AC454
Application Note
RTG4 SRAM Initialization After Power-up Using µPROM

Microsemi
a Microchip company

a MICROCHIP company

Microsemi 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
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), 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. Learn more at www.microsemi.com.

51900454. 6.0 11/21

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 6.0

The following is a summary of the changes made in this revision.

- Updated the design and document for Libero SoC v2021.2.
- Updated Figure 1, page 3.
- Updated Hardware Implementation, page 9.
- Updated Clocking Structure, page 9.
- Updated Reset Structure, page 10.

1.2 Revision 5.0

The following is a summary of the changes made in this revision.

- Added Setting Up the Demo Design, page 12.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 14.
- Added Appendix 2: Running the TCL Script, page 17.
- Removed the references to Libero version numbers.

1.3 Revision 4.0

The document was updated for Libero SoC v11.9 SP1.

1.4 Revision 3.0

The document was updated for Libero SoC v11.8 SP2.

1.5 Revision 2.0

The procedure to reset RAM block contents using µPROM was added. For more information, refer to Appendix 5: How to Reset RAM Block Contents Using µPROM, page 20.

1.6 Revision 1.0

The first publication of this document.

2 RTG4 SRAM Initialization After Power-up Using µPROM

This application note describes how to initialize the static random access memory (SRAM) blocks of Microsemi RTG4™ field programmable gate array (FPGA) with user data after power-up. The design for this application note uses a large SRAM (LSRAM) block, which is initialized by an FPGA fabric master through the Advanced Microcontroller Bus Architecture Advanced Peripheral Bus interface (AMBA APB bus).

RTG4 FPGA devices have embedded SRAM blocks (LSRAM and micro SRAM (µSRAM)) in the fabric. Both LSRAM and µSRAM blocks are placed in multiple rows within the FPGA fabric, and they can be accessed through the fabric routing architecture. Table 2, page 4 lists the number of LSRAM, µSRAM, and micro programmable read-only memory (µPROM) blocks available in the RT4G150 device.

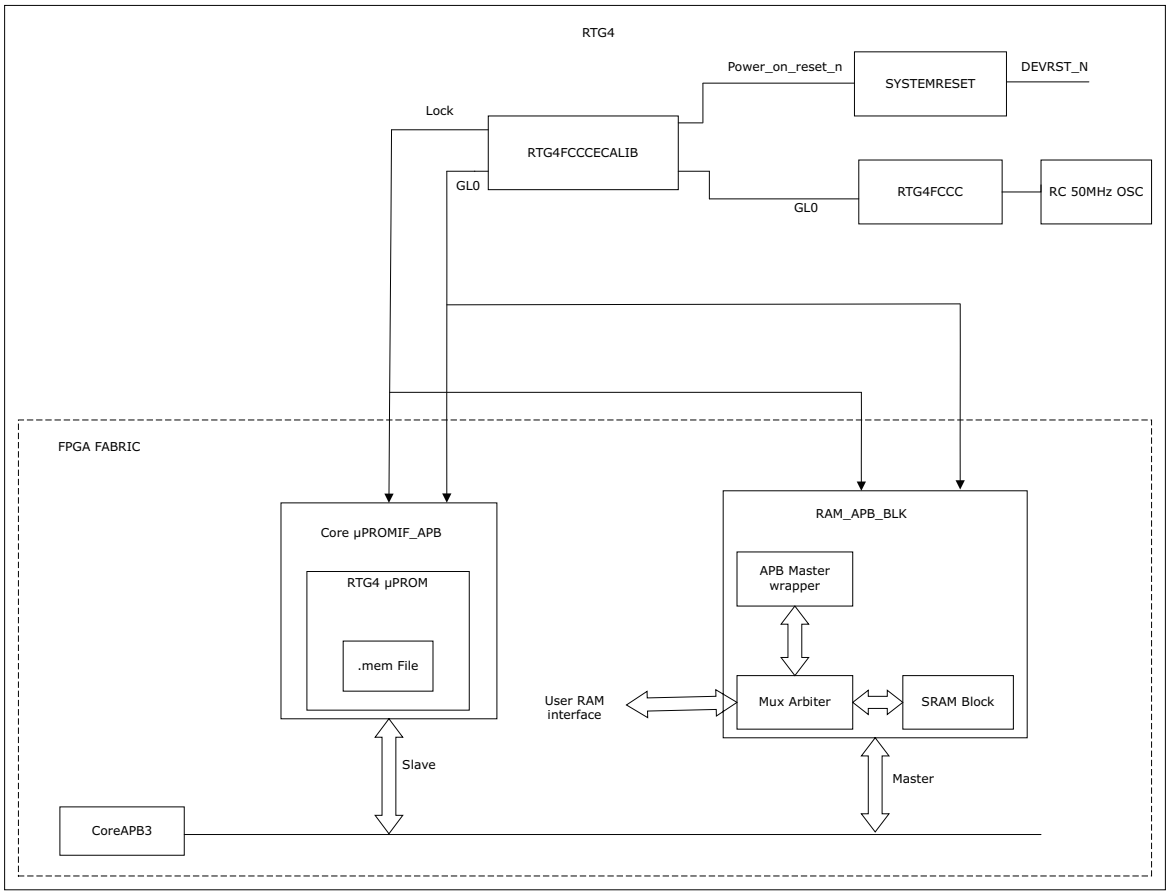
LSRAMs are used for larger data storage (up to 24,576 bits), whereas µSRAMs are used for smaller data storage (up to 1536 bits). Both LSRAM and µSRAM are volatile. As a result, data is lost after the device power-down. After power-up, the state of SRAM is unknown.

µPROM can be used to store programmable data for initializing the LSRAM and µSRAM blocks. µPROM cells are located at the bottom of the FPGA fabric, and they can be accessed through the fabric interface. This application note implements a design that uses µPROM to initialize the LSRAM block. µPROM stores up to 10,400 36-bit words (374,400 bits of data). It supports only read operations during normal device operation after the device is programmed. For more information about µPROM features and architecture, refer to the *µPROM* section in *UG0574: RTG4 FPGA Fabric User Guide*.

Figure 1, page 3 shows the top-level block diagram of the design used in this application note.

The LSRAM initialization data (`.mem` file) is stored in the µPROM during programming. The `.mem` file is a data storage client that contains the µPROM memory content. The fabric APB bus master wrapper reads the µPROM and stores the initialization data in LSRAM. The CoreµPROMIF_APB IP core implements the APB bus slave wrapper logic to provide read-only access to the µPROM. This design is simulated and validated using the RTG4 Development Kit.

Figure 1 • **Top-Level Block Diagram**



2.1 Design Requirements

The following table lists the hardware and software requirements for this demo design.

Table 1 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 Development Kit • 12 V adapter (provided with the kit)	Rev B or later
Host PC or laptop	64-bit Windows 7 and 10
Software Requirements	
Libero® System-on-Chip (SoC)	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	
Host PC drivers	USB to UART drivers

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

Before you start:

1. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location: *https://www.microsemi.com/product-directory/design-resources/1750-libero-soc*
2. For demo design files download link:

   http://soc.microsemi.com/download/rsc/?f=rtg4_ac454_df

2.3 Embedded SRAM and µPROM Blocks

The following table lists the number of SRAM blocks (including LSRAM and µSRAM) and µPROM blocks available in the RT4G150 device.

Table 2 • **SRAM and µPROM Blocks in the RT4G150 Device**

Type	Number of blocks in RT4G150
LSRAM 24.5 Kb blocks	209
µSRAM 1.5 Kb blocks	210
µPROM 381 Kb block	1

LSRAM is configured in a two-port mode in this design. One port is dedicated for write operations and the other for read operations. The read and write operations are synchronous and require a clock edge. LSRAM supports both pipelined read and non-pipelined read (flow-through) operations. µSRAM blocks have two read data ports (port A and port B) and one write data port (port C). Read operations are executed in synchronous and asynchronous modes. Write operation is performed only in synchronous mode.

µPROM supports only read operation during normal device operation. Read operation is performed using the fabric interface in synchronous mode only.

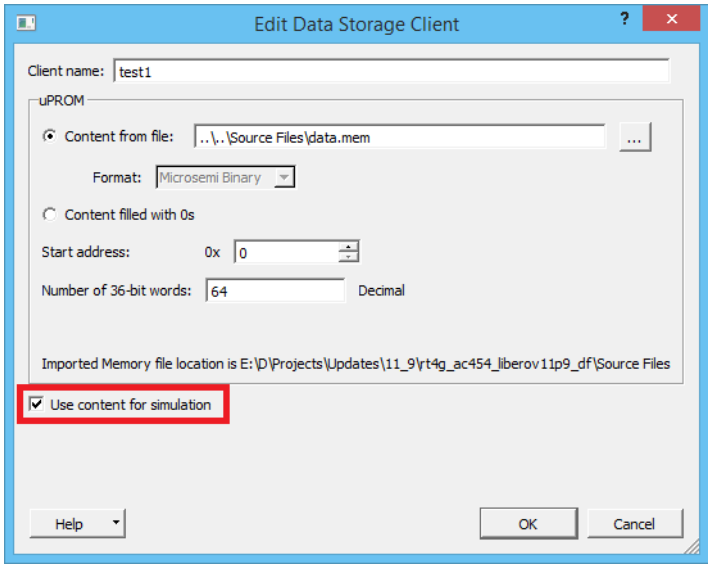
For more information about the features and use models of LSRAM, µSRAM, and µPROM, refer to *UG0574: RTG4 FPGA Fabric User Guide*.

2.4 Data Storage in RTG4 µPROM

This design uses the µPROM memory content for initializing the LSRAM block after power-up. CoreµPROMIF_APB IP provides easy access to µPROM memory for APB bus masters. It performs address translation, allowing APB bus masters to address µPROM using word-aligned addressing. To access µPROM memory using CoreµPROMIF_APB, RTG4µPROM core is instantiated along with CoreµPROMIF_APB in SmartDesign. These cores are available in the Libero SoC Catalog. For more information about instantiating CoreµPROMIF_APB and mapping RTG4µPROM, refer to the *System Integration* section in *CoreUPROMIF_APB_HB.pdf*.

In this design, a data storage client is created to store LSRAM initialization data. It is configured to store 36-bit words in 64 locations during the configuration of µPROM. The data storage client file is provided with the design file (refer to Appendix 3: Design Files, page 18). The RTG4µPROM core memory configurator GUI in Libero SoC is used for creating multiple data storage clients. The following figure shows how to create a single data storage client for this design. To allow µPROM content for simulation, select the **Use content for simulation** check box.

Figure 2 • **µPROM Memory Configurator GUI**



2.5 Design Description

This design includes:

- Fabric APB Master, page 5
- CoreµPROMIF_APB, page 8

2.5.1 Fabric APB Master

The fabric master performs the following functions:

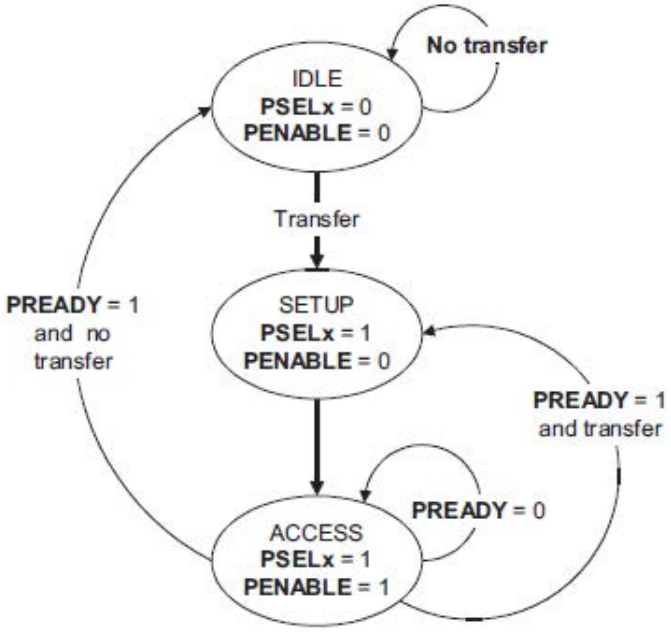
1. Acts as AMBA Advanced Peripheral Bus 3 (APB3) bus interface that reads data from the µPROM memory using CoreµPROMIF_APB slave interface.
2. Loads data into LSRAM via the APB bus I/F.

The state machine operates through the following states of an APB bus cycle:

- **IDLE:** This is the default state of the APB bus.
- **SETUP:** When a transfer is required the state machine moves to this state. In this state, the required PSEL signal is asserted. The APB bus remains in this state for one clock cycle and moves to the ACCESS state on the rising edge of PCLK.
- **ACCESS:** The PENABLE signal is asserted in this state. The address, write, and select signals are asserted and must remain stable during the transition from SETUP to ACCESS state. The PREADY signal controls the exit from this state in the following ways:
  - If **PREADY** is held low by the slave, the state machine remains in this state.
  - If **PREADY** is held high by the slave, this state is exited and moved back to the IDLE state if no more transfers happen. If another transition happens, the state machine moves to the SETUP state.

The following illustration shows the states in the APB bus cycle.

Figure 3 • **APB3 State Diagram**



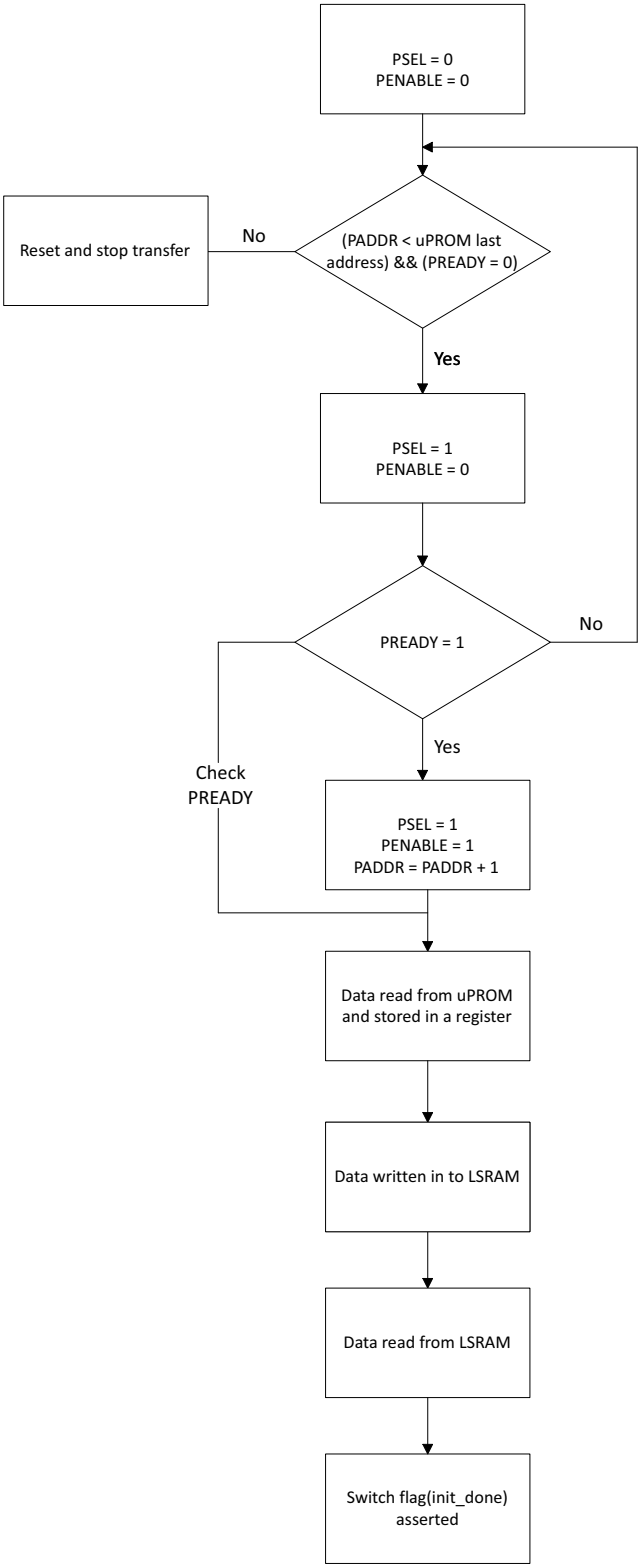
After the fabric master interface reads data from µPROM memory and loads that data into fabric LSRAM, it asserts a switch flag (init_done) to the mux arbiter block. This block lets the LSRAM ports to be used for initialization on design startup and then releases them for user access once initialization is complete.

The LSRAM block is configured as a two-port with a depth of 512 and a width of 36. The APB3 master wrapper logic generates the required read and write operations for LSRAM using the PREADY signal from the slave interface. The PREADY signal is also used for inserting wait states.

The APB fabric master interface generates address and controls the signals on the bus after the rising edge of PCLK. If PREADY is HIGH, the APB master enters the data phase to perform a read or write operation. During the data phase, if PREADY is LOW, then the APB slave extends the data phase. The APB fabric master must hold the data throughout extended cycles. The APB master will only read and write the APB slaves when PREADY is HIGH.

The following illustration shows the states of the fabric master interface.

Figure 4 • **Fabric Master State Diagram**



2.5.1.1 Fabric APB Master Interface Description

The following table lists the fabric APB master interface signals.

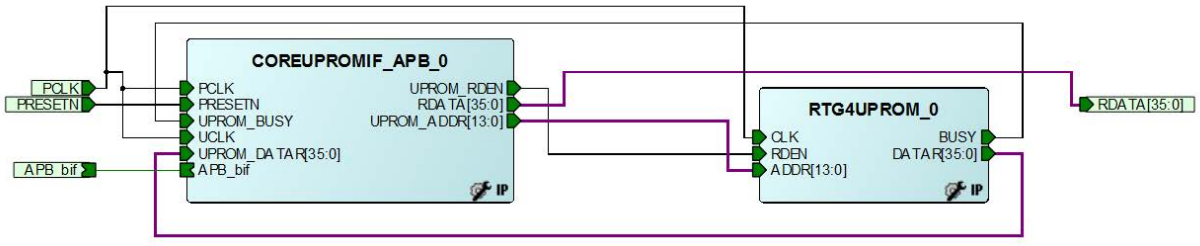
Table 3 • **Fabric APB Master Interface Signals**

Signal	Direction	Description
PCLK	Input	APB bus clock
PRESETn	Input	APB bus active low reset
PRDATA[31:0]	Input	APB bus input 32 bit read data
RDATA[35:0]	Input	36 bit µPROM read data
PREADY	Input	APB bus ready signal
PSLVERR	Input	APB bus error reporting signal
PWRITE	Output	APB bus write access signal
PWDATA[31:0]	Output	APB bus 32 bit wide write data
PENABLE	Output	APB bus enable signal
PSEL	Output	APB bus slave select signal
PADDR[15:2]	Output	APB bus slave address signal
Mem_data_out[35:0]	Output	Output data for LSRAM
rd_en	Output	LSRAM read enable signal
wr_en	Output	LSRAM write enable signal
raddr[8:0]	Output	LSRAM read address signal
waddr[8:0]	Output	LSRAM write address signal
Init_done	Output	Initialization done signal

2.5.2 CoreµPROMIF_APB

RTG4 µPROM is accessed using RTG4 µPROM core or by using CoreµPROMIF_APB IP core. The CoreµPROMIF_APB IP core makes the RTG4 µPROM block appear as a transparent memory on the APB bus interface. In this design, the CoreµPROMIF_APB IP core is used to perform address translation to allow APB bus masters to directly address µPROM using word aligned addressing. The CoreµPROMIF_APB IP core also prevents reading an invalid address space in µPROM. The CoreµPROMIF_APB IP core must be instantiated along with the RTG4µPROM core and connected as shown in the following figure.

Figure 5 • **CoreµPROMIF_APB and RTG4µPROM SmartDesign Connection**

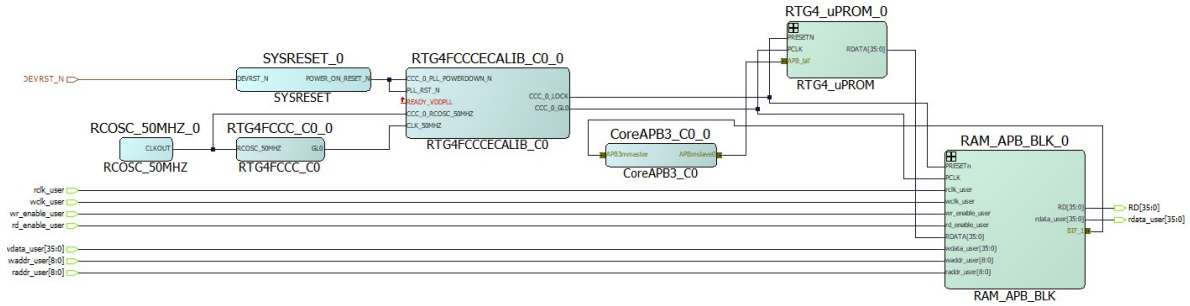


For more information about other features and advantages of using CoreµPROMIF_APB, refer to *CoreUPROMIF_APB_HB.pdf*.

2.6 Hardware Implementation

This design uses the Sysreset signal, a 50 MHz RC oscillator, the RTG4 fabric clock conditioning circuit (FCCC), the RAM_APB_BLK_0 block (LSRAM block with master wrapper), and the CoreµPROMIF_APB and RTG4µPROM IP cores, as shown in the following figure. The IP cores, along with the LSRAM wrapper, are used to initialize the fabric SRAM by moving data from µPROM to fabric LSRAM via the APB interface. This design uses the 50 MHz RC oscillator as a reference clock for the fabric CCC. The fabric CCC is connected to RTG4FCCCECALIB which generates a 30 MHz clock, which is used as the system clock.

Figure 6 • **SmartDesign Top-Level Diagram**



The RAM_APB_BLK_0 module contains LSRAM as a two-port memory with the depth and width configured as 512 × 36. This module can be modified to initialize LSRAM blocks configured in any of the various aspect ratios and operating modes supported by RTG4. For more information about using variations of LSRAM, refer to Appendix 4: Customizing RAM Wrapper Interface, page 19.

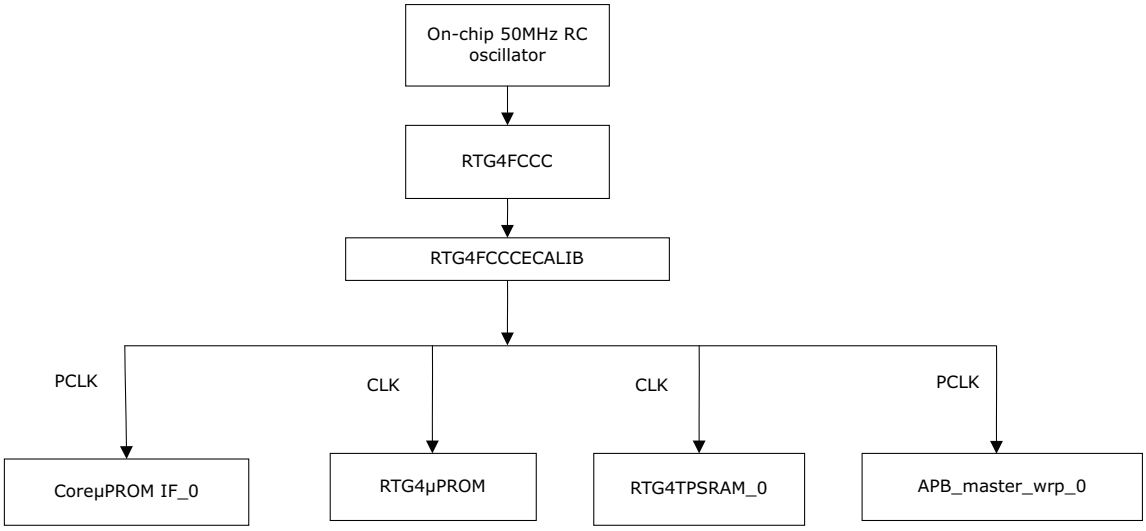
The RTG4µPROM core imposes a maximum frequency constraint of 30 MHz on the µPROM clock. CoreµPROMIF_APB has pre-scalar clock logic that generates µPROM clock from PCLK frequency with the condition of µPROM clock frequency not exceeding 30 MHz. In this design, the µPROM read operation frequency is 30 MHz.

For more information about operating µPROM at a clock frequency other than 30 MHz, refer to the *Design Constraints* section in *CoreUPROMIF_APB_HB.pdf*.

2.7 Clocking Structure

The on-chip 50 MHz oscillator gives the reference frequency to RTG4FCCC_0,which is connected to RTG4FCCCECALIB generates a 30 MHz clock (GL0) and drives COREUPROMIF_0, RTG4UPROM, RTG4TPSRAM_0, and APB_master_wrp_0 blocks. The following figure shows the reset structure of the design.

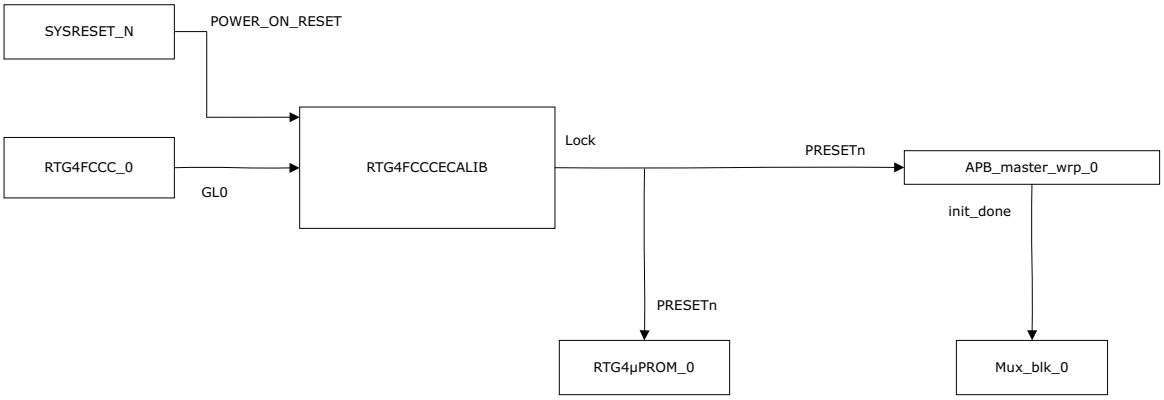
Figure 7 • **Clocking Structure**



2.8 Reset Structure

The POWER_ON_RESET and GL0 signals are given to RTG4FCCCECALIB core and the output signal is used to reset the RTG4uPROM_0 and APB_master_wrp_0 block. After reset, the APB_master_wrp_0 block generates the init_done signal to reset the mux_blk_0. The following figure shows the reset structure of the design.

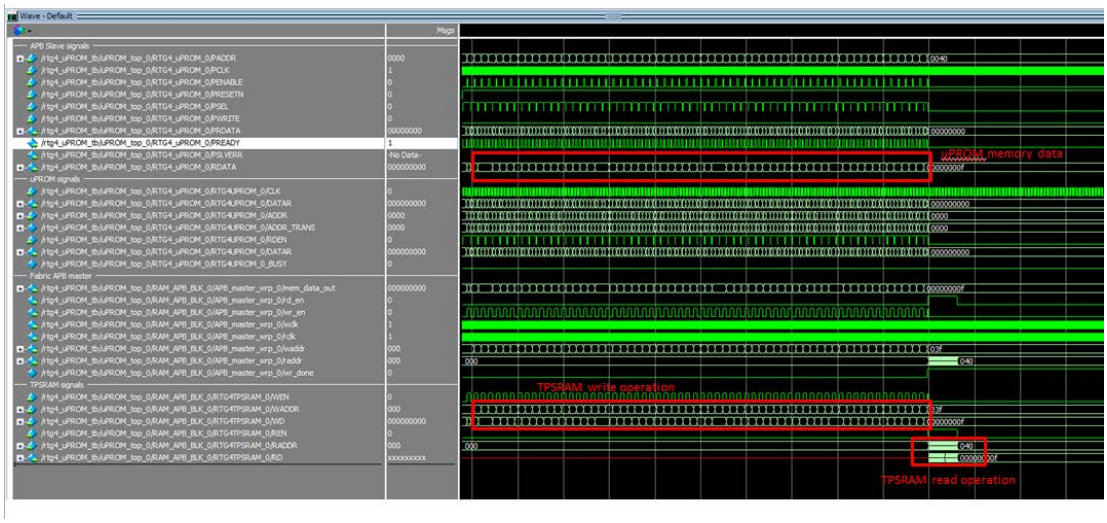
Figure 8 • **Reset Structure**



2.9 Simulating the Design

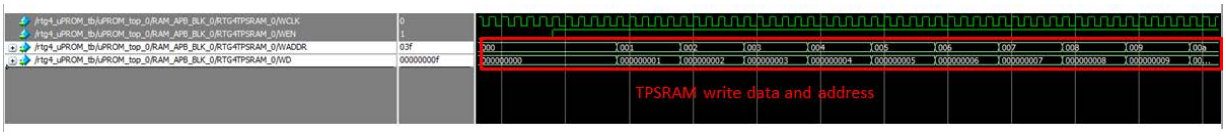
The design files mentioned in Appendix 3: Design Files, page 18 include testbench files that are required for simulating the design. As shown in the following figure, data is read from µPROM using APB interface and loaded to LSRAM. This data can be seen using the RD[35:0] output signal of the TPSRAM block (highlighted in the figure).

Figure 9 • **Waveform of RTG4 LSRAM Initialization Using µPROM**



The following figure shows the write data and write address of TPSRAM.

Figure 10 • **TPSRAM Write Data and Write Address**



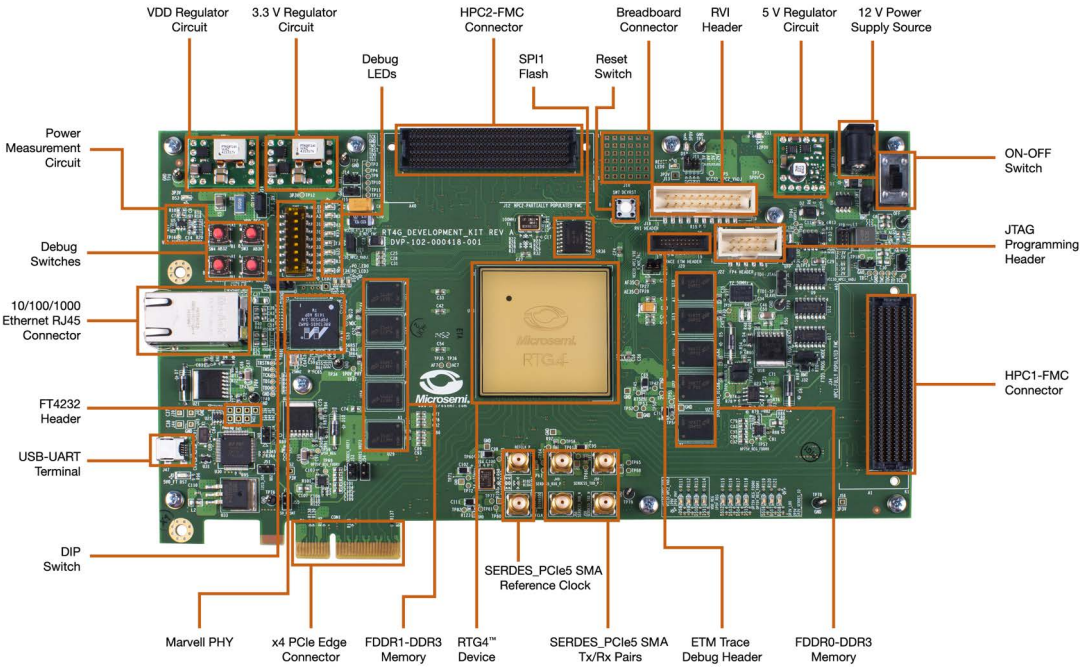
The following figure shows the read data and read address of TPSRAM.

Figure 11 • **TPSRAM Read Data and Read Address**

2.10 Setting Up the Demo Design

The following figure shows the RTG4 Development Kit board.

Figure 12 • **RTG4 Development Kit Board**



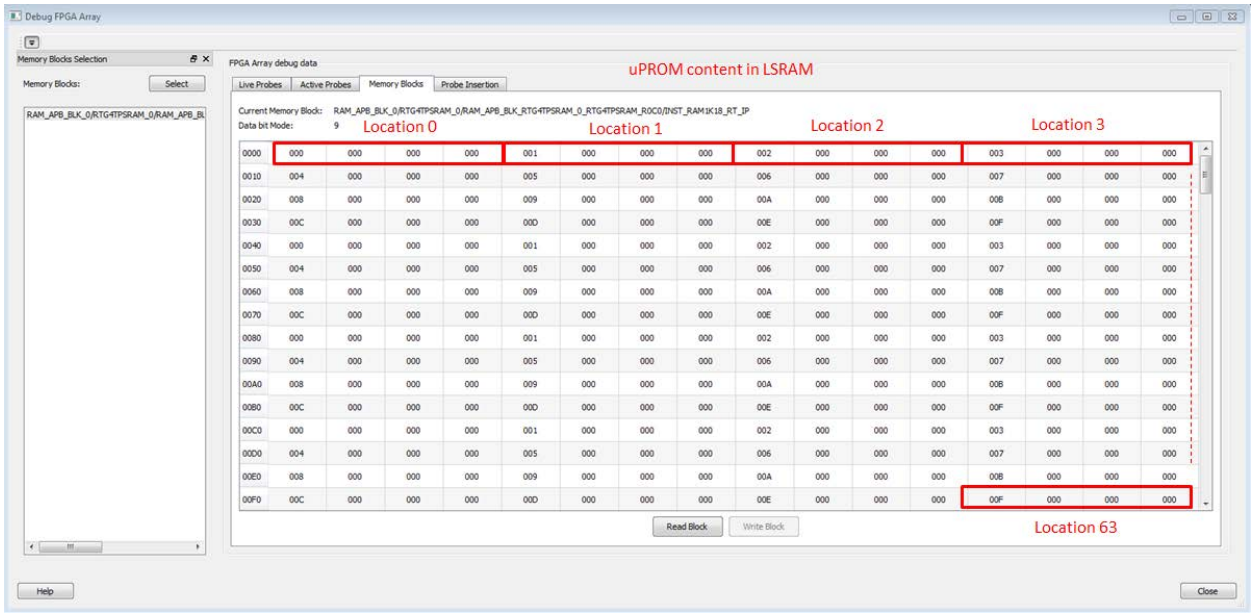
2.11 Running the Design

To program the RTG4 Development Kit with the job file provided as part of the design files using FlashPro Express software, refer to Appendix 1: Programming the Device Using FlashPro Express, page 14.

After the RTG4 Development kit is programmed successfully, open SmartDebug in Libero SoC and check for the LSRAM content. The LSRAM content must match with the memory file that is loaded into µPROM through data client configurator, as shown in the following figure.

For more information about running SmartDebug to view LSRAM memory block content, refer to *TU0530: SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools Tutorial*.

Figure 13 • **SmartDebug LSRAM Read Data**



2.12 Conclusion

This application note describes how to initialize the RTG4 FPGA LSRAM with user data programmed into the µPROM. It provides an interface that can be instantiated in a user design to perform LSRAM initialization using µPROM. It also explains how to simulate and validate the design on RTG4 Development Board.

3 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the programming job file using FlashPro Express.

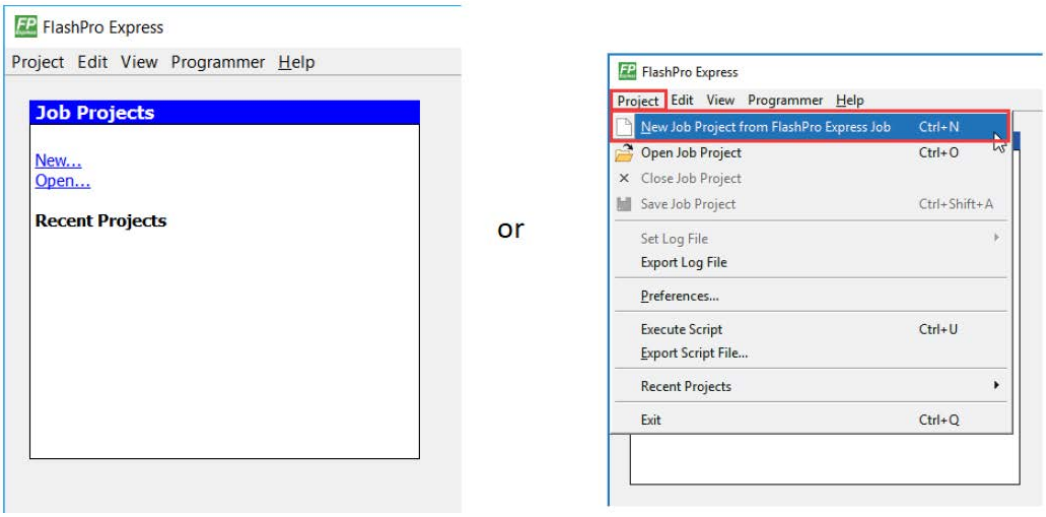
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in *Table 3 of UG0617: RTG4 Development Kit User Guide*.
2. Optionally, jumper **J32** can be set to connect pins 2-3 when using an external FlashPro4, FlashPro5, or FlashPro6 programmer instead of the default jumper setting to use the embedded FlashPro5.

Note: The power supply switch, **SW6** must be switched **OFF** while making the jumper connections.

3. Connect the power supply cable to the **J9** connector on the board.
4. Power **ON** the power supply switch **SW6**.
5. If using the embedded FlashPro5, connect the USB cable to connector **J47** and the host PC. Alternatively, if using an external programmer, connect the ribbon cable to the JTAG header **J22** and connect the programmer to the host PC.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 14 • **FlashPro Express Job Project**



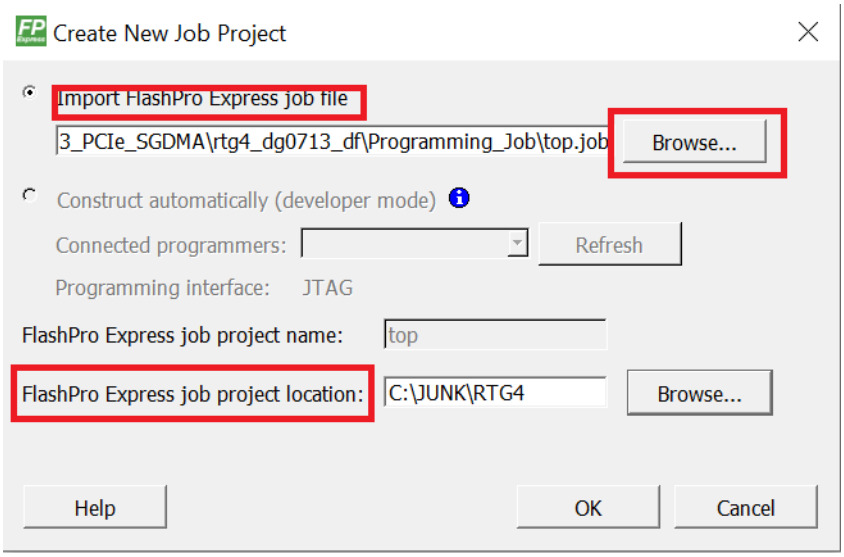
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:

  `<download_folder>\rtg4_ac454_df\Programming_Job`

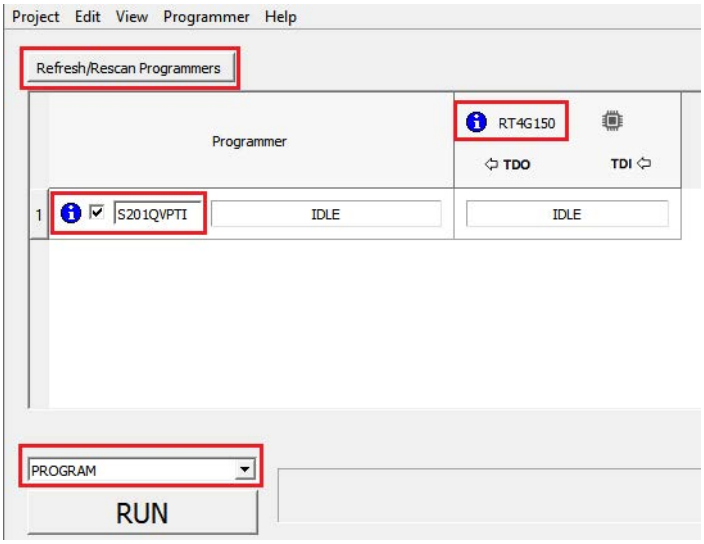
- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 15 • **New Job Project from FlashPro Express Job**



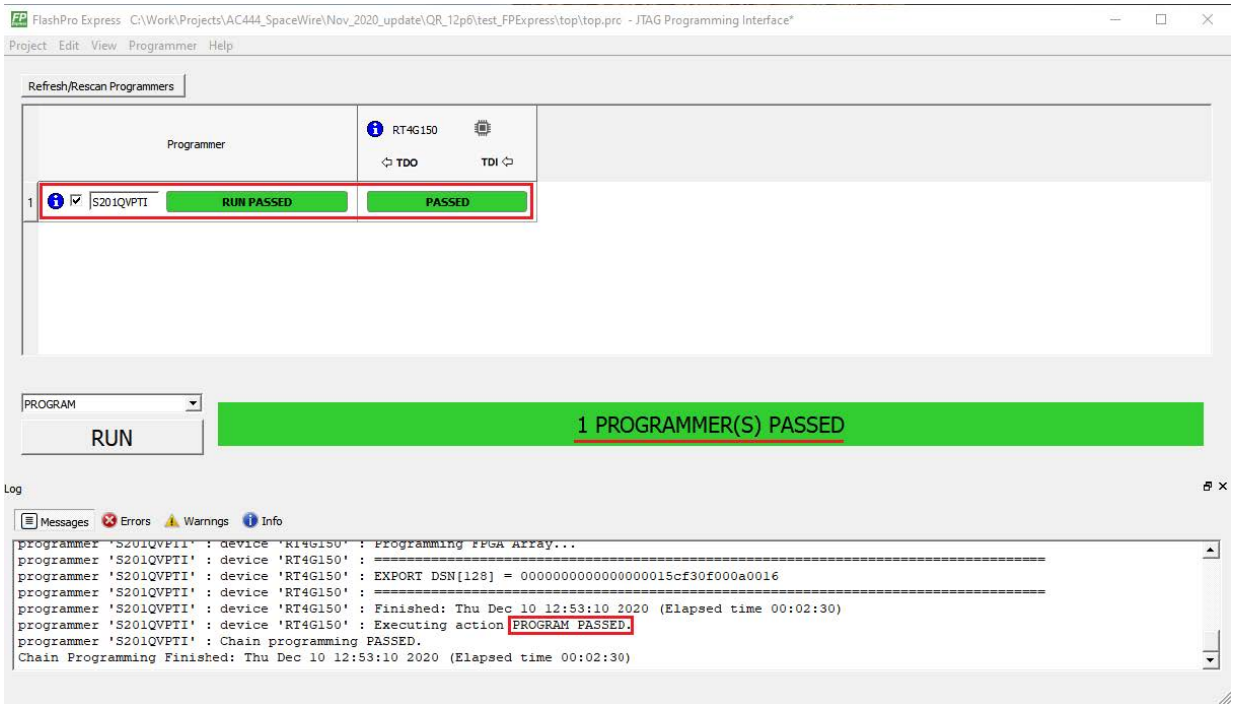
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 16 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 17 • **FlashPro Express—RUN PASSED**



12. Close **FlashPro Express** or click **Exit** in the Project tab.

4 Appendix 2: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded TCL_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to **rtg4_ac454_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.

5 Appendix 3: Design Files

You can download the design files from the following location on the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=rtg4_ac454_df

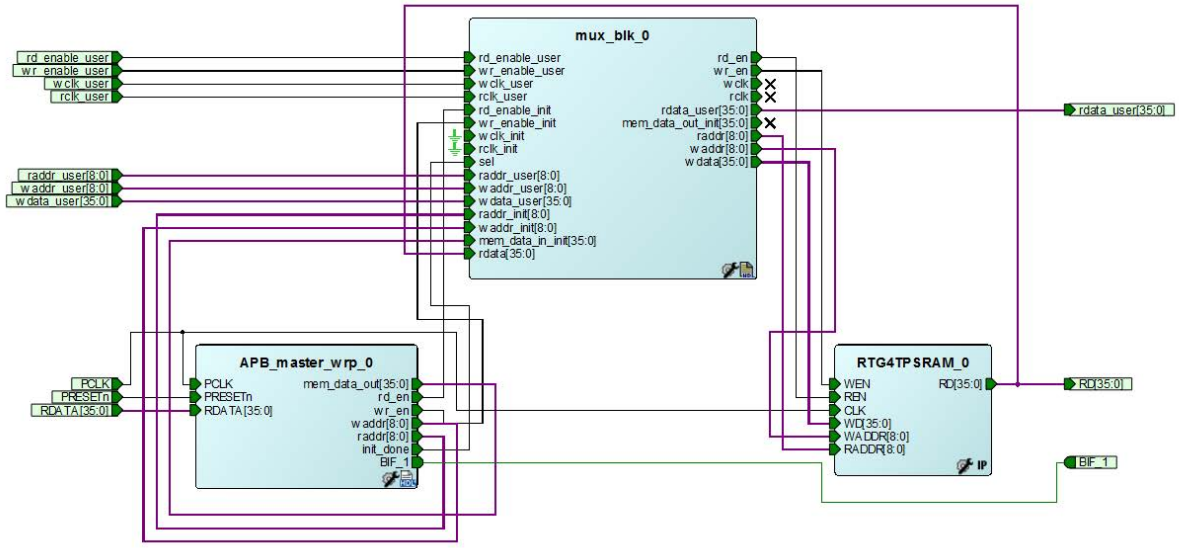
The design files consist of a Verilog version of Libero project folder, source file (data storage memory client), and programming file (`*.job`) for RTG4 Development Kit. Refer to the `readme.txt` file included in the design files for the directory structure and description.

6 Appendix 4: Customizing RAM Wrapper Interface

This section describes how to customize a RAM wrapper interface according to the LSRAM variation configuration. The Fabric APB master code needs to be modified to support other LSRAM configurations that are different from those shown in this application note. The following figure shows the RAM APB wrapper block, which consists of the following blocks:

- **RTG4TPSRAM_0:** RTG4 LSRAM configured as two-port mode with 512 depth and 36 width
- **APB_master_wrp_0:** Fabric APB master interface
- **Mux_blk_0:** mux arbiter block to switch SRAM ports

Figure 18 • **APB Master Wrapper SmartDesign**



The RTG4TPSRAM_0 setting must be updated based on the specific variation used. Also, the fabric APB master RTL code must be modified, changing the DATA_WIDTH and ADDR_WIDTH parameters as necessary. After the modifications are made, SmartDesign must be connected and regenerated. This design supports a data width of 36.

7 Appendix 5: How to Reset RAM Block Contents Using µPROM

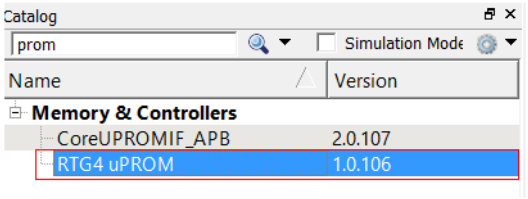
In Libero SoC, the RTG4 uPROM Configurator allows the addition of a client for resetting the contents of all RAM blocks. At device power-up or when the DEVRST_N signal goes active, the RAM initialization client initializes all RAM blocks to zero. After device power-up, the data read from all µSRAM and LSRAM address locations is zero until the RAM blocks are written to.

Note: When you read the initial zero from an address in a RAM block that has ECC enabled, its SB_CORRECT and DB_DETECT flags get asserted. The flags for any given RAM address location reset once that address location is written to.

The following steps describe how to initialize RAM blocks using UPROM Configurator in Libero SoC. The design must consist of RTG4 dual-port LSRAM, two-port LSRAM, or micro SRAM blocks.

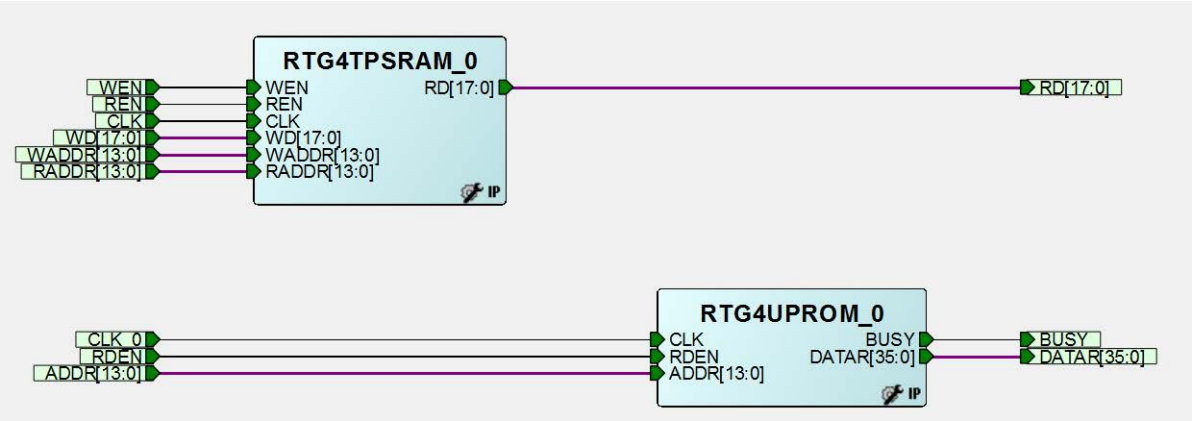
1. From the Libero Catalog, select the RTG4 uPROM macro, as shown in the following figure, and drag it on to the SmartDesign canvas.

Figure 19 • **Catalog Window**



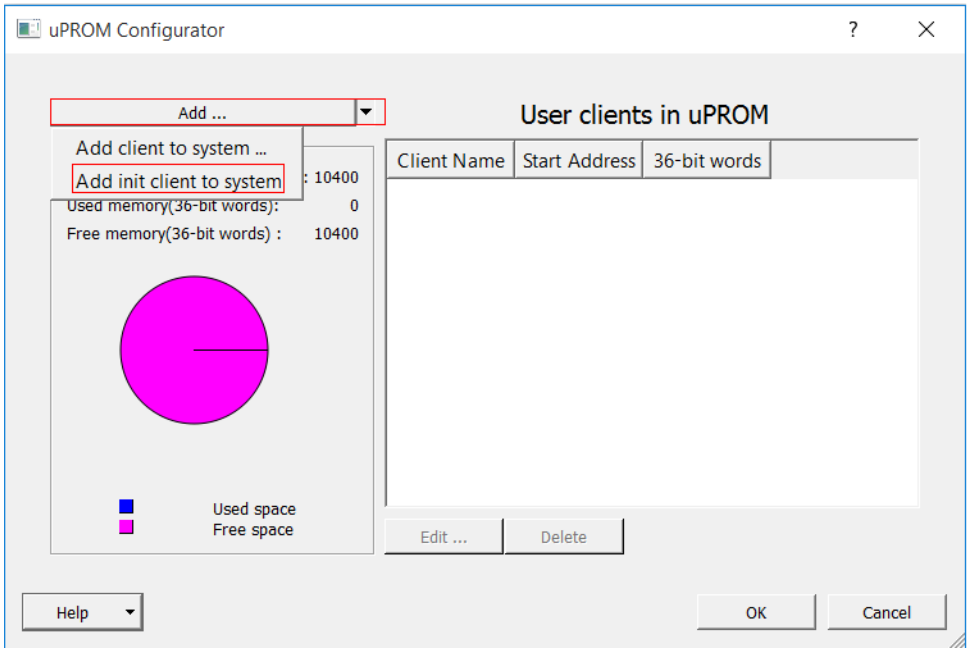
The following figure shows a sample SmartDesign block consisting of the RTG4 µPROM Configurator and a two-port large SRAM instance configured as 16 1K × 18.

Figure 20 • **Sample SmartDesign Block**



2. Open the **uPROM Configurator**, click **Add …**, and select **Add init client to system**, as shown in the following figure.

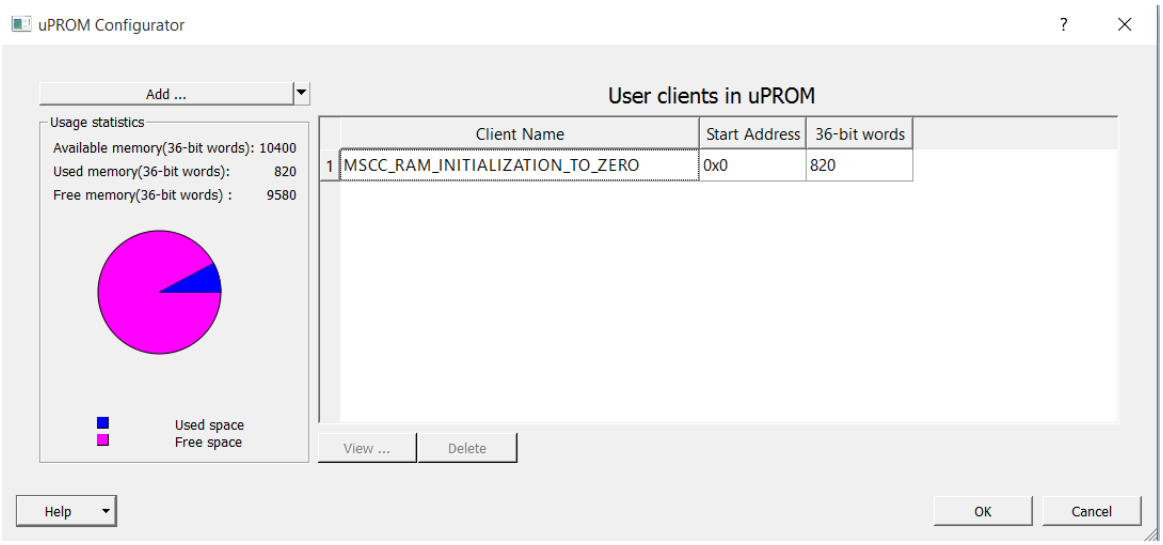
Figure 21 • **uPROM Configurator**



The read only data client **MSCC_RAM_INITIALIZATION_TO_ZERO** gets added at address location 0x0, as shown in the following figure. This initializes all RAMs in the device using broadcast feature.

Note: This client cannot be edited, but it can be deleted.

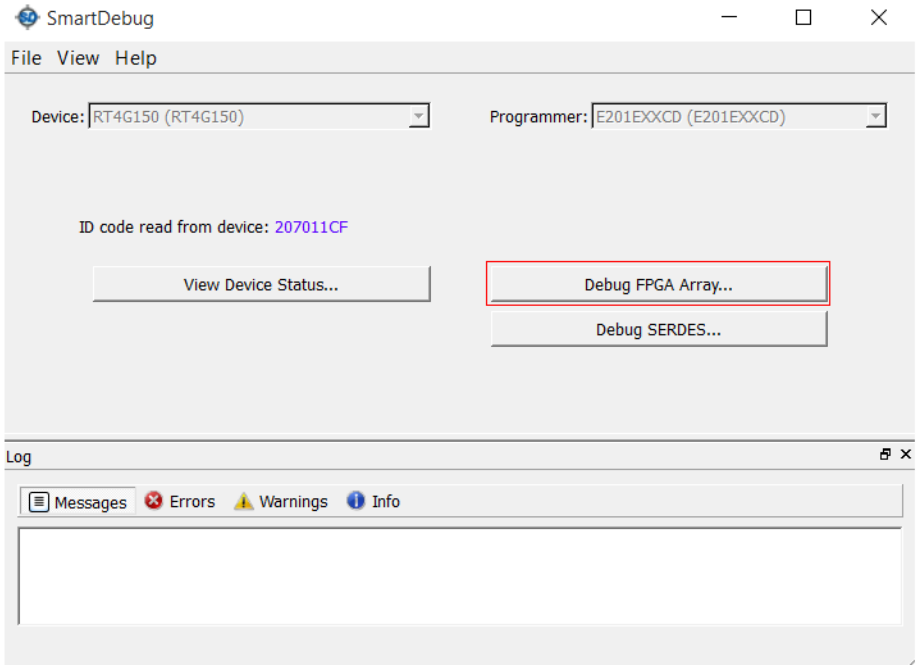
Figure 22 • **uPROM Configurator—Initialization client**



3. After adding the initialization client, run the design flow till the *Run PROGRAM Action* step.

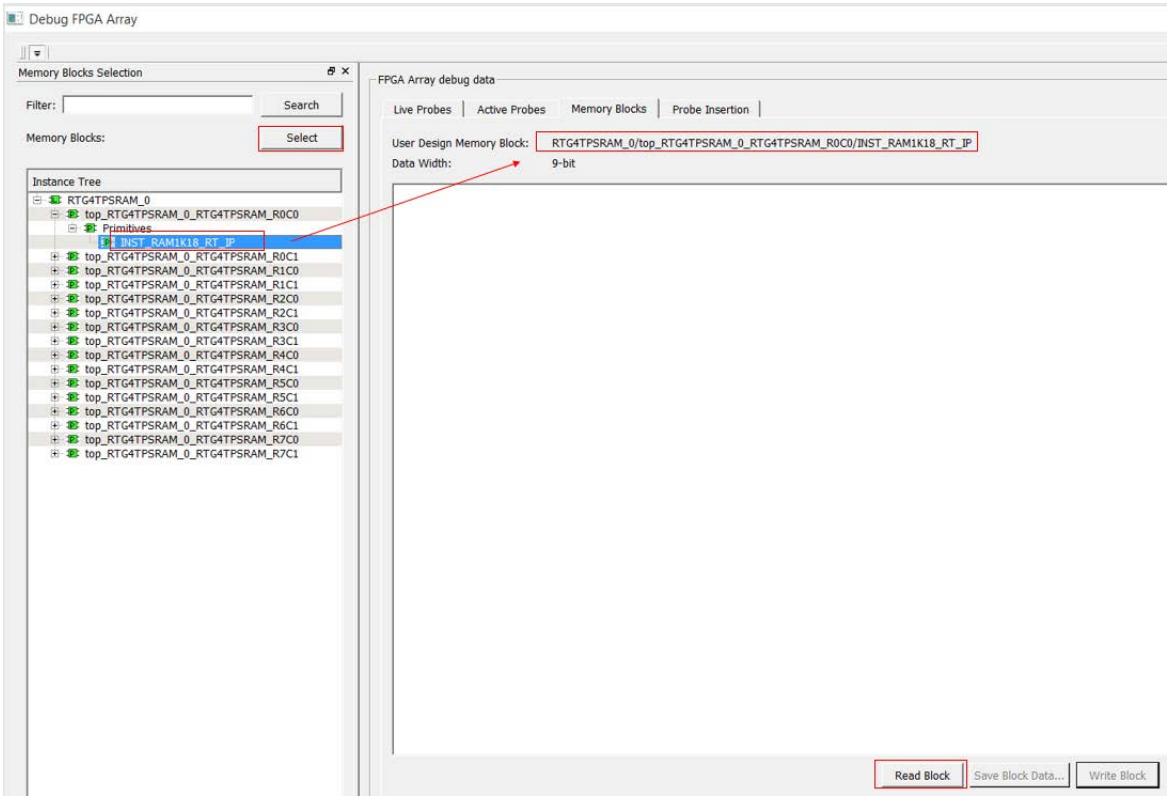
4. After the device gets programmed, double-click the **SmartDebug** design in the **Design Flow** tab to check whether the RAM block contents are reset.
5. In SmartDebug, click **Debug FPGA Array…**. as shown in the following figure.

Figure 23 • **SmartDebug Window**



6. In the **Debug FPGA Array** window, click the **Memory Blocks** tab, select the **INST_RAM1K18_RT_IP** instance, and click **Read Block** to read its content, as shown in the following figure.

Figure 24 • **Debug FPGA Array Window**



The entire RAM block content is reset to zero, as shown in following figure. You can select the other instance and read its content, which will also be reset to zero.

Figure 25 • **RAM Contents Reset to Zero**