SmartFusion2 SoC FPGA - Dynamic Configuration of AHB Bus Matrix - Libero SoC v11.7

Table of Contents

Purpose

This application note describes how to configure the weight values dynamically for the advanced high performance bus (AHB) bus matrix masters to access the AHB bus matrix slave using the weighted round-robin (WRR) arbitration in a SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device.

Introduction

SmartFusion2 SoC FPGA devices support the AHB bus matrix, which is a multi-layer AHB bus matrix. The AHB bus matrix has ten masters and seven direct slaves. This application note provides a reference design that has two fabric masters connected to the FIC_0 and FIC_1 interfaces. These two fabric masters can access a single slave Embedded SRAM (eSRAM1) using the WRR arbitration.

AHB Bus Matrix Overview

The connection of the masters and slaves to the AHB bus matrix is shown in Figure 1. The AHB bus matrix allows multiple masters to access a single slave through an arbitration mechanism.

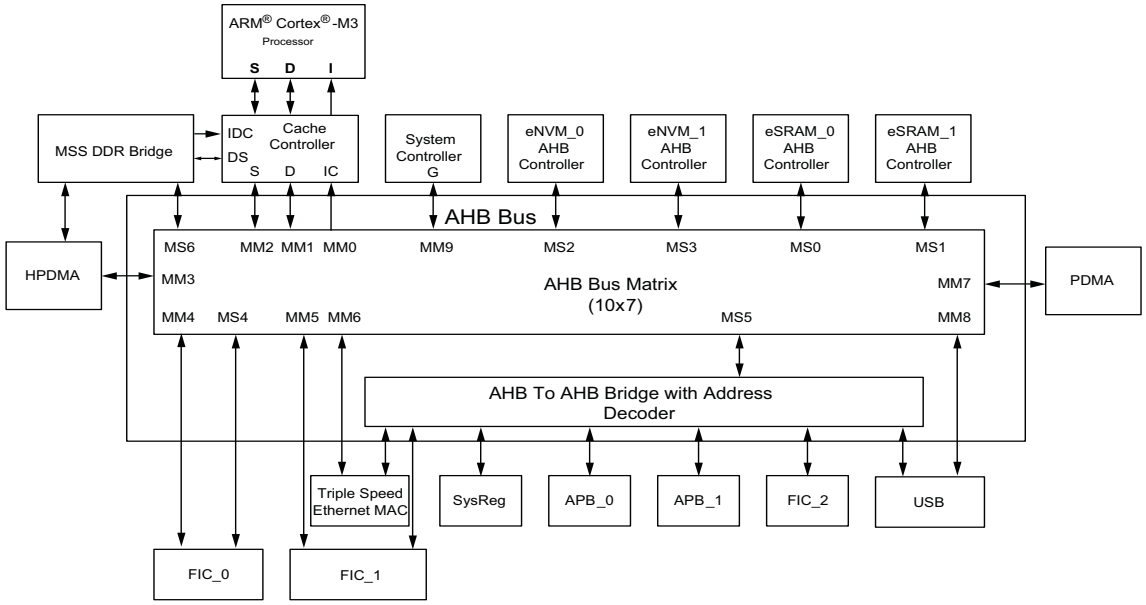

Figure 1 • **AHB Bus Matrix Block Diagram with all the 10 Masters and 7 Slaves (10 x 7)**

Arbitration

Arbitration is performed at two levels when more than one master attempts to access a single slave at the same time. At the first-level, the fixed higher priority masters (processor bus masters MM0, MM1, MM2, and MM9) are evaluated for any access request to the slave. The non-processor buses are then evaluated in a round-robin fashion for any access request to the slave. The arbitration mechanism uses pure round-robin and the WRR techniques.

The priority levels of each master are listed in Table 1.

Table 1 • **Master's Priority During Slave Arbitration**

MM Number	Masters	Priority
MM0	IC-Bus	2 Fixed
MM1	D-Bus	1 Fixed
MM2	S-Bus	3 Fixed
MM3	HPDMA	4 WRR
MM4	FIC_0	4 WRR
MM5	FIC_1	4 WRR
MM6	MAC	4 WRR
MM7	PDMA	4 WRR
MM8	USB	4 WRR
MM9	G	4 Fixed

Pure Round-Robin Arbitration

This is the default arbitration mode after reset wherein the programmable weight value for each of the master is 1. In this mode, the arbitration scheme for each slave port is identical. The processor masters have higher priority over the non-processor masters. Each non-processor master accessing a slave has equal priority based on a round-robin fashion.

For locked transactions, the master issuing the lock retains ownership of the slave until the locked transaction is complete. The priorities for masters in pure round-robin arbitration is shown in Figure 2.

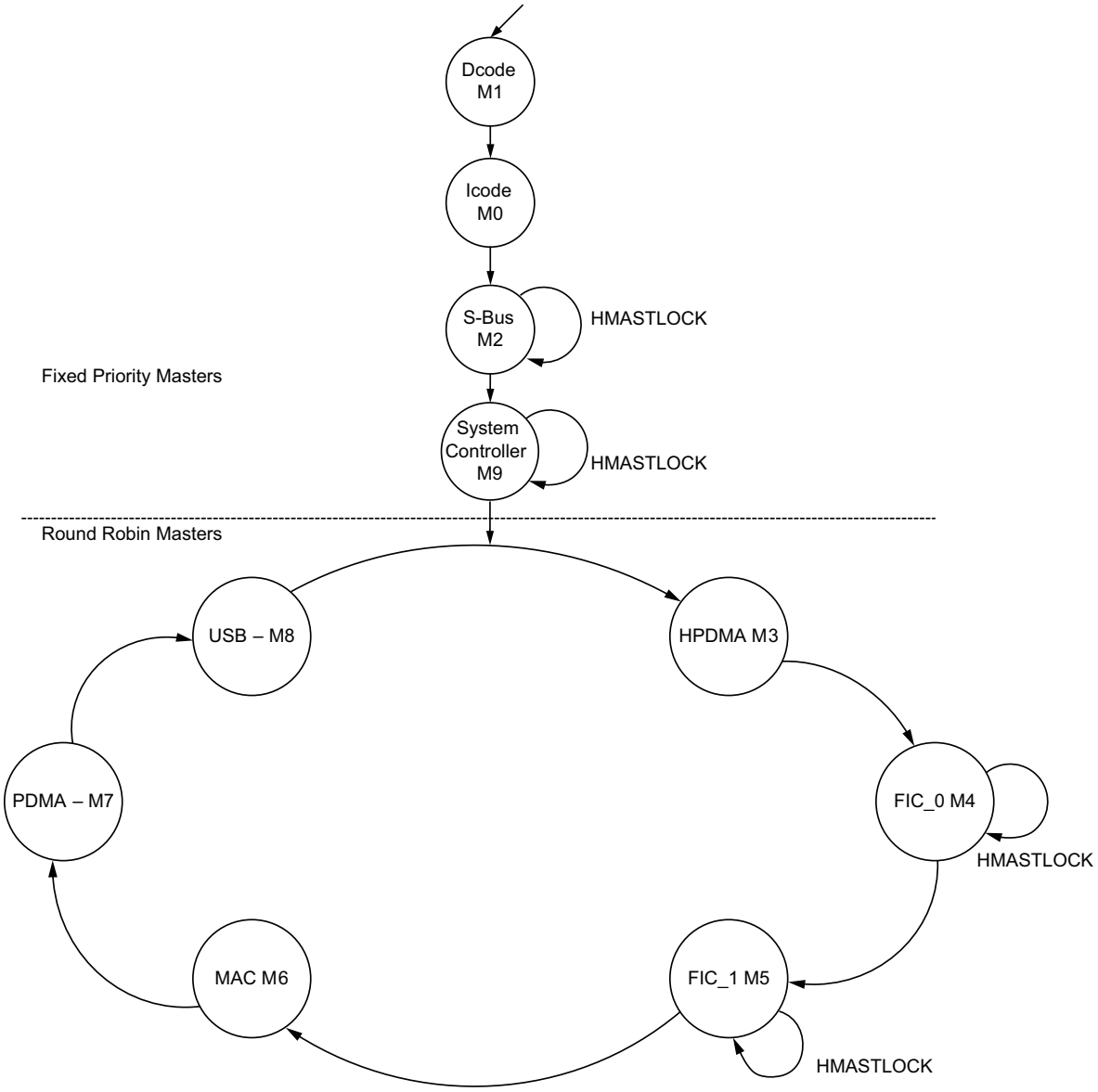

***Figure 2* • Pure Round Robin Arbitration Scheme**

WRR Arbitration

In this mode, the Programmable Weight (SW_WEIGHT_<master name>) can be configured to operate as WRR. The slave arbiter operates on a round-robin basis with each of the master interfaces having a maximum of N consecutive access opportunities to the slave in each “round” of arbitration. The value of N is determined by the programmed weight for the master and the maximum latency of the eSRAM0/1 parameter.

Each master (except the D-Code processor bus) has a programmable weight value that can be configured from 1 to 32. Maximum latency values for fixed priority masters can be configured from 1 to 8. The D-code bus does not need a programmable weight because it has the highest priority. The arbitration scheme of each slave on WRR arbitration is shown in Figure 3.

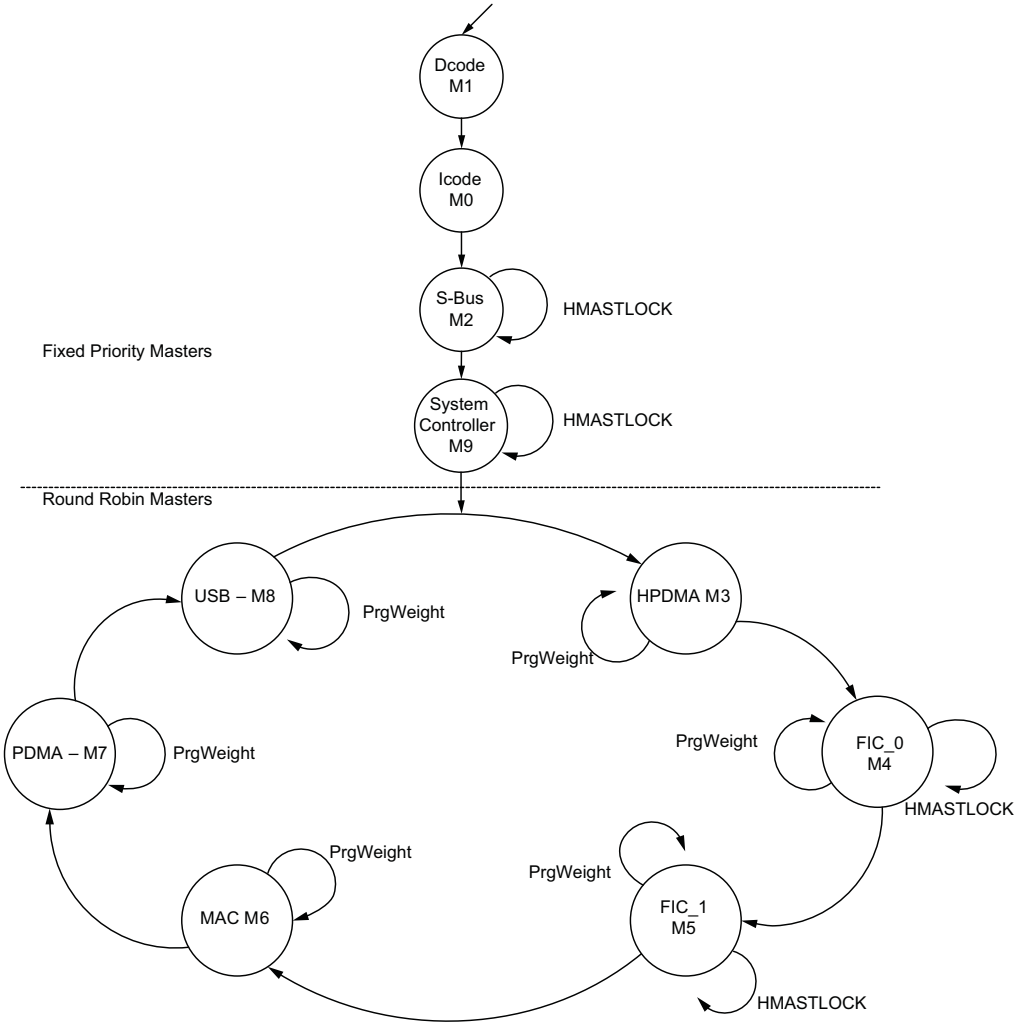

***Figure 3* • WRR and Fixed priority Slave Arbitration Scheme**

References

The following list of references is used in this document:

- *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*
- *TU0310: Interfacing User Logic with the Microcontroller Subsystem Tutorial*

Design Requirements

Table 2 lists the design requirements.

Table 2 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Advanced Development Kit: • 12 V adapter • USB A to mini-B cable	Rev B 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*
USB to UART drivers	–
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	–

Note: *For this application note, SoftConsole v3.4 SP1 is used. For using SoftConsole v4.0, see the *TU0546: SoftConsole v4.0 and Libero SoC v11.7 Tutorial*.

Design Description

The AHB bus matrix is configured with different weight values for multiple masters to access a single slave. This configuration can be done in the microcontroller subsystem (MSS) block using the Libero SoC software. Configuring weight values for masters from Libero SoC using the AHB Bus Matrix configurator is shown in Figure 4.

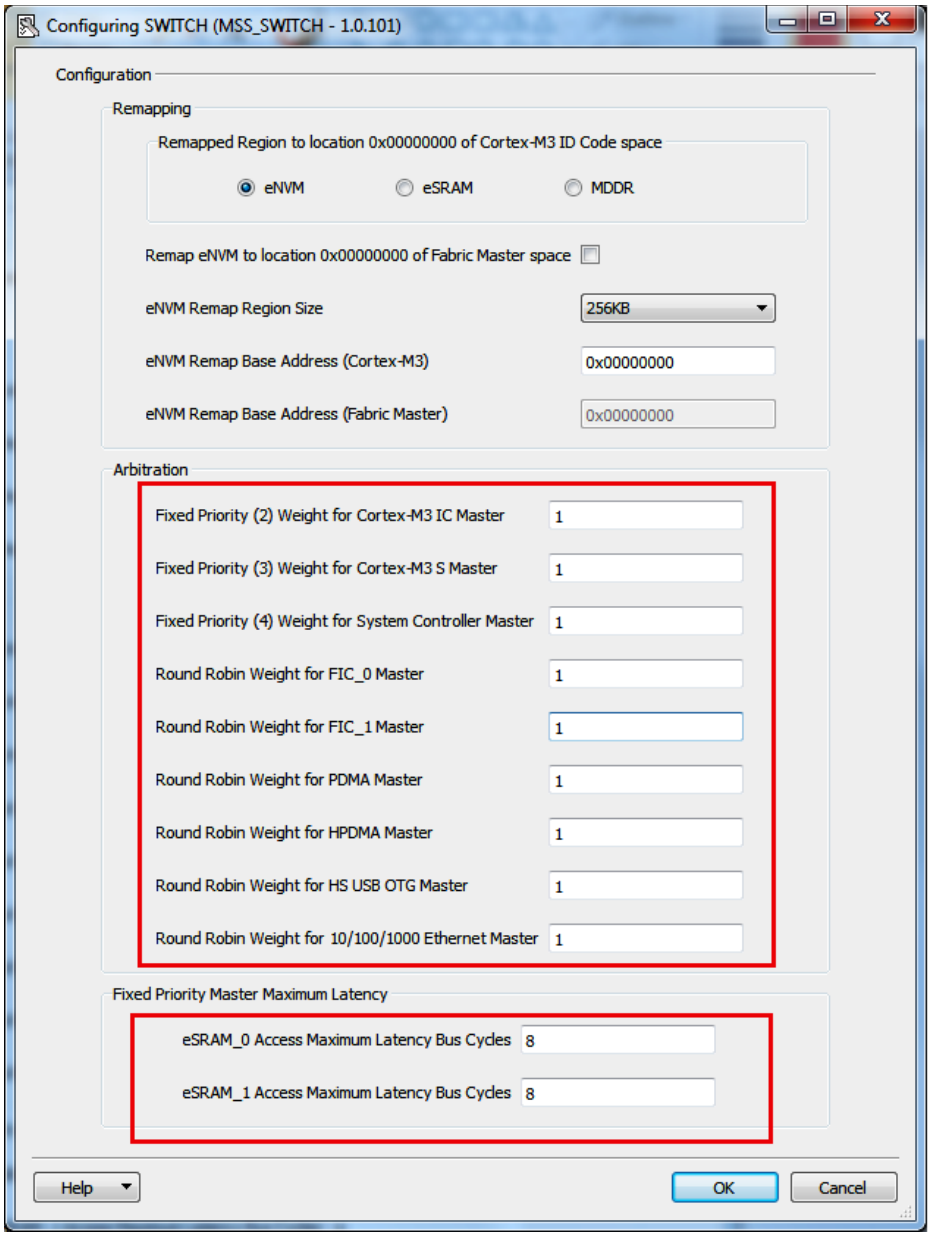

***Figure 4* • AHB Bus Matrix Configurator**

In this application note, for demonstration purposes, instead of changing the weight values in the AHB bus matrix configurator, the weight values for Fabric masters are configured at runtime by taking user entered weight values from HyperTerminal and writing the same to the weight configuration registers mentioned in Table 3 on page 7.

Table 3 provides the system registers that are used in this design for configuring the AHB bus matrix.

Table 3 • **System Registers**

Register	Description
MASTER_WEIGHT0_CR	Configures WRR master arbitration scheme for masters.
MASTER_WEIGHT1_CR	Configures WRR master arbitration scheme for masters.

Figure 5 shows the block diagram of the complete design.

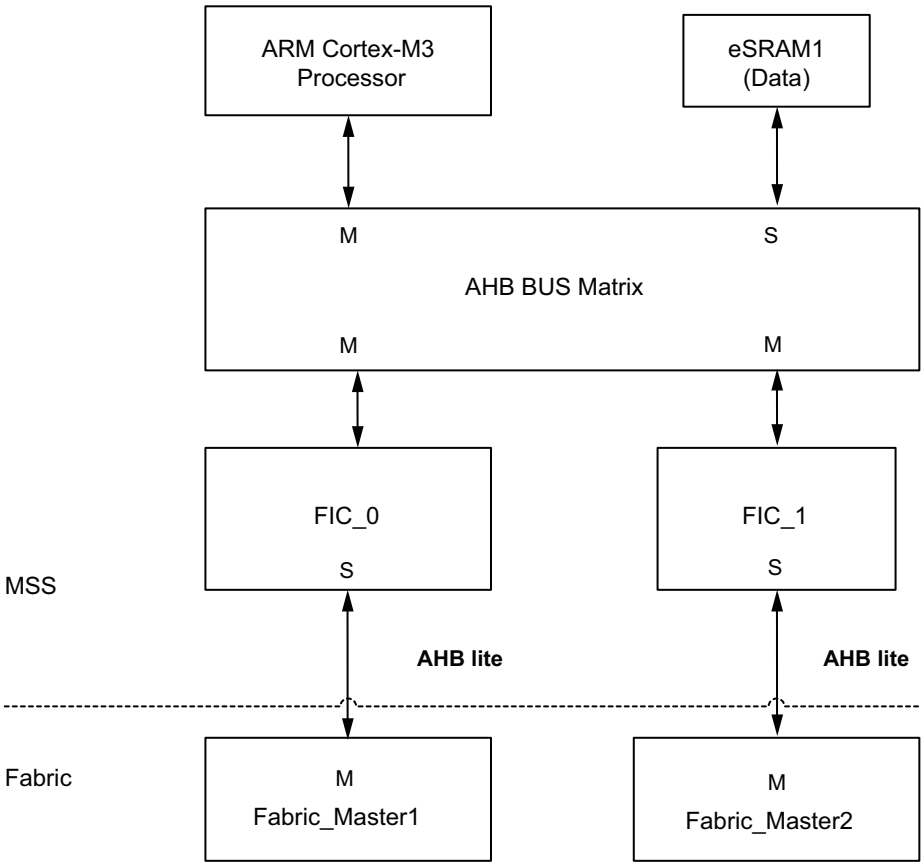

Figure 5 • **Reference Design Block Diagram**

Hardware Implementation

The example design consists of two AHB masters in the FPGA fabric that write 32-bit data to the AHB bus matrix slave eSRAM1. The Fabric_Master1 is connected to the slave interface of FIC_0 using Bypass mode and the Fabric_Master2 is connected to the slave interface of FIC_1using Bypass mode.

Figure 6 and Figure 7 on page 9 show the FIC_0 and FIC_1 configuration with interface type as AHB-Lite slave and the **Use Bypass Mode** option selected.

To implement WRR arbitration for fabric masters use only Bypass mode. Refer to the Fabric Interface Controller chapter of the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide* for more information on Bypass mode. Figure 11 on page 11 gives the SmartDesign window of all the blocks.

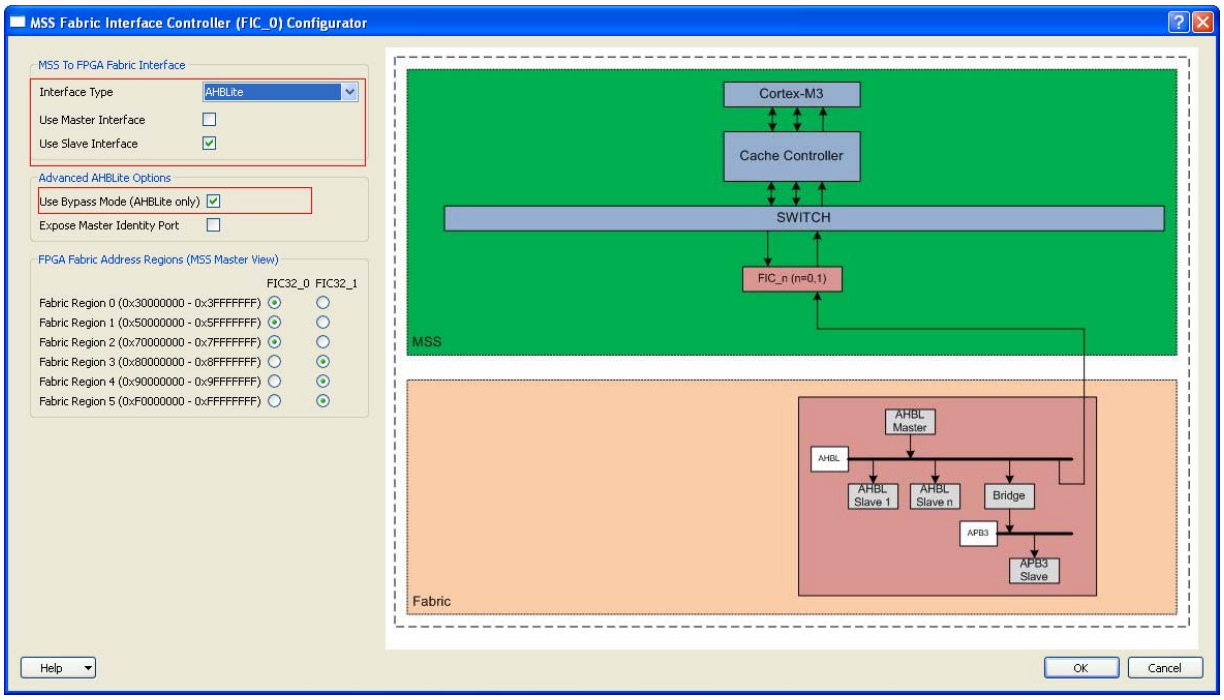

***Figure 6* • FIC_0 Configuration for Bypass Mode**

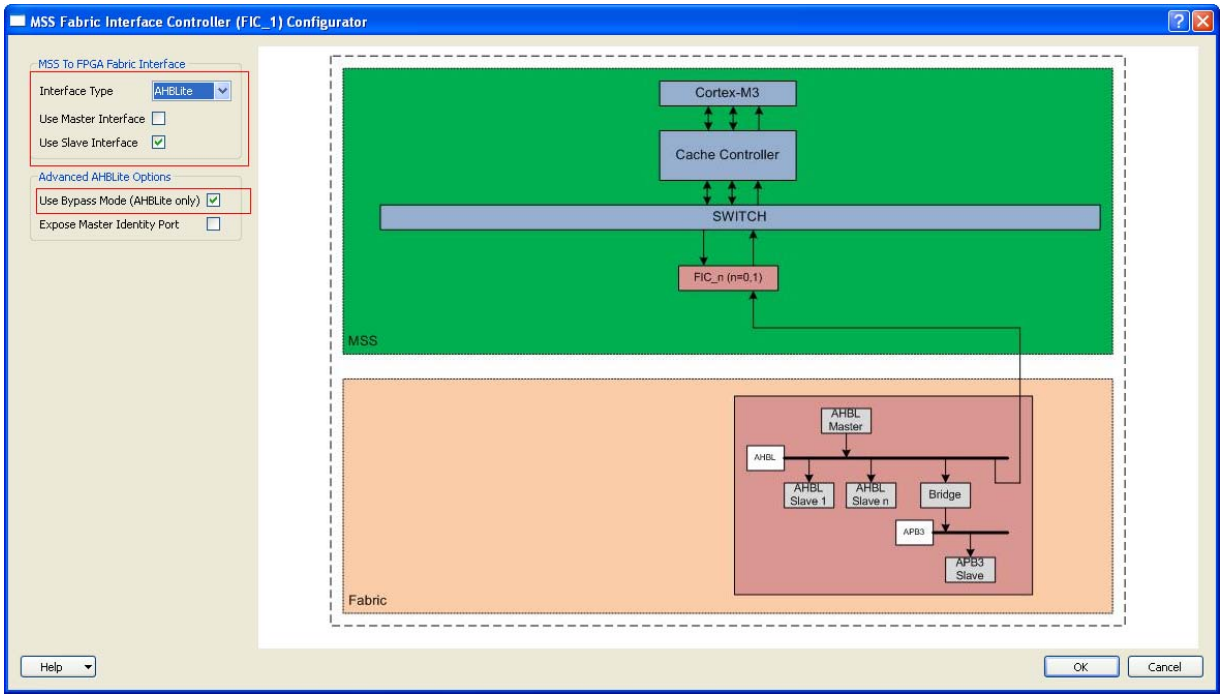

Figure 7 • **FIC_1 Configuration for Bypass Mode**

MSS is configured with MMUART0 connected to the fabric for the user interface to select the weight values from HyperTerminal. At this stage, Fabric_Master1 and Fabric_Master2 are running at 100 MHz clock.

Fabric_Master1 writes 1024 words to the eSRAM1 locations starting from the address 0x20008000 and Fabric_Master2 writes 1024 words to the eSRAM1 locations starting from the address 0x2000C000. Interrupt is generated when both masters complete 1024 transfers.

Number of accesses to slave for each master is displayed on HyperTerminal. Residual clock count for each master in the last access is displayed on HyperTerminal. Number of accesses taken by each master to complete 1024 transfers depends on weight configured for that master. Lesser weight master needs more number of accesses and higher weight master needs less number of accesses to transfer same number of words.

Simulation Results

Simulation results for Fabric_Master1 and Fabric_Master2 with different weights are shown in Figure 8, Figure 9, and Figure 10 on page 11.

Fabric_Master1 with weight 30 takes 35 accesses to write 1024 words to eSRAM1.In the last access number of clock cycles left are 26.

Fabric_Master2 with weight 26 takes 40 accesses to write 1024 words to eSRAM1. In the last access, number of clock cycles left are 16, as shown in Figure 8.

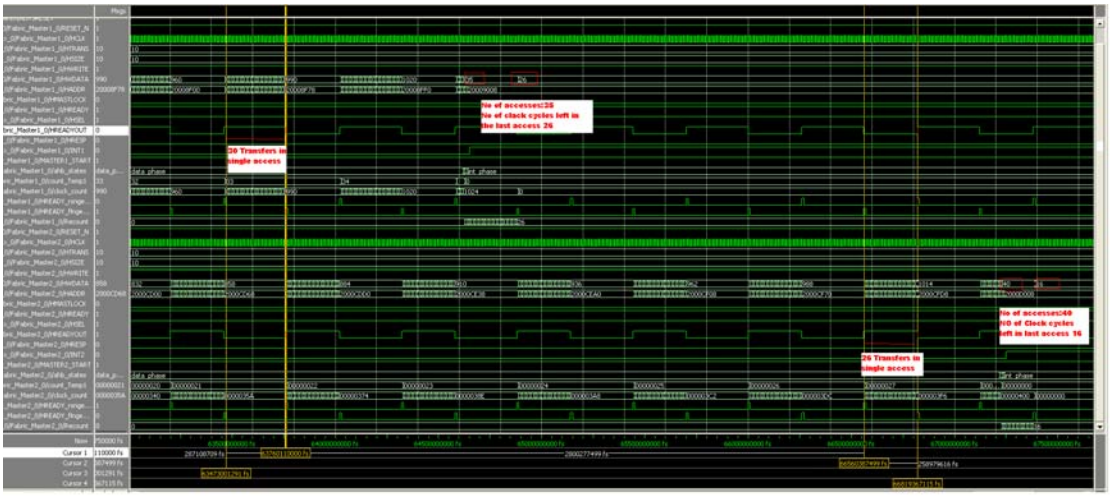

***Figure 8* • Simulation Results for Fabric_Master1 with Weight 30 and Fabric_Master2 with Weight 26**

Fabric_Master1 with weight 10 takes 103 accesses to write 1024 words to eSRAM1. In the last access, number of clock cycles left are 6. Fabric_Master2 with weight 12 takes 86 accesses to write 1024 words to eSRAM1. In the last access number of clock cycles left are 8, as shown in Figure 9.

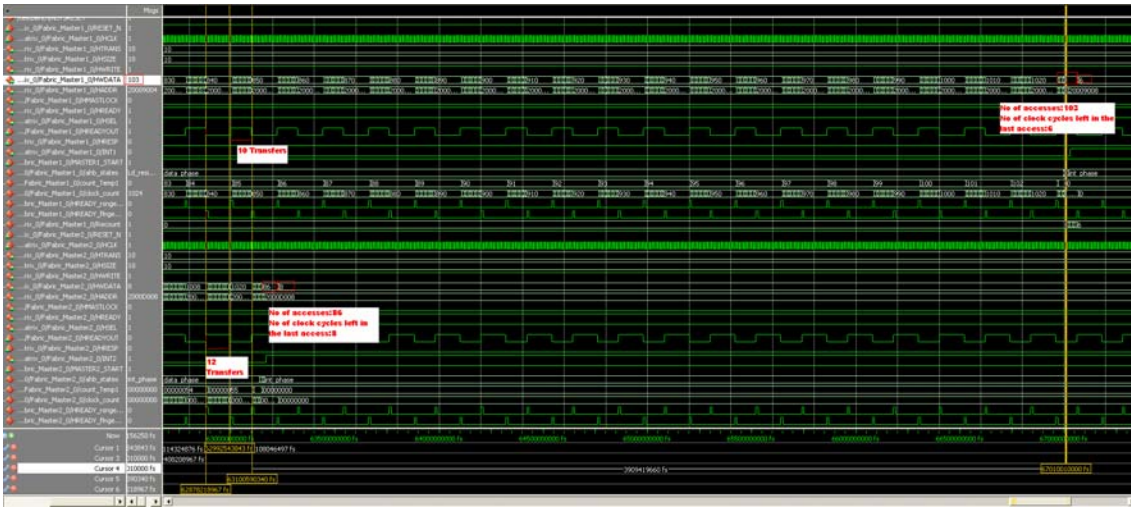

***Figure 9* • Simulation Results for Fabric_Master1 with Weight 10 and Fabric_Master2 with Weight 12**

Fabric_Master1 with weight 8 takes 128 accesses to write 1024 words to eSRAM1. In the last access number of clock cycles left are 0. Fabric_Master2 with weight 7 takes 147 accesses to write 1024 words to eSRAM1. In the last access, number of clock cycles left are 5, as shown in Figure 10.

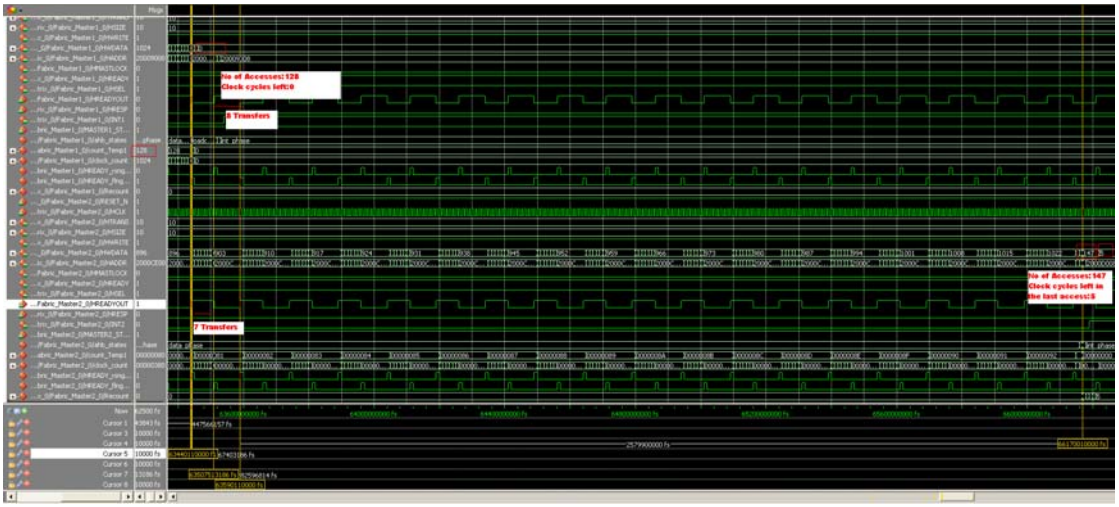

***Figure 10* • Simulation Results for Fabric_Master1 with Weight 8 and Fabric_Master2 with Weight 7**

Figure 11 illustrates the top-level hardware design.

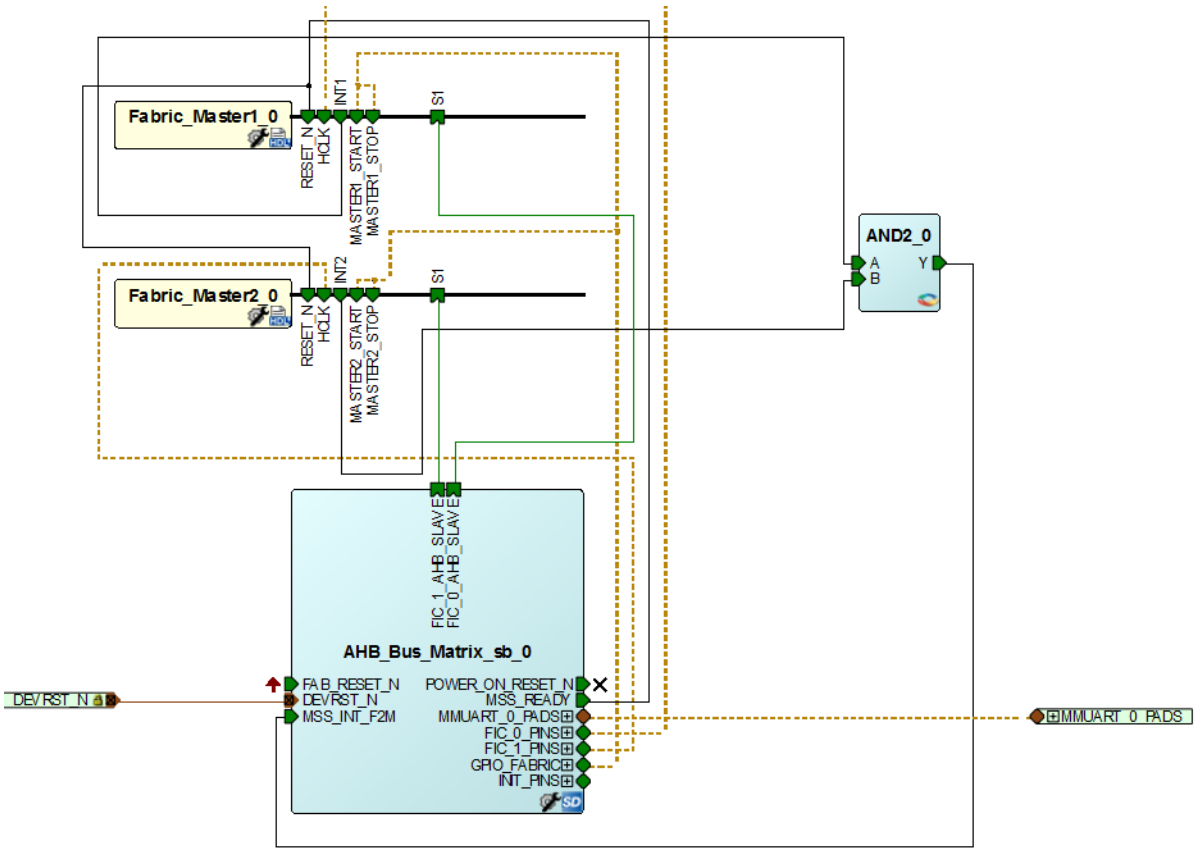

***Figure 11* • SmartDesign Window with Blocks in Hardware Design**

Software Implementation

The software design performs the following operations:

- Initializes the AHB bus matrix.
- Selects the weight values using HyperTerminal session.
- Displays the eSRAM1 accesses count and residual clock transfer count for each fabric master.

The following application program interface (APIs) are implemented in the application layer drivers of the AHB bus matrix:

- **AHBBus_init():** This API resets all the system registers of the AHB bus matrix mentioned in Table 3 on page 7.
- **void master_select():** This API takes weight values as inputs and decides the system registers to be modified. It calculates the value of weight to be set for the system register MASTER_WEIGHT0_CR/ MASTER_WEIGHT1_CR.
- **void set_weight():** In this API, the weight values calculated are assigned to the register MASTER_WEIGHT0_CR or the MASTER_WEIGHT1_CR based on the decision made in the above API.

The following firmware drivers are used in this application:

- SmartFusion2 MSS general-purpose input/output (GPIO) driver
- SmartFusion2 MSS Multi-Mode universal asynchronous receiver/transmitter (MMUART) driver:
  - To communicate with the serial terminal program running on host PC.

Figure 12 shows the flow of sample example implemented in `main.c`.

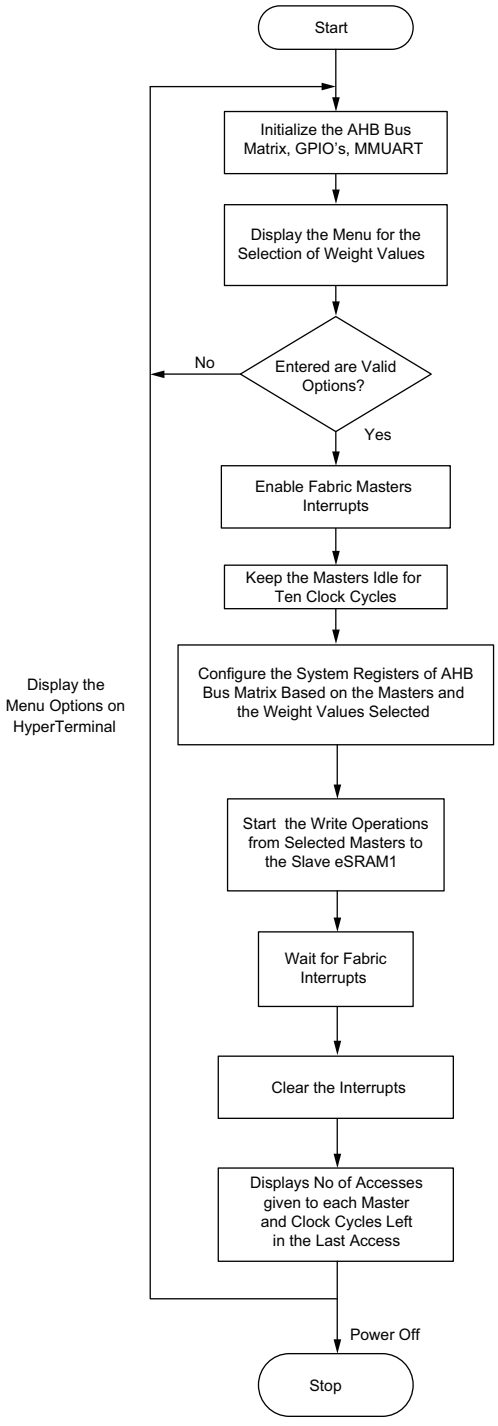

***Figure 12* • Flow Chart of the Application in the `main.c` File**

Running the Design

This section describes the board settings and steps to run the design.

Board Settings

Connect the jumpers on the SmartFusion2 Advanced Development Kit, as described in Table 4. While making the jumper connections, the power supply switch SW7 on the board must be in OFF position.

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 programming through FTDI.

Steps to Run the Design

The following steps describe how to run the design:

1. Connect the host PC to the J33 connector using the USB A to mini-B cable. The USB to UART bridge drivers are automatically detected. From the detected four COM ports, right-click one of the COM ports and select **Properties**. The selected COM port properties window is displayed as shown in Figure 13. Ensure that the Location in the **Properties** window is displayed as "**on USB FP5 Serial Converter C**", refer to Figure 13.

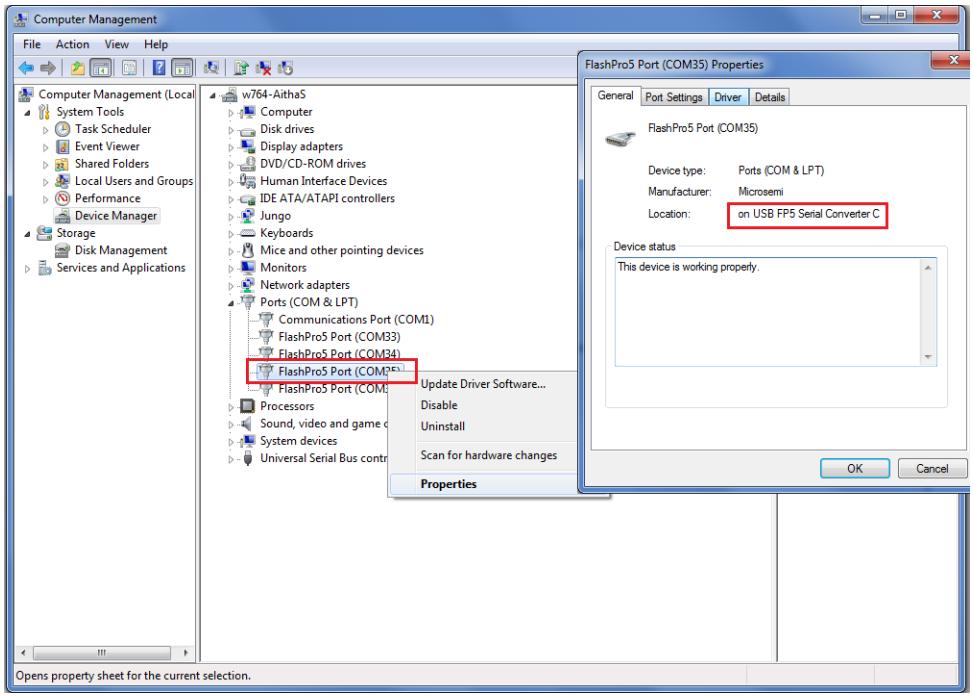

Figure 13 • **Properties Window**

2. Install the USB Driver, if USB drivers are not detected.

3. For serial terminal communication through the future technology devices international (FTDI) mini USB cable, install the FTDI D2XX driver. The drivers and installation guide can be downloaded from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*. Connect the power supply to the J42 connector and change the power supply switch SW7 to ON.
4. Start HyperTerminal session and select com port (as shown in Figure 14) with a 115,200 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If the HyperTerminal program is not available in the system, use other free serial terminal emulation programs such as PuTTY or TeraTerm. Refer to the *Configuring Serial Terminal Emulation Programs tutorial* for configuring HyperTerminal, TeraTerm, or PuTTY.
5. Program the SmartFusion2 Advanced Development Kit with the provided programming file (refer to "Appendix: Design and Programming Files" section on page 18) using FlashPro software and power cycle the board after successful programming. A welcome message is displayed as shown in Figure 14.

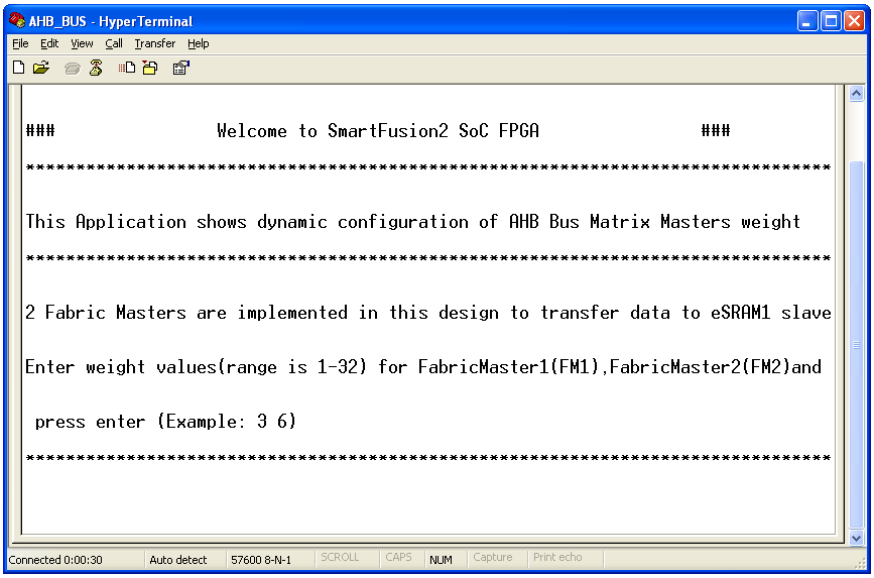

Figure 14 • **Welcome Message and Weight Selection in HyperTerminal Session**

6. Enter the weight values for the Fabric masters, as shown in Figure 15.

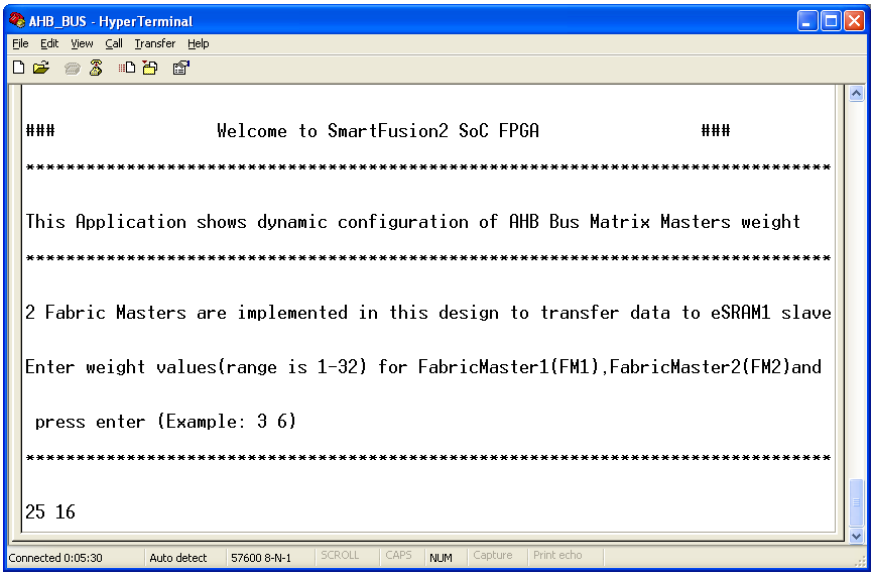

Figure 15 • **Entering Weight Values**

7. After entering the weight values, number of accesses taken by each master to write 1024 words to eSRAM1 and clock cycles left in the last access are displayed on HyperTerminal as shown in Figure 16 and Figure 17 on page 17.

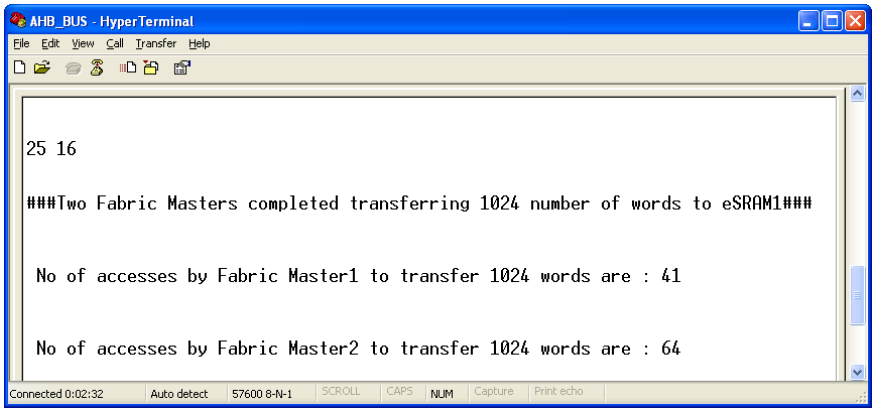

Figure 16 • **Displaying Number of eSRAM1 Accesses**

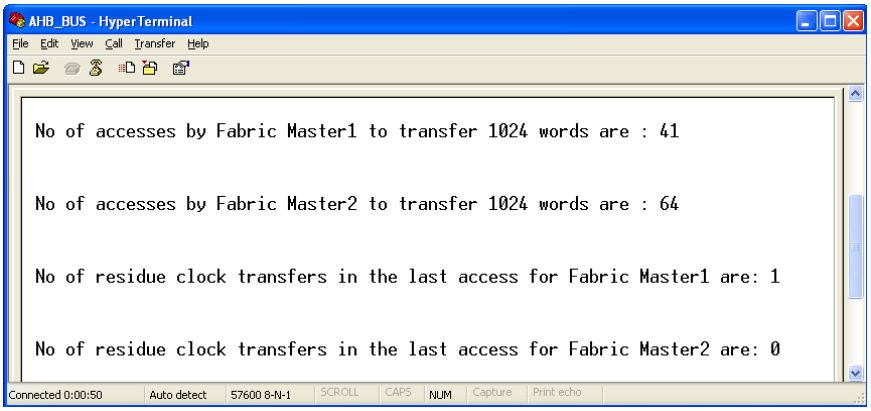

Figure 17 • **Displaying Residual Clock Transfers**

Note: The menu keeps repeating till the board is powered-down.

Conclusion

This application note shows the capabilities of the in-built AHB bus matrix of SmartFusion2 SoC FPGAs. The application level drivers described in this application note allow dynamic configuration of the AHB bus matrix master weight values as per the design requirements.

Appendix: Design and Programming Files

Download the design files from the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2s_ac388_liberov11p7_df

The design file consists of Libero VHDL, SoftConsole software project, and programming files (*.stp) for SmartFusion2 Advanced Development Kit.
Refer to the `Readme.txt` file included in the design file for the directory structure description and the changes to be done in the application code if the project is regenerated.

Download the programming files from the Microsemi SoC Products Group website:
http://soc.microsemi.com/download/rsc/?f=m2s_ac388_liberov11p7_pf

The programming file consists of STAPL programming file (*.stp) for SmartFusion2 Advanced Development Kit.

List of Changes

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

Revision*	Changes	Page
Revision 10 (February 2016)	Updated the document for Libero SoC v 11.7 software release (SAR 76537)	NA
Revision 9 (October 2015)	Updated the document for Libero SoC v 11.6 software release (SAR 71672).	NA
Revision 8 (February 2015)	Updated the document for Libero SoC v 11.5 software release (SAR 64192).	NA
Revision 7 (September 2014)	Updated the document for Libero SoC v 11.4 software release (SAR 61049).	NA
	Updates made to maintain the style and consistency of the document.	NA
Revision 6 (May 2014)	Updated Figure 11(SAR 57101).	11
	Added "Design Requirements" section (SAR 57101).	5
Revision 5 (November 2013)	Updated the document for Libero SoC v 11.2 software release (SAR 52886)	NA
Revision 4 (June 2013)	Updated the document for Libero SoC v11.0 software release (SAR 47624 and 46110).	NA
Revision 3 (March 2013)	Updated for Libero SoC v11.0 beta SP1 release (SAR 45835). The Release Mode section was removed along with Figures 10 and 11 that were updated in November 2012.	NA
Revision 2 (November 2012)	Added Release Mode section (SAR 42988).	15
	Modified "Running the Design" section (SAR 42988).	14
Revision 1 (November 2012)	Modified "Introduction" section (SAR 42846).	1
	Updated Figure 5, Figure 11, Figure 14, Figure 15, Figure 16, Figure 10, and Figure 11 (SAR 42846).	n/a
	Modified "Hardware Implementation" section (SAR 42846).	8
	Modified "Software Implementation" section (SAR 42846).	12
	Modified "Appendix: Design and Programming Files" section (SAR 42846).	18

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

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 & 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; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet Solution; 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**.

51900256-10/02.16