IGLOO2 FPGA SRAM Initialization from eNVM - Libero SoC v11.7

Table of Contents

Purpose

This application note describes how to initialize the IGLOO®2 field programmable gate array (FPGA) embedded static random access memory (SRAM) block with the user data after power-up. This application note explains the method for initializing the large SRAM (LSRAM) and microSRAM (µSRAM) blocks in IGLOO2 FPGA using a fabric master logic block. A design example is also provided in this application note.

Introduction

The IGLOO2 FPGA devices have SRAM blocks in the fabric. There are two types of SRAM blocks in the IGLOO2 FPGA fabric: LSRAMs and µSRAMs. LSRAMs are large SRAM blocks embedded in the IGLOO2 FPGA fabric device, and are used for creating big FIFOs or storing large amount of data. The µSRAMs are small SRAM blocks embedded in the IGLOO2 FPGA device. The LSRAM and uSRAM blocks are volatile memory types where the stored data disappears when the power goes off. After powering-up the device, the content of the SRAM is unknown. There are some applications, which require the SRAM data to be initialized and validated after powering-up the device.

Figure 2 on page 5 shows a block diagram of the design examples available in this application note. The SRAM initialization data is stored in eNVM during programming and after powering-up the device. A fabric master block transfers the data from the eNVM to SRAM blocks. The reference designs use the SRAM block configured as a two-port memory, but this approach can be used for all the variations of LSRAM and µSRAM in the IGLOO2 FPGA device. The reference design is simulated and tested using the IGLOO2 Evaluation Kit.

References

The following list of references is used in this document:

- *UG0445: IGLOO2 FPGA and SmartFusion2 SoC FPGA Fabric User Guide*
- *UG0447: SmartFusion2 SoC FPGA and IGLOO2 FPGA High-Speed Serial Interfaces User Guide*
- *TU0530: SmartFusion2 SoC FPGA and IGLOO2 FPGA SmartDebug Hardware Design Debug Tools Tutorial*
- *UG0478: IGLOO2 Evaluation Kit User Guide*

Design Requirements

Table 1 lists the design requirements.

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
IGLOO2 Evaluation Kit • 12 V Wall-Mounted Power Supply (Included with the kit) • FlashPro4 programmer (Included with the kit) • M2GL010T-1FGG484 device	Rev C or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero® System-on-Chip (SoC)	v11.7

Embedded SRAM Blocks in IGLOO2 FPGAs

This section describes different fabric SRAM blocks that are available in various IGLOO2 devices and clarifies their differences.

Table 2 lists different types of fabric SRAM blocks available in various IGLOO2 FPGA devices:

Table 2 • **SRAM Blocks in Various IGLOO2 FPGA Devices**

Features	M2GL005	M2GL010	M2GL025	M2GL050	M2GL090	M2GL150
LSRAM 18K Blocks	10	21	31	69	109	236
µSRAM 1K Blocks	11	22	34	72	112	240
Total RAM (KBits)	191	400	592	1314	2074	4488

LSRAM blocks can be configured as a dual-port SRAM or two-port SRAM. The LSRAM that is configured as the dual-port SRAM provides two independent access ports: Port A and Port B. In dual-port SRAM mode, data can be written to either or both the ports; also can be read from either or both the ports. Each port has:

- Its own address
- Data in
- Data out
- Clock
- Clock enable
- Write enable

The LSRAM configured as two-port SRAM has Port A dedicated for read operations, and Port B dedicated for write operations. The read and write operations in LSRAM are performed in the Synchronous mode and require a clock edge.

µSRAM has two read ports (Port A and Port B) and one write port (Port C). The read operation can be performed in both synchronous and asynchronous modes. The write operation can be done only in synchronous mode. See the *UG0445: SmartFusion2 SoC FPGA and IGLOO2 FPGA Fabric User Guide* for more information.

The SRAM blocks support rich variations in size and features of memory blocks for IGLOO2 FPGA devices; however, these variations require changes when initializing the SRAM blocks for a specific implementation; these changes do not affect the fundamentals of the reference design. Therefore, the reference design in this application note uses the LSRAM block configured as two-port memory. Customizing the reference design for different feature and size of SRAM is discussed in the "Customizing the Wrapper Interface" section on page 12.

IGLOO2 FPGA eNVM Controller for Data Storage

The design example uses the eNVM array in HPMS as the source of SRAM initialization. The flash memory block in the eNVM is used to store the SRAM initialization data, and the data is loaded to SRAM after powering-up the device. The eNVM Controller is an advanced high-performance bus (AHB) slave that provides access to eNVM. It converts the logical AHB addresses to physical eNVM addresses and allows commanding the eNVM to perform specific tasks such as read, write, and delete operations. See the *Embedded eNVM Controller* section in the *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide* for more information.

In the design examples, a data storage client is created to load the SRAM initialization, which is configured to be 64 x 8. Figure 1 shows the eNVM Configurator and the data storage client graphical user interface (GUI) in Libero SoC. To allow the eNVM data storage client for simulation, select the **Use Content for Simulation** check box.

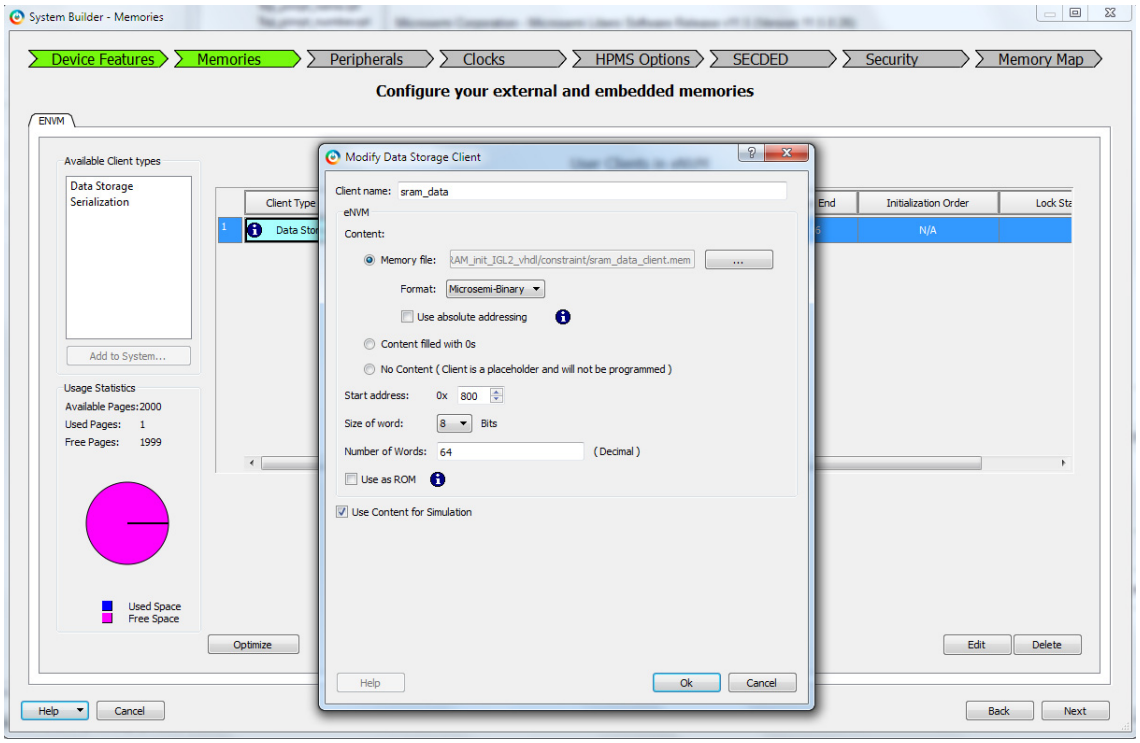

***Figure 1* • eNVM Configurator GUI**

SRAM Initialization Reference Design

The reference design is described and analyzed in the following sections:

- **Fabric Master Design Example**—discusses the functionality, architecture, and operation of the design.
- **Hardware Implementation**—discusses the hardware implementations and demonstrates the functionality of the code by running SmartDebug and looking at the SRAM content on the IGLOO2 Evaluation Kit board.
- **Customizing the Wrapper Interface**—provides guidelines on how to instantiate and how to use the reference design in the user design.

Fabric Master Design Example

This section describes the functionality, architecture, and operation of the fabric master design example. The fabric master acts as an advanced peripheral bus v3 (APB3) master to read data from eNVM after power-up and load the fabric SRAM block. Figure 2 shows a top-level block diagram of the fabric master design example.

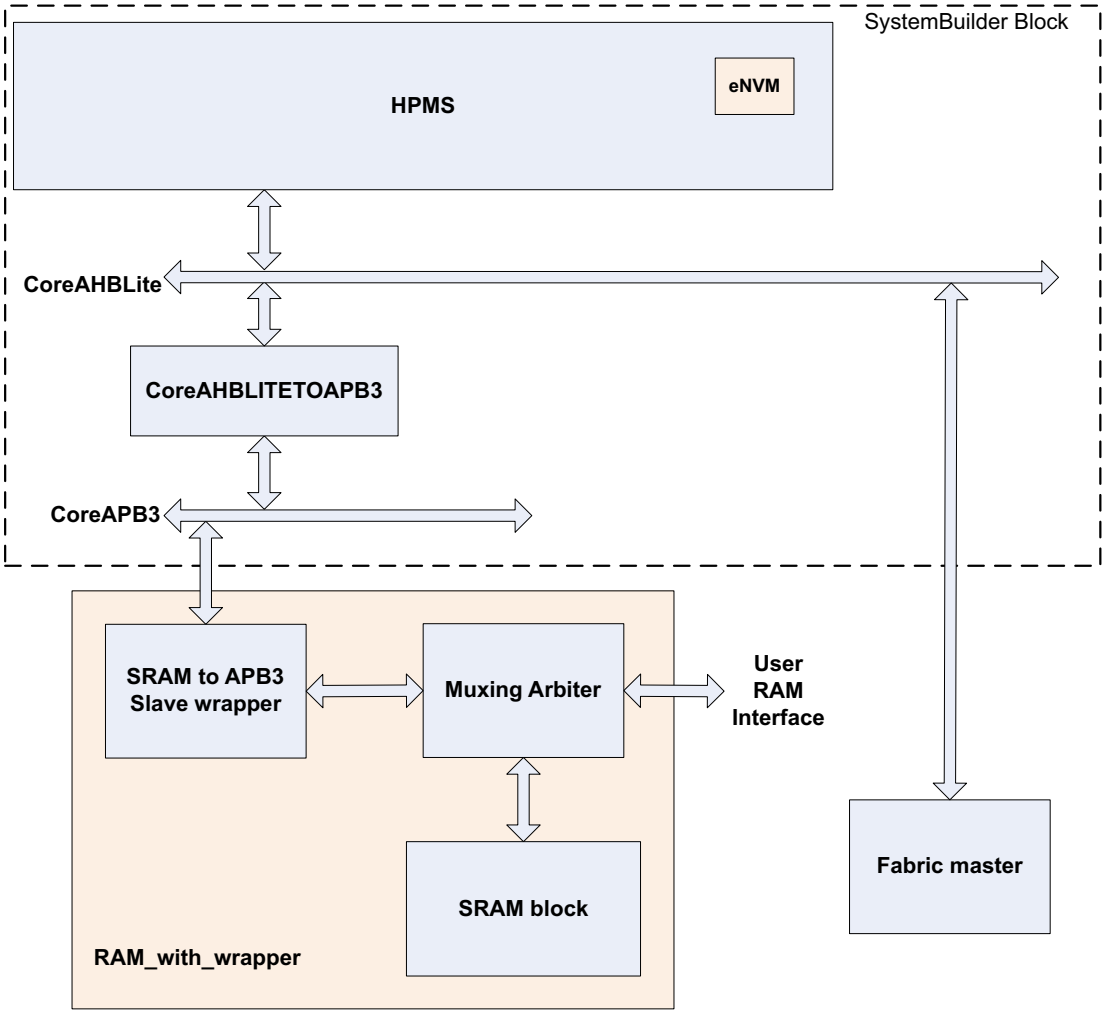

***Figure 2* • Top-Level Block Diagram**

After the fabric master finishes the SRAM initialization, the APB3 wrapper interface asserts a section signal (SEL) to the muxing arbiter and switches the SRAM ports as user ports. It also allows you to read and write to the SRAM blocks. The INIT_DONE output of the reference design indicates the sequence of initialization done.

The SRAM block is configured as two-port memory with a depth of 64-bit and width of 8-bit. This design implements an APB3 slave wrapper interface on port A and port B of the SRAM block, and the APB3 wrapper is connected to HPMS. You can also implement the AHB-lite wrapper instead of APB3 wrapper on the SRAM block and connect to HPMS. However, the APB3 interface is much simpler than the AHB-lite interface, and it is easy to create the design interface in the SRAM ports. The APB3 slave wrapper interface is connected to the HPMS through the CoreAPB3, CoreAHBLITETOAPB3, and CoreAHBLite and fabric interface controller (FIC_0) interface as shown in Figure 2 on page 5. FIC_0 enables the connectivity between the fabric and HPMS, which is a part of HPMS. It performs a bridging functionality between the HPMS and FPGA fabric. It can either be configured in the AHB-lite mode or in the APB3 mode. In this design example, the FIC_0 is configured in the AHB-lite mode so that the other AHB-lite blocks in the fabric can be connected to the HPMS through this FIC_0 in real application.

Fabric Master Block

The fabric master block acts as an AHB-lite master logic to read data from eNVM and write to SRAM. The AHB-lite master drives the address and controls the signals onto the bus after rising the edge of HCLK. If HREADY is in the low state, the AHB-lite master does not move to the next state. If HREADY is in the high state, the AHB-lite master goes to the data phase to perform the read or write operation. During data phase, if HREADY is low (AHB-lite slave wants to extend the data phase), the AHB-lite master must hold the data throughout extended cycles. The AHB-lite master reads or writes the data only after HREADY is in the high state.

Figure 3 shows the simplified state diagram of the fabric master.

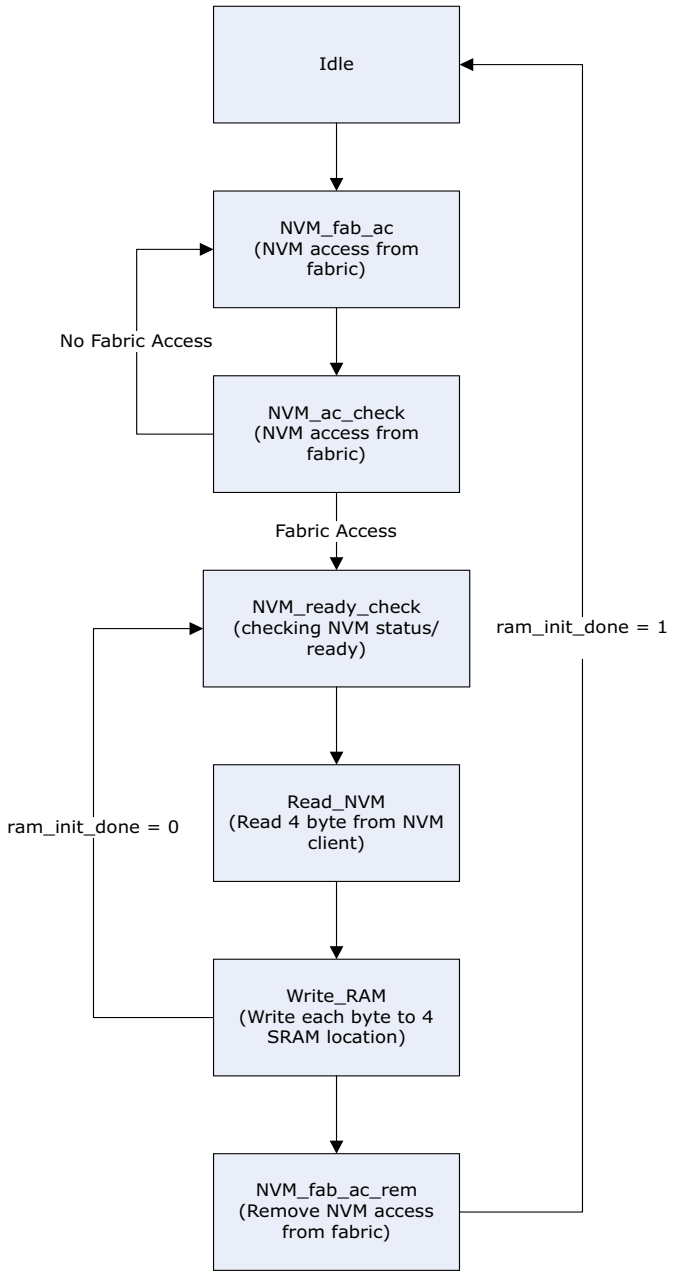

***Figure 3* • Fabric Master State Diagram**

SRAM to APB3 Wrapper

The SRAM to APB3 slave wrapper block allows connecting the SRAM block to the advanced microcontroller bus architecture (AMBA) APB3 bus system. Figure 4 shows the state diagram for APB3 bus system.

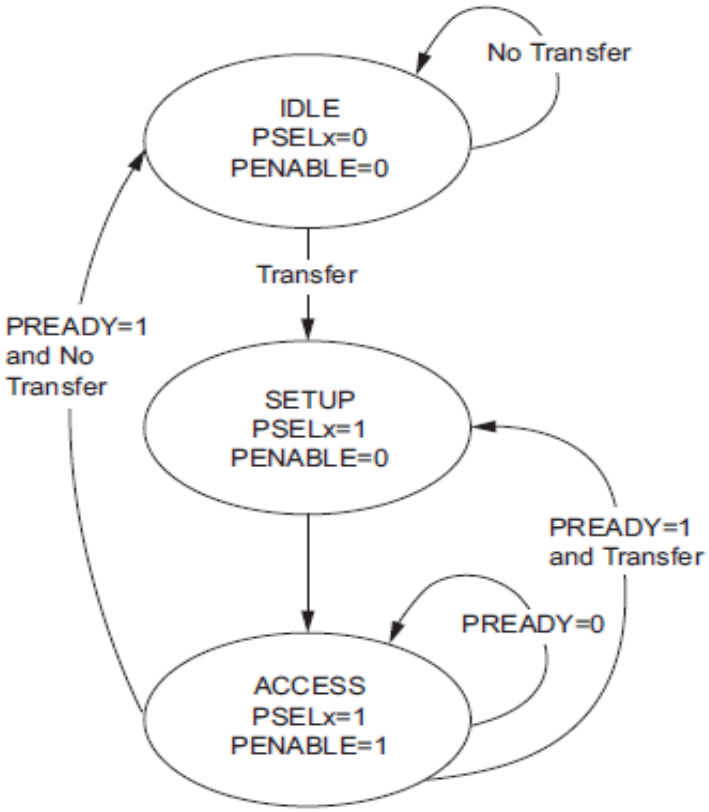

***Figure 4* • APB3 State Diagram**

Following are the three states that explain the APB3 specifications:

- **IDLE**—default state for the peripheral bus.
- **SETUP**—when a transfer is required, the bus moves to this state where the appropriate select signal PSELx is asserted. The bus remains in this state for a complete clock cycle and always moves to the ACCESS state on the next rising edge of the clock.
- **ACCESS**—asserts the PENABLE signal. The address, write and select signals must be stabled during the transition from the SETUP to ACCESS state. The transition from the ACCESS state is controlled by the PREADY signal from the slave.
  - If PREADY is held low by the slave, the peripheral bus remains in the ACCESS state.
  - If PREADY is held high by the slave, no more transitions are required from the ACCESS state to the IDLE state. Alternatively, if another transition follows and the bus moves directly to the SETUP state.

In the above design example, the wrapper logic generates the read and write operations enabled for SRAM using the PSEL, PWRITE, and PENABLE signals. The PREADY signal is used to insert Wait state.

Status Output

The ram_init_done output of the reference design indicates the sequence of initialization done. At power-up, the ram_init_done is asserted as low to indicate the beginning of initialization process. It remains low until the fabric master finishes reading the data from eNVM and writing to SRAM. The high ram_init_done output indicates the ending of initialization process. SRAM interface Port A and Port B are available for read and write operations.

Interface Description for Fabric Master Design

Table 3 shows the top-level interface signal descriptions.

Table 3 • **Top-Level Interface Signal Descriptions**

Signal	Direction	Description
raddr_user[4:0]	Input	User read address
rclk_user	Input	User read clock
rd_enable_user	Input	User read enable
rdata_user[7:0]	Output	User read data
waddr_user	Input	User write address
rdata_user[7:0]	Output	User read data
wclk_user	Input	User write clock
wdata_user[4:0]	Input	User write data
wr_enable_user	Input	User write enable
RESP_err[1:0]	Output	AHB error response
ram_init_done	Output	SRAM initialization complete
DEVRST_N	Input	Active Low reset
SEL	Output	Selection for RAM muxing logic (for debug only)
INT_OUT	Output	Interrupt for APB transaction (for debug only)
ahb_busy	Output	Fabric master status

Hardware Implementation

The SRAM block is configured as two ports memory with a depth of 64-bit and a width of 8-bit in both the design examples. In addition, the design example uses a 50 MHz RC OSC as a reference clock for the fabric phase-locked loop (PLL). The fabric PLL then generates a 100 MHz clock that is used as the main system clock. See "Appendix: Design and Programming Files" section on page 13 to download the design examples.

Simulating Reference Design

The design file includes the testbench files to run simulation in the Libero SoC software.

The simulation waveform has the following sequence:

1. Writing to eNVM Control register to get access.
2. Checking the eNVM status register for NVM ready/busy bit.
3. Reading the eNVM data client.
4. Writing the data to SRAM.

Figure 5 shows the simulation waveform.

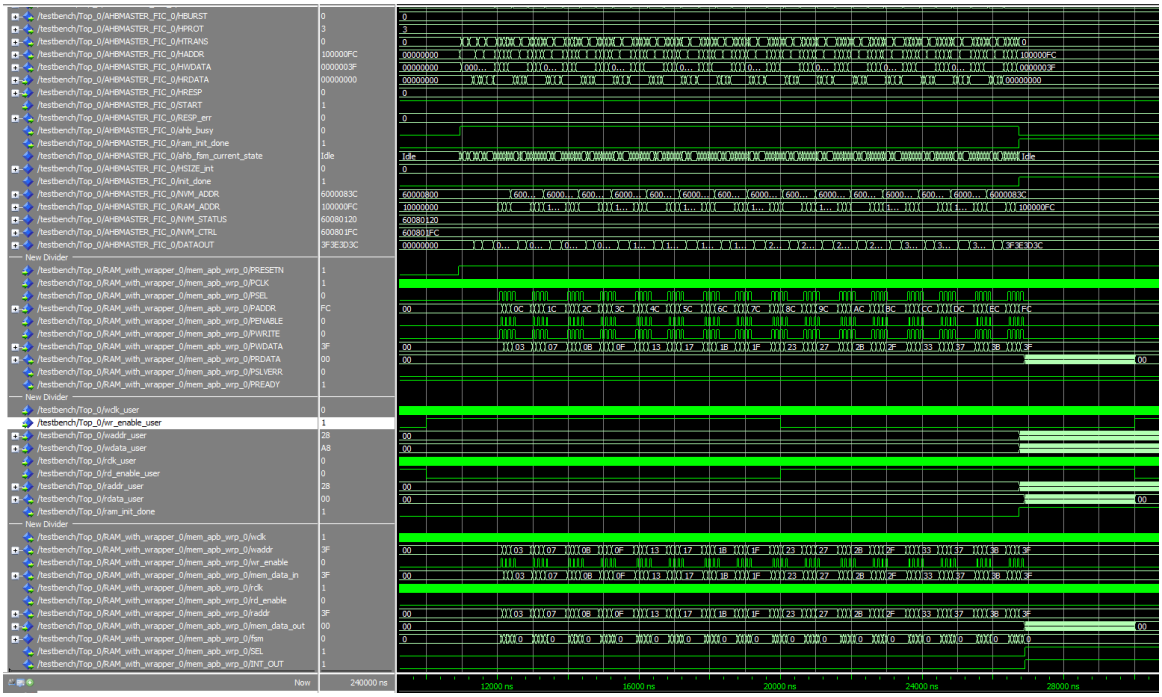

Figure 5 • **Fabric Master Design Example Waveform**

Running the Design

The following procedure describes running the fabric master design example in the IGLOO2 Evaluation Kit board:

1. Open the design example in Libero v11.7 or newer version. Refer to software release notes for updating the design example.
2. Program the IGLOO2 device in IGLOO2 Evaluation Kit with the provided STAPL file using the FlashPro4 and power cycle board.
3. Connect the USB to the host PC. See "Appendix: Design and Programming Files" section on page 13.
4. Open SmartDebug in Libero SoC and look at the SRAM content. The SRAM content must match with a data file that is loaded in eNVM through data client, as shown in Figure 6.

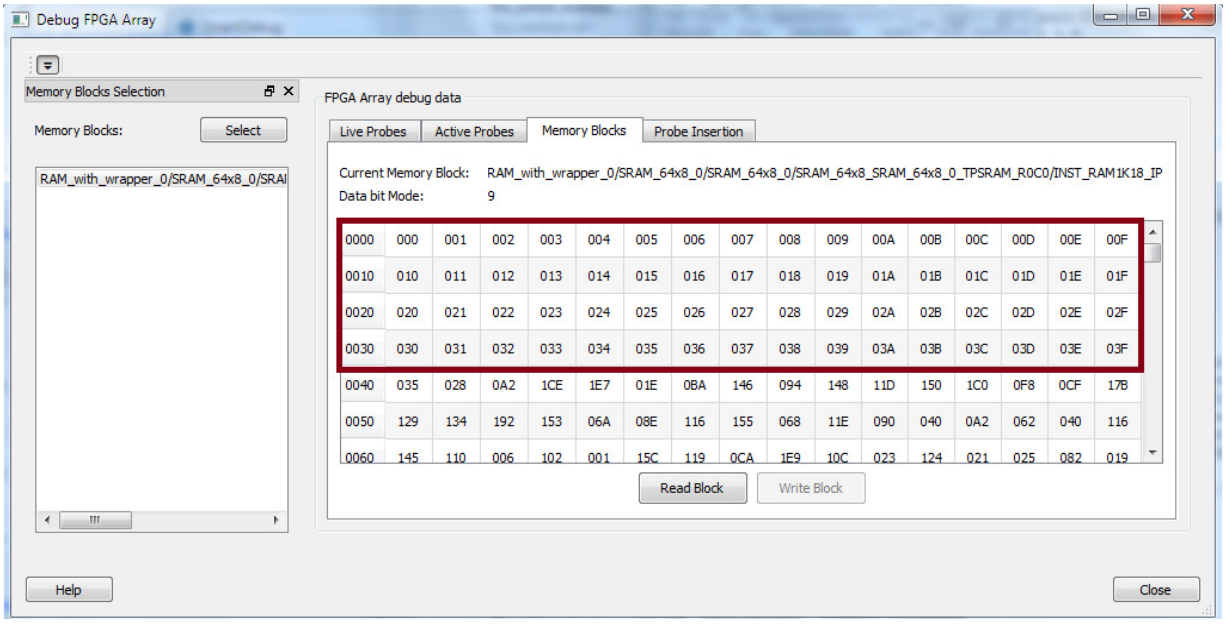

***Figure 6* • HyperTerminal Showing Design Example**

See *the TU0530: SmartFusion2 SoC FPGA and IGLOO2 FPGA SmartDebug Hardware Design Debug Tools Tutorial* for more information on running SmartDebug to view the SRAM content.

Customizing the Wrapper Interface

This section describes how to customize SRAM initialization block. The RAM_with_wrapper block presented in the design example can be modified based on the user SRAM configuration. In addition, the fabric master code needs to be modified based on the user SRAM settings. Figure 7 shows the RAM_with_wrapper block. Following are the three blocks:

- SRAM64x8_0: Two-port SRAM block with depth 64-bit and width 8-bit
- mem_apb_wrp_0: Creates APB3 wrapper on SRAM port
- mux_blk_0: Creates the muxing arbiter

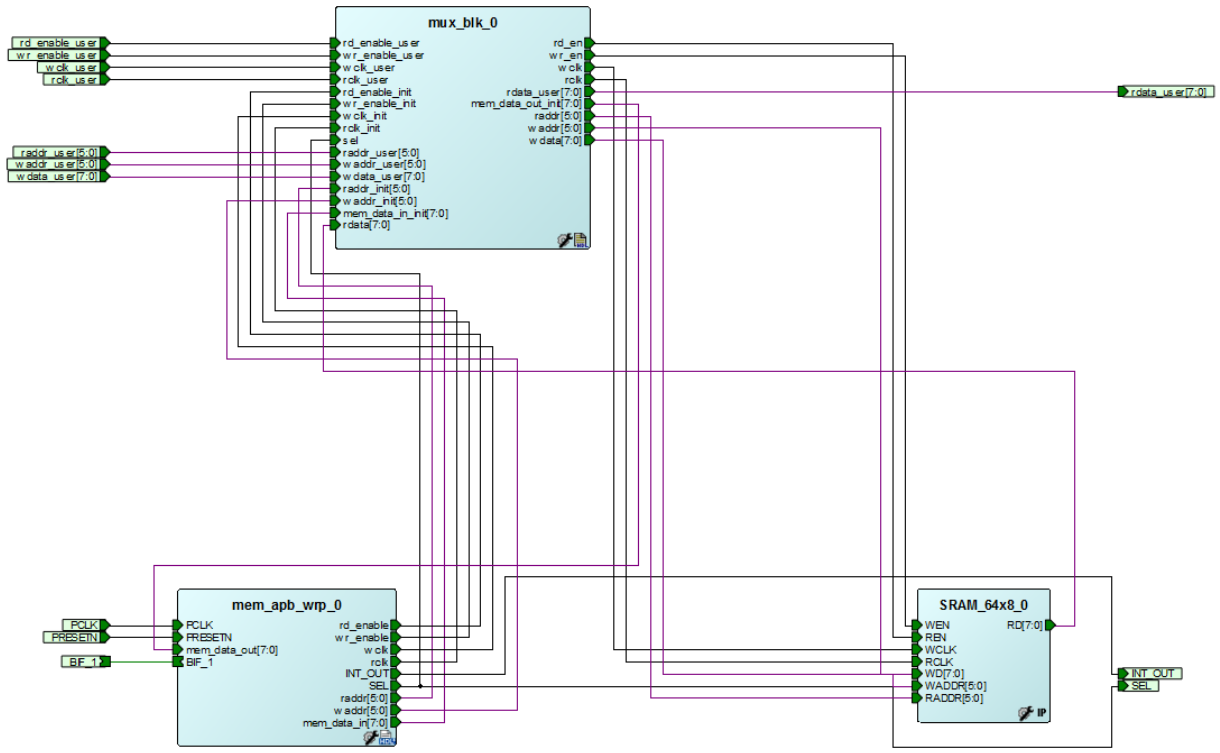

***Figure 7* • RAM_with_wrapper Block**

Depending on the user SRAM block configuration, the SRAM64x8_0 setting needs to be updated. In addition, the DATA_WIDTH and ADDR_WIDTH parameter in mem_apb_wrp and mux_blk file must be modified according to the user design requirement and the blocks must be re-connected, if needed.

Note: The wrapper interface that is used in the design example supports up to 8-bit DATA_WIDTH.

Conclusion

IGLOO2 FPGA devices have embedded SRAM blocks, which allow to store and read the data effectively. The design example shows how the SRAM blocks in IGLOO2 FPGA fabric can be initialized after power-up. It uses an eNVM block to initialize the SRAM after power-up. The eNVM can also be updated using programming, flash loader, or by writing to eNVM, if needed. This application note presents an interface that can be instantiated into the user design, performing the initialization at power-up. The reference design utilizes a very small portion of the FPGA logic for implementation, and does not affect the performance of the main design. The design example initializes a 64 x 8 SRAM block, but the SRAM block can be easily modified to support memory organizations of different width and depth.

Appendix: Design and Programming Files

You can download the design files from Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2gl_ac421_sram_initialization_eNVM_liberov11p7_df

The design file consists of both VHDL and Verilog version of Libero project and programming files (`*.stp`) for IGLOO2 Evaluation Kit board. See the `readme.txt` file included in the design file for the directory structure and description.

List of Changes

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

Date	Changes	Page
Revision 6 (February 2016)	Updated the document for Libero v11.7 software release (SAR 76442).	NA
Revision 5 (February 2015)	Updated the document for Libero v11.5 software release (SAR 64227).	NA
	Updated Figure 1, Figure 5, and Figure 6.	4, 10, and 11
Revision 4 (December, 2014)	Removed all instances of and references to M2GL100 device from Table 2 (SAR 62858).	2
Revision 3 (August, 2014)	Updated the design files link under "Appendix: Design and Programming Files" section.	13
Revision 2 (July, 2014)	Added "Purpose" section.	1
	Placed "References" section after Introduction section.	2
	Updated "Design Requirements" section for Libero v11.4.	2
	Updated the ram_init_done row in Table 3.	9
	Updated the "Appendix: Design and Programming Files" section for VHDL and Verilog version.	13
Revision 1 (April, 2014)	First 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

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 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**.

51900289-6/02.16