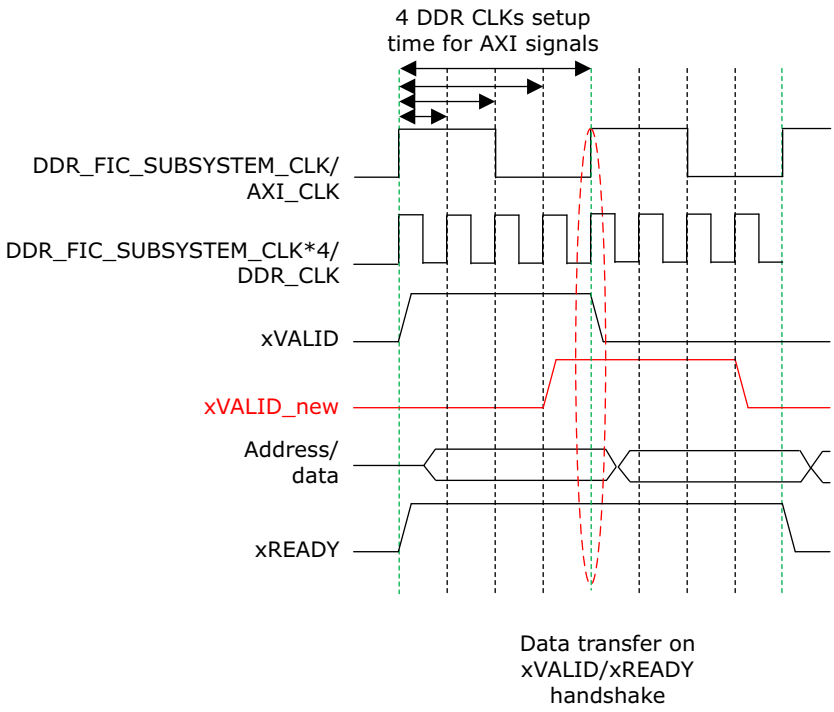

***Figure 7* • Timing Diagram for 4:1 Ratio**

When implementing 4:1 ratio timing optimization technique, the following SDC constraints need to be added to the timing constraint file (.sdc), which is provided as part of the design files. Refer to "Appendix A: Design files" on page 25 for more information.

For FDDR

```
# The users must adjust the ddr_clock_frequency to match their application.
set ddr_clock_frequency 320
# Apply new max delay for a 4:1 clock ratio (new valid paths need to meet one DDR clock cycle setup time and other paths need to meet one AXI clock cycle setup time)
# DDR_AXI FF setup time (Libero SoC 11.7) = DDR_AXI FF setup time (Libero SoC 11.5) + (n - 1) * DDR clock period == DDR_AXI FF setup time (Libero SoC 11.5) +3* DDR clock period
set delay1 [ expr 7000/$ddr_clock_frequency ]
set delay2 [ expr 4000/$ddr_clock_frequency ]
set_max_delay $delay1 -to [ get_pins { \
*/INST_FDDR_IP:F_ARADDR*  */INST_FDDR_IP:F_ARBURST*  */INST_FDDR_IP:F_ARID* */INST_FDDR_IP:F_ARLEN*\
*/INST_FDDR_IP:F_ARLOCK*  */INST_FDDR_IP:F_ARSIZE*  */INST_FDDR_IP:F_AWADDR* */INST_FDDR_IP:F_AWBURST* \
*/INST_FDDR_IP:F_AWID*  */INST_FDDR_IP:F_AWLEN*  */INST_FDDR_IP:F_AWLOCK* */INST_FDDR_IP:F_AWSIZE* \
*/INST_FDDR_IP:F_WDATA*  */INST_FDDR_IP:F_WID*  */INST_FDDR_IP:F_WLAST */INST_FDDR_IP:F_WSTRB* \
*/INST_FDDR_IP:F_BREADY */INST_FDDR_IP:F_RMW_AXI */INST_FDDR_IP:F_RREADY \
} ]
set_max_delay $delay2 -to [ get_pins { \
```

```
*/INST_FDDR_IP:F_ARVALID* \
*/INST_FDDR_IP:F_AWVALID* \
*/INST_FDDR_IP:F_WVALID \
} ]
```

For MDDR

```
# The users must adjust the ddr_clock_frequency to match their application.
set ddr_clock_frequency 320
# Apply new max delay for a 4:1 clock ratio (new valid paths need to meet one DDR clock cycle setup time and other paths need to meet one AXI clock cycle setup time)
# DDR_AXI FF setup time (Libero SoC 11.7) = DDR_AXI FF setup time (Libero SoC 11.5) + (n - 1) * DDR clock period == DDR_AXI FF setup time (Libero SoC 11.5) +3* DDR clock period
set delay1 [ expr 7000/$ddr_clock_frequency ]
set delay2 [ expr 4000/$ddr_clock_frequency ]
set_max_delay $delay1 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARADDR*  */INST_MSS_*_IP:F_ARBURST*  */INST_MSS_*_IP:F_ARID* */INST_MSS_*_IP:F_ARLEN*\
*/INST_MSS_*_IP:F_ARLOCK*  */INST_MSS_*_IP:F_ARSIZE*  */INST_MSS_*_IP:F_AWADDR* */INST_MSS_*_IP:F_AWBURST* \
*/INST_MSS_*_IP:F_AWID*  */INST_MSS_*_IP:F_AWLEN*  */INST_MSS_*_IP:F_AWLOCK* */INST_MSS_*_IP:F_AWSIZE* \
*/INST_MSS_*_IP:F_WDATA*  */INST_MSS_*_IP:F_WID*  */INST_MSS_*_IP:F_WLAST */INST_MSS_*_IP:F_WSTRB* \
*/INST_MSS_*_IP:F_BREADY */INST_MSS_*_IP:F_RMW_AXI */INST_MSS_*_IP:F_RREADY \
} ]

set_max_delay $delay2 -to [ get_pins { \
*/INST_MSS_*_IP:F_ARVALID* \
*/INST_MSS_*_IP:F_AWVALID* \
*/INST_MSS_*_IP:F_WVALID \
} ]
```

SmartFusion2 Reference Design Description

The design consists of MDDR controller, which is configured to access the DDR3 memory available in the SmartFusion2 Advanced Development Kit board. DDR fabric interface controller (DDR_FIC) is configured for the AXI bus interface. It also uses a two-port static random-access memory (TPSRAM) IP. Figure 8 shows the top-level view of the design. The highlighted block contains the timing optimization logic based on the DDR to AXI clock ratio used in the design, that is, 2:1, 3:1, and 4:1 ratio. Separate design files are provided for each DDR to AXI ratio. See "Appendix A: Design files" on page 25 for more information about design files.

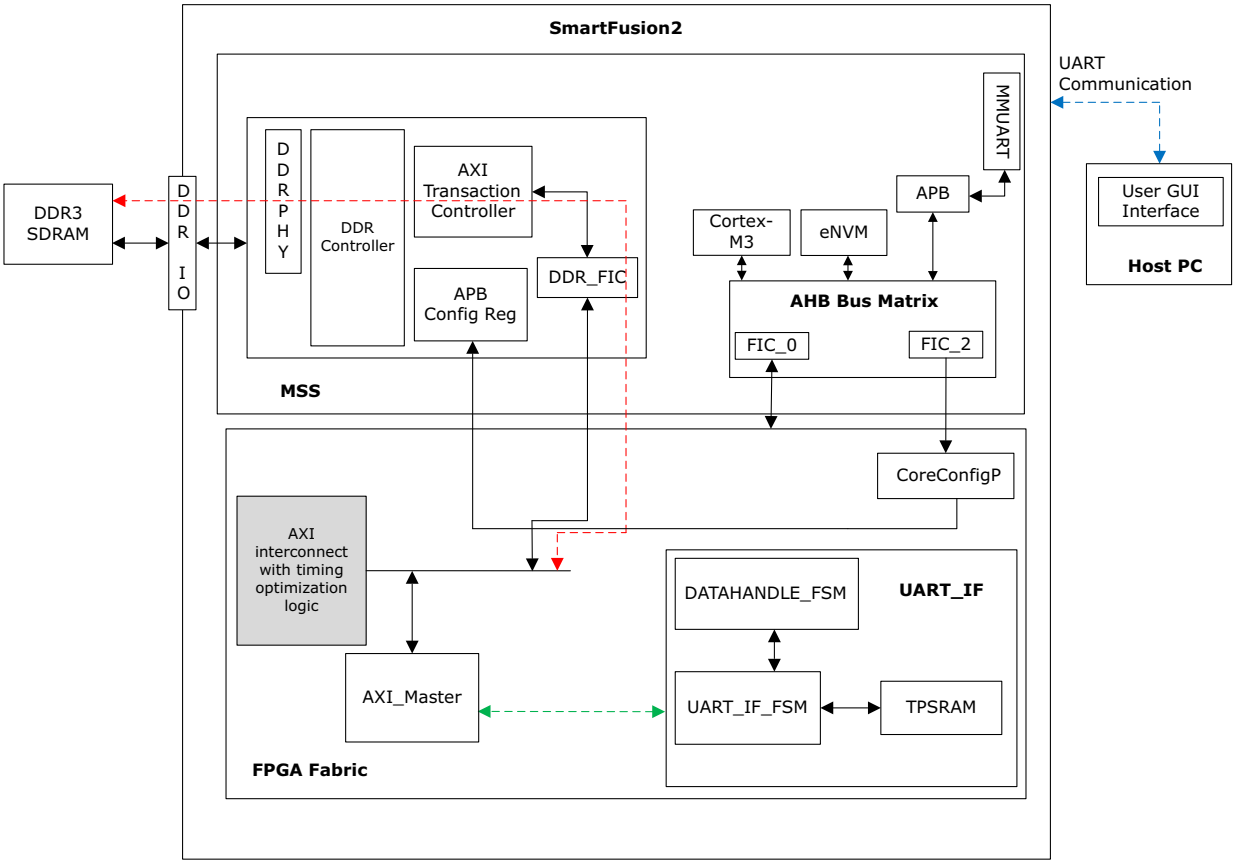

***Figure 8* • SmartFusion2 Top-Level Block Diagram**

Table 2 lists the MSS_CCC generated clocks for 2:1, 3:1, and 4:1 DDR to AXI CLK ratio designs.

Table 2 • **MSS_CCC Generated Clocks**

2:1 Ratio	
Clock Name	**Frequency in MHz**
M3_CLK	166.6
MDDR_CLK	333.2
DDR_SMC_FIC_CLK	166.6
APB_0	166.6
APB_1	166.6
FIC_0_CLK	83
3:1 Ratio	
Clock Name	**Frequency in MHz**
M3_CLK	111
MDDR_CLK	333
DDR_SMC_FIC_CLK	111
APB_0	111
APB_1	111
FIC_0_CLK	111
4:1 Ratio	
Clock Name	**Frequency in MHz**
M3_CLK	80
MDDR_CLK	320
DDR_SMC_FIC_CLK	80
APB_0	80
APB_1	80
FIC_0_CLK	80

The reference design provided in this application note consists of an AXI master implemented in the FPGA fabric, which accesses the DDR3 memory present in the SmartFusion2 Advanced Development Kit board using the MDDR controller. The AXI master logic communicates to the MDDR controller through the AXI interface and the DDR_FIC interface. The optimization method can reside between an existing AXI master and the DDR_FIC AXI slave interface and no changes are required to the AXI master design. If there are multiple masters, user can add arbiter logic and use the same optimization method.

The FIC_0 interface is configured to use a slave interface with the AHB-Lite (AHBL) interface. The FIC_2 interface is configured to initialize the MSS DDR using the ARM Cortex-M3 processor along with CoreConfigP macro. MMUART_0 is used as an interface for communicating with the host PC. MDDR is configured to use the DDR3 interface and routes the AXI interface to the FPGA fabric.

The read/write operations initiated by the SF2_MDDR_Demo utility are sent to the UART_IF block using the universal asynchronous receiver/transmitter (UART) protocol. AXI master receives the address and the data from the UART_IF block. During a write operation, the UART_IF block sends the address and data to the AXI master logic. During a read operation, it sends the address to the AXI master and stores the read data in TPSRAM. When the read operation is complete, the read data is sent to the host PC through UART.

SmartFusion2 Hardware Implementation

This section describes how the timing optimization logic is added to the reference design for the SmartFusion2 Advanced Development Kit board.

Figure 9 shows the top-level SmartDesign of the reference design for the 2:1 DDR to AXI clock ratio. The highlighted block is a register-transfer level (RTL) logic, which implements the AXI interconnect logic to connect an AXI master and a slave and the timing optimization logic for 2:1 ratio as discussed in the "Timing Optimization Techniques" section on page 3.

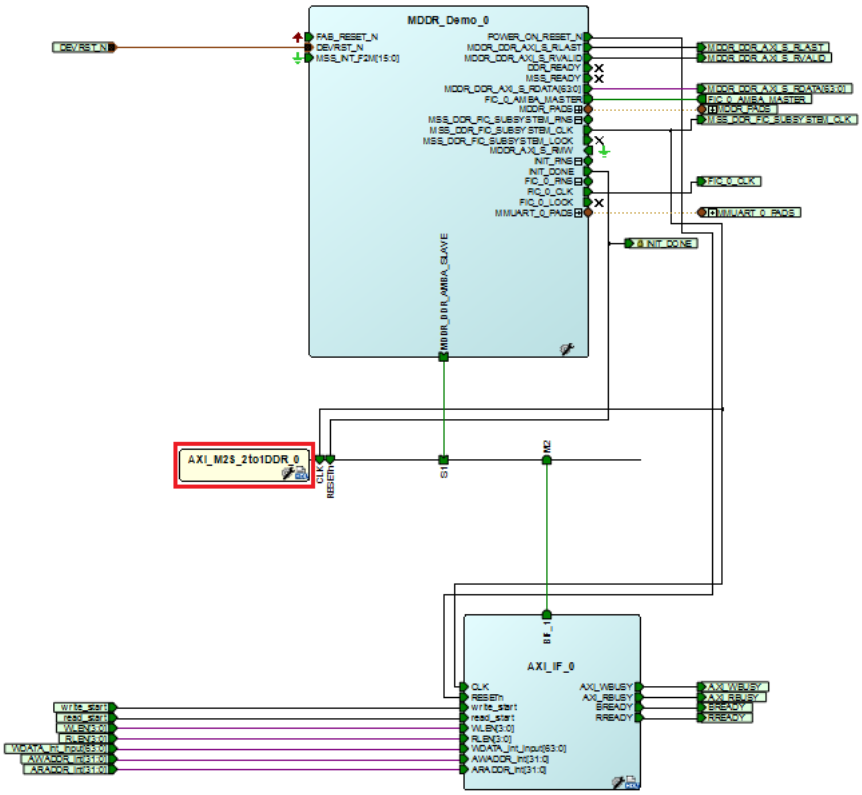

***Figure 9* • SmartFusion2 Top-Level SmartDesign for 2:1 Ratio**

Figure 10 shows the top-level SmartDesign of the reference design for 3:1 DDR to AXI clock ratio. The highlighted block is an RTL logic, which implements the AXI interconnect logic to connect an AXI master and a slave and the timing optimization logic for 3:1 ratio as discussed in the "Timing Optimization Techniques" section on page 3.

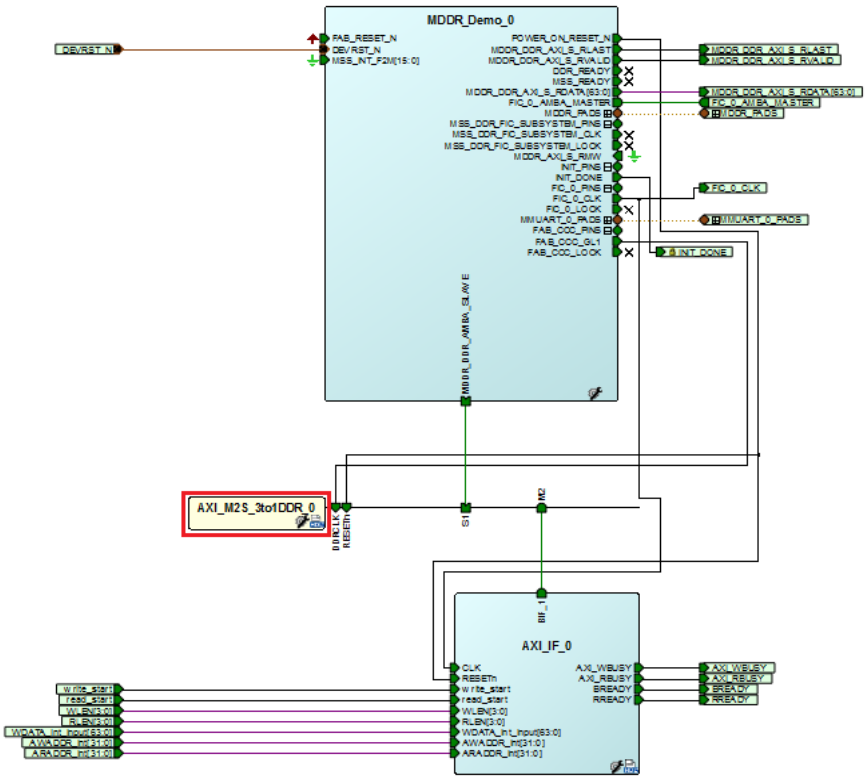

***Figure 10* • SmartFusion2 Top-Level SmartDesign for 3:1 Ratio**

Figure 11 shows the top-level SmartDesign of the reference design for 4:1 DDR to AXI clock ratio. The highlighted block is an RTL logic, which implements the AXI interconnect logic to connect an AXI master and a slave and the timing optimization logic for 4:1 ratio as discussed in the "Timing Optimization Techniques" section on page 3.

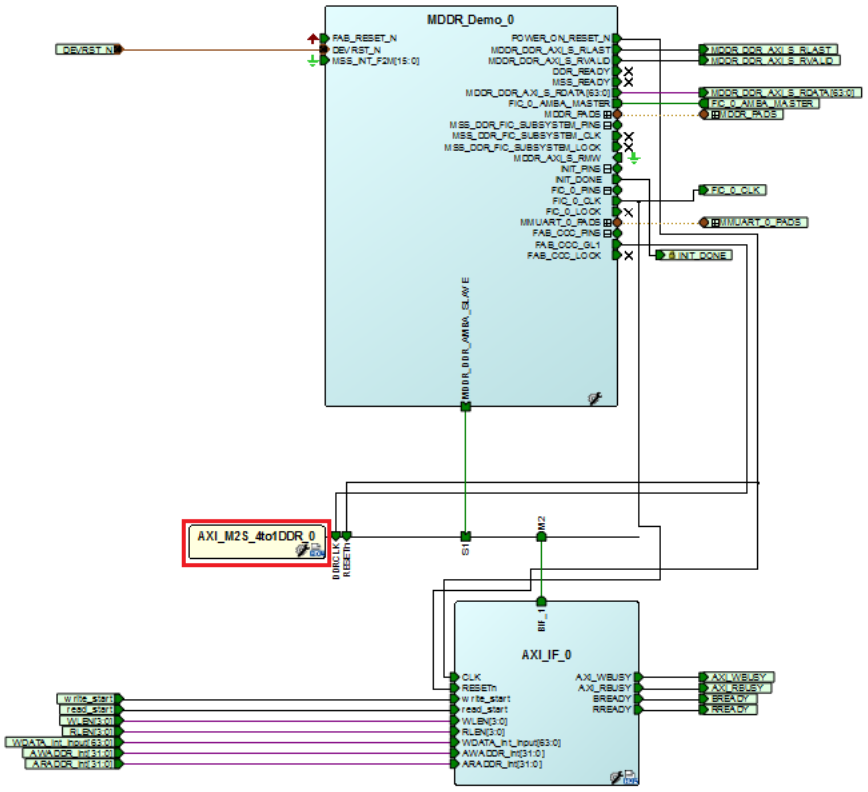

***Figure 11* • SmartFusion2 Top-Level SmartDesign for 4:1 Ratio**

For more information about other SmartDesign blocks, how to configure system builder, MDDR/FDDR subsystem, and DDR3 memory, see the following documents:

- *DG0568: Interfacing SmartFusion2 SoC FPGA with External LPDDR Memory through MDDR Controller Demo Guide*
- *UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide*

Note: Simulation model of timing optimization for AXI3 DDR interfaces using Smartfusion2 or IGLOO2 is not supported in the current software release.

SmartFusion2 Software Implementation

The software design reference performs the following operations:

- Setting the DDR3 SDRAM base address to 0xA0000000 and the FIC0 region base address to 0x30000000
- Initializing and configuring MMUART_0 to have 115200 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control
- Reading user-selected options from the GUI utility
- Reading 32-bit address and 64-bit data value from GUI utility, if user option is single or burst write
- Reading 32-bit address value, if user option is single or burst read
- Storing user option, data, and address value into the FIC0 registers, which are used to initiate AXI write or read transactions
- If the user option is single or burst write, address and data from UART_IF block are sent to AXI master
- If the user option is single or burst read, address value from UART_IF block is sent to AXI master and the read data is read from TPSRAM

List of firmware drivers used in this application:

- SmartFusion2 MSS NVM driver – Provides access to SmartFusion2 eNVM
- SmartFusion2 MSS HPDMA driver
- SmartFusion2 MSS UART driver – To communicate with the serial terminal program running on the host PC

IGLOO2 Reference Design Description

The reference design consists of MDDR controller, which is configured to access the LPDDR memory available in the IGLOO2 Evaluation Kit board. DDR_FIC is configured for the AXI bus interface. It also uses a TPSRAM IP. Figure 12 shows the top-level view of the design. The highlighted block contains the timing optimization logic based on the DDR to AXI clock ratio used in the design(2:1, 3:1, and 4:1). Separate design files are provided for each DDR to AXI ratio. See "Appendix A: Design files" on page 25 for more information about design files.

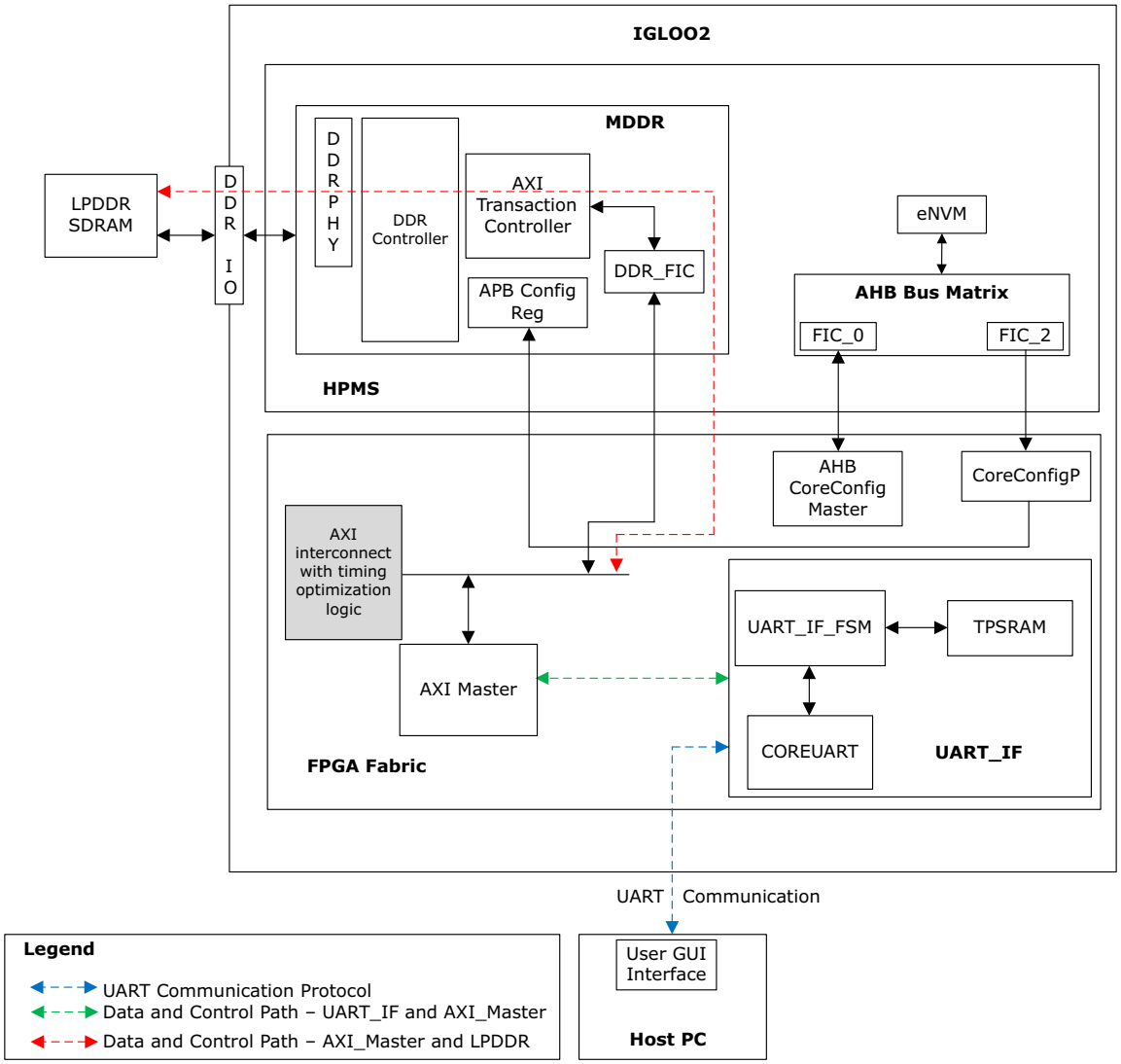

Figure 12 • **IGLOO2 Top-Level Block Diagram**

Table 3 lists the HPMS generated clocks for 2:1, 3:1, and 4:1 DDR to AXI CLK ratio designs

Table 3 • **HPMS Generated Clocks for 2:1, 3:1, and 4:1**

2:1 Ratio	
Clock Name	Frequency in MHz
HPMS_CLK	83
MDDR_CLK	166
DDR_SMC_FIC_CLK	83
FIC_0_CLK	20.75
3:1 Ratio	
Clock Name	Frequency in MHz
HPMS_CLK	55.3
MDDR_CLK	166
DDR_SMC_FIC_CLK	55.3
FIC_0_CLK	13.825
4:1 Ratio	
Clock Name	Frequency in MHz
HPMS_CLK	41.5
MDDR_CLK	166
DDR_SMC_FIC_CLK	41.5
FIC_0_CLK	10.375

In this reference design, AXI master implemented in the FPGA fabric accesses the LPDDR memory present in the IGLOO2 Evaluation Kit board using the MDDR controller. The AXI master logic communicates to the MDDR controller through the CoreAXI interface and the DDR_FIC interface.The read/write operations initiated by the IGL2_MDDR_Demo utility are sent to the UART_IF block using the UART protocol. AXI master receives the address and data from the UART_IF block.

During a write operation, the UART_IF block sends the address and data to the AXI master logic. During a read operation, the UART_IF block sends the address to AXI master and stores the read data in TPSRAM. When the read operation is complete, the read data is sent to the host PC through UART.

Note: IGLOO2 hardware implementation is similar to SmartFusion2.

For more information about IGLOO2 reference design for 2:1, 3:1, and 4:1 DDR to AXI clock ratio and how to run the design on the IGLOO2 Evaluation kit See "Appendix A: Design files" section on page 25 and *DG0534: Interfacing IGLOO2 FPGA with External LPDDR Memory through MDDR Controller.*

Jumper Settings for SmartFusion2

The following table lists the jumpers that need to be connected on the SmartFusion2 Advanced Development Kit board.

Table 4 • **SmartFusion2 Advanced Development Kit Jumper Settings**

Jumper	Pin (From)	Pint (To)	Comments
J116, J353, J354, J54	1	2	These are the default jumper settings of the Advanced Development Kit board. Ensure that these jumpers are set accordingly.
J123	2	3	
J124, J121, J32	1	2	JTAG programing through FTDI

Note: Switch OFF the power switch (SW7), while connecting the jumpers.

Connecting Host PC to Board

1. Connect the FlashPro4 programmer to the FP4 HEADER J37 connector of the SmartFusion2 Advanced Development Kit board.
2. Connect the J33 connector on the SmartFusion2 Advanced Development Kit board to the host PC using the USB mini-B (FTDI interface) cable.

Jumper Settings for IGLOO2

The following table lists the jumpers that need to be connected on IGLOO2 Evaluation Kit board.

Table 5 • **IGLOO2 Evaluation Kit Jumper Settings**

Jumper	Pin (From)	Pint (To)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

Note: Switch OFF the power switch(SW7), while connecting the jumpers.

Connecting Host PC to Board

1. Connect the Flashpro4 programmer to the J5 connector of the IGLOO2 Evaluation Kit.
2. Connect the power supply to the J6 connector.
3. Switch ON the power supply switch (SW7).

USB Driver Installation

The FTDI D2XX driver for serial terminal communication can be installed through the FTDI mini USB cable. The drivers and the installation guide can be downloaded from www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.

Check the Device Manager to verify that the USB to UART bridge drivers are detected, as shown in Figure 13 and Figure 14 on page 23.

For the SmartFusion2 Advanced Development Kit board, ensure that the COM port location is specified as **on USB Serial Converter C**, as shown in Figure 13.

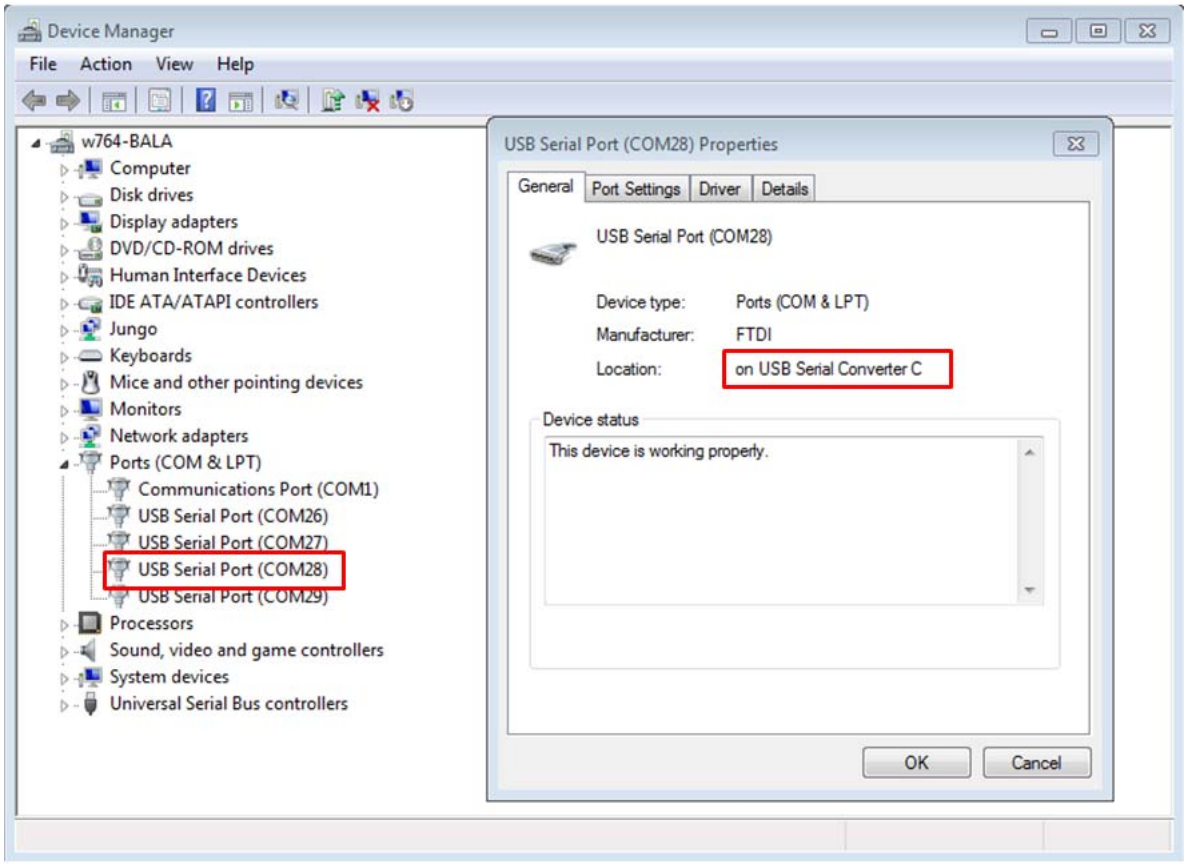

***Figure 13* • USB to UART Bridge Drivers for SmartFusion2 Advanced Development Kit Board**

For the IGLOO2 Evaluation Kit board, ensure that the COM port location is specified as **on USB Serial Converter D**, as shown in Figure 14.

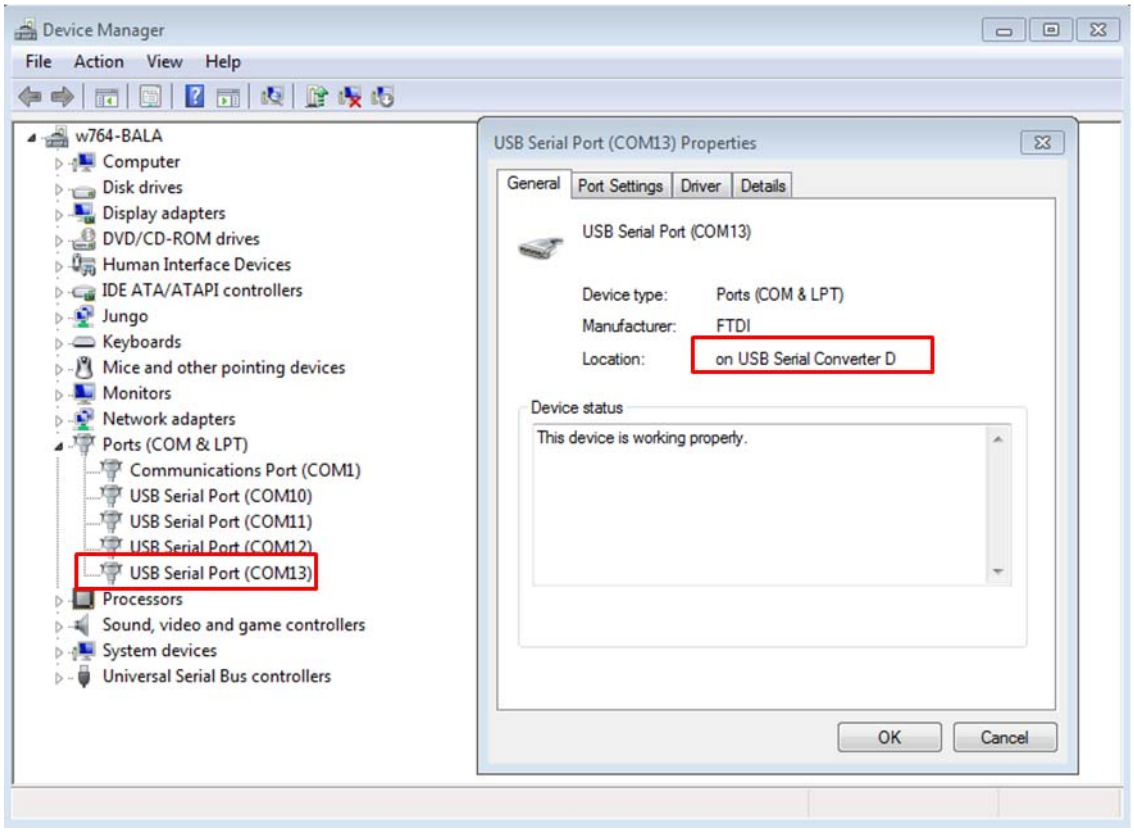

***Figure 14* • USB to UART Bridge Drivers for IGLOO2 Evaluation Kit Board**

For using USB 3.0, see the "Appendix B: Finding Correct COM port number when using the USB 3.0" section of the *DG0568: Interfacing SmartFusion2 SoC FPGA with External LPDDR Memory through MDDR Controller Demo Guide.*

Running the Design

1. Connect the power supply to the J42 connector for the SmartFusion2 Advanced Development Kit board or the J6 connector for the IGLOO2 Evaluation Kit board.
2. Connect the FlashPro4 programmer to the FP4 HEADER J37 connector for the SmartFusion2 Advanced Development Kit board or J5 connector for the IGLOO2 Evaluation Kit board.
3. Switch ON the power supply switch (SW7).
4. Program the SmartFusion2 Advanced Development Kit board or IGLOO2 Evaluation Kit board with the generated or provided *.stp file (Refer to "Appendix A: Design files" section on page 25) using FlashPro4.

For detailed instructions to run the design, see 'Running the Hardware Demo' section of the following documents:

- *DG0568: Interfacing SmartFusion2 SoC FPGA with External LPDDR Memory through MDDR Controller* for running the design on the SmartFusion2 Advanced Development Kit board.
- *DG0534: Interfacing IGLOO2 FPGA with External LPDDR Memory through MDDR Controller* for running the design on IGLOO2 Evaluation Kit board.

Conclusion

This application note describes the recommended optimization techniques for meeting timing closure on the SmartFusion2/IGLOO2 designs that use non 1:1 DDR to AXI clock ratios, that is, 2:1, 3:1, and 4:1 ratios.

Appendix A: Design files

The design files can be downloaded from the following link on the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2s_m2gl_ac450_liberov11p7_df

For the 2:1, 3:1, and 4:1 DDR to AXI clock ratios, the design files are available separately, each of which consists Libero SoC Verilog project, timing constraints file (.sdc), SoftConsole software project (SmartFusion2), demo utility (SmartFusion2 and IGLOO2), and programming files (*.stp) for the SmartFusion2 Advanced Development Kit board and the IGLOO2 Evaluation Kit board. Refer to the `Readme.txt` file included in the design file for the directory structure and description.

Appendix B: Implementation of Timing Optimization Logic

This appendix provides information on the RTL logics to be added to the AXI interconnect block when implementing the timing optimization logic for 2:1, 3:1, and 4:1 DDR:AXI clock ratio-based designs. The registers highlighted in red are the new VALID signals sent to the AXI slave interface.

Add the following RTL logic for the 2:1 DDR to AXI clock ratio designs in the AXI interconnect block.

```
/****************2:1 DDR:AXI clock ratio timing optimization technique ******************/
always@(negedge CLK, negedge RESETn)
begin
  if( RESETn == 1'b0 )
    begin
        AWVALID_new <= 1'b0;
        WVALID_new  <= 1'b0;
        ARVALID_new <= 1'b0;

    end
  else
    begin
        AWVALID_new <= AWVALID && AWREADY;
        WVALID_new  <= WVALID && WREADY;
        ARVALID_new <= ARVALID && ARREADY;

    end
end
/****************2:1 DDR:AXI clock ratio timing optimization technique ******************/
```

Add the following RTL logic for the 3:1 DDR to AXI clock ratio designs in the AXI interconnect block.

```
/****************3:1 DDR:AXI clock ratio timing optimization technique ******************/
always@(posedge DDRCLK, negedge RESETn)
begin
  if( RESETn == 1'b0 )
    begin
        AWVALID_1 <= 1'b0;
        WVALID_1  <= 1'b0;
        ARVALID_1 <= 1'b0;

    end
  else
    begin
        AWVALID_1 <= AWVALID && AWREADY;
        WVALID_1  <= WVALID && WREADY;
        ARVALID_1 <= ARVALID && ARREADY;

    end
end

always@(posedge DDRCLK, negedge RESETn)
begin
  if( RESETn == 1'b0 )
    begin
        AWVALID_new <= 1'b0;
        WVALID_new  <= 1'b0;
        ARVALID_new <= 1'b0;

    end
  else
    begin
        AWVALID_new <= AWVALID_1;
        WVALID_new  <= WVALID_1;
        ARVALID_new <= ARVALID_1;

    end
end
/****************3:1 DDR:AXI clock ratio timing optimization technique ******************/
```

Add the following RTL logic for the 4:1 DDR to AXI clock ratio designs in the AXI interconnect block.

```
/*****************=4:1 DDR:AXI clock ratio timing optimization technique ******************/

always@(posedge DDRCLK, negedge RESETn)
begin
  if( RESETn == 1'b0 )
    begin
        AWVALID_1 <= 1'b0;
        WVALID_1  <= 1'b0;
        ARVALID_1 <= 1'b0;

    end
  else
    begin
        AWVALID_1 <= AWVALID && AWREADY;
        WVALID_1  <= WVALID && WREADY;
        ARVALID_1 <= ARVALID && ARREADY;

    end
end

always@(posedge DDRCLK, negedge RESETn)
begin
  if( RESETn == 1'b0 )
    begin
        AWVALID_2 <= 1'b0;
        WVALID_2  <= 1'b0;
        ARVALID_2 <= 1'b0;

    end
  else
    begin
        AWVALID_2 <= AWVALID_1;
        WVALID_2  <= WVALID_1;
        ARVALID_2 <= ARVALID_1;

    end
end

always@(posedge DDRCLK, negedge RESETn)
begin
  if( RESETn == 1'b0 )
    begin
        AWVALID_new <= 1'b0;
        WVALID_new  <= 1'b0;
        ARVALID_new <= 1'b0;

    end
  else
    begin
        AWVALID_new <= AWVALID_2;
        WVALID_new  <= WVALID_2;
        ARVALID_new <= ARVALID_2;

    end
end

/*****************=4:1 DDR:AXI clock ratio timing optimization technique ******************/
```

For more information about how to connect user logic with AXI interface and how to configure AXI interconnect on SmartFusion2 devices, see the *AC409: Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks in the SmartFusion2 Devices Application Note.*

List of Changes

The following table shows the important changes made in this document for each revision.

Revision	Changes	Pages
Revision 1 (May 2016)	Initial release	NA

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
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center 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; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

51900450-1/05.16