Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer’s responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided “as is, where is” and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

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.
Contents

1 Revision History ......................................................... 1
  1.1 Revision 3.0 ......................................................... 1
  1.2 Revision 2.0 ......................................................... 1
  1.3 Revision 1.0 ......................................................... 1

2 Error Detection and Correction on RTG4 LSRAM Memory ................. 2
  2.1 Design Requirements ............................................. 2
  2.2 Demo Design ....................................................... 3
    2.2.1 Features ...................................................... 4
    2.2.2 Description .................................................. 4
    2.2.3 Test Modes ................................................... 4
      2.2.3.1 Loop Test ............................................... 4
      2.2.3.2 Manual Test ............................................ 5
  2.3 Clocking Structure ............................................... 5
  2.4 Reset Structure .................................................. 6

2.5 Setting Up the Demo Design ....................................... 6
  2.5.1 Jumper Settings ............................................... 6
  2.5.2 Programming the Demo Design ................................ 9
  2.5.3 Installing the EDAC Demo GUI ................................ 10
  2.5.4 Graphical User Interface .................................... 11

2.6 Running the Demo ................................................ 12
  2.6.1 Performing Loop Test ...................................... 13
  2.6.2 Performing Manual Test .................................... 14

2.7 Conclusion ......................................................... 15
# Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Top-Level Block Diagram</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Clocking Structure</td>
<td>5</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Reset Structure</td>
<td>6</td>
</tr>
<tr>
<td>Figure 4</td>
<td>USB to UART Bridge Drivers</td>
<td>7</td>
</tr>
<tr>
<td>Figure 5</td>
<td>On-board FlashPro5 Interface</td>
<td>8</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Board Set Up</td>
<td>8</td>
</tr>
<tr>
<td>Figure 7</td>
<td>FlashPro - New Project</td>
<td>9</td>
</tr>
<tr>
<td>Figure 8</td>
<td>FlashPro Programmer Status</td>
<td>10</td>
</tr>
<tr>
<td>Figure 9</td>
<td>EDAC GUI Installation</td>
<td>10</td>
</tr>
<tr>
<td>Figure 10</td>
<td>EDAC Demo GUI</td>
<td>11</td>
</tr>
<tr>
<td>Figure 11</td>
<td>EDAC GUI</td>
<td>12</td>
</tr>
<tr>
<td>Figure 12</td>
<td>1 Bit Error Correction Tab - Loop Test</td>
<td>13</td>
</tr>
<tr>
<td>Figure 13</td>
<td>2 Bit Error Detection Tab - Loop Test</td>
<td>13</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Writing to LSRAM Memory Location</td>
<td>14</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Injecting Error to LSRAM Memory Location</td>
<td>14</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Reading from LSRAM Memory Location</td>
<td>15</td>
</tr>
</tbody>
</table>
# Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Design Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>Jumper Settings</td>
<td>6</td>
</tr>
</tbody>
</table>
The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 3.0
Updated the document for Libero v11.9 SP1 software release.

1.2 Revision 2.0
Updated the document for Libero v11.8 SP2 software release.

1.3 Revision 1.0
The first publication of this document.
2 Error Detection and Correction on RTG4 LSRAM Memory

This reference design describes the error detection and correction (EDAC) capabilities of the RTG4™ FPGA LSRAMs. In a single event upset (SEU) susceptible environment, RAM is prone to transient errors caused by heavy ions. These errors can be detected and corrected by employing error correction codes (ECCs). The RTG4 FPGAs RAM blocks have built-in EDAC controllers to generate the error correction codes for correcting 1-bit error or detecting 2-bit error.

If 1-bit error is detected, the EDAC controller corrects the error bit and sets the error correction flag (SB_CORRECT) to active high. If a 2-bit error is detected, the EDAC controller sets the error detection flag (DB_DETECT) to active high.

For more information about RTG4 LSRAM EDAC functionality, see UG0574: RTG4 FPGA Fabric User Guide.

In this reference design, the 1-bit error or 2-bit error is introduced manually. EDAC is observed using a graphical user interface (GUI), utilizing the UART interface to access the LSRAM for data reads/writes, Libero® SoC SmartDebug (JTAG) is used to inject the errors into LSRAM memory.

2.1 Design Requirements

Table 1, page 2 lists the reference design requirements for running the RTG4 LSRAM EDAC demo.

<table>
<thead>
<tr>
<th>Design Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Requirements</td>
<td></td>
</tr>
<tr>
<td>RTG4 Development Kit:</td>
<td>Rev B kit with RT4G150-CB1657PROTO FPGA</td>
</tr>
<tr>
<td>– USB 2.0 cable</td>
<td></td>
</tr>
<tr>
<td>– FlashPro4 programmer</td>
<td></td>
</tr>
<tr>
<td>– 12 V, 5A AC power adapter and cords</td>
<td></td>
</tr>
<tr>
<td>Software Requirements</td>
<td></td>
</tr>
<tr>
<td>Libero® System-on-Chip (SoC)</td>
<td>v11.9 SP1</td>
</tr>
<tr>
<td>FlashPro programming software</td>
<td>v11.9 SP1</td>
</tr>
<tr>
<td>SmartDebug</td>
<td>v11.9 SP1</td>
</tr>
<tr>
<td>Host PC drivers</td>
<td>USB to UART drivers</td>
</tr>
<tr>
<td>Operating system</td>
<td>Any 64-bit Windows 7</td>
</tr>
</tbody>
</table>
2.2 Demo Design

Download the demo design files from the Microsemi website at:

http://soc.microsemi.com/download/rsc/?f=rt4g_dg0703_liberov11p9sp1_df

The demo design files include:

- Libero SoC project
- GUI Installer
- Programming files
- Readme.txt file

The GUI application on the host PC issues commands to the RTG4 device through the USB-UART interface. This UART interface is designed with CoreUART, which is a logic IP from Libero SoC IP catalog. The CoreUART IP in the RTG4 fabric receives commands and transmits them to the command decoder logic. The command decoder logic decodes the read or write command which is executed using the memory interface logic.

The memory interface block is used to read/write and monitor the LSRAM error flags. The built-in EDAC corrects the 1-bit error while reading from LSRAM and provides corrected data to the user interface but does not write corrected data back to LSRAM. The built-in LSRAM EDAC does not implement scrubbing feature. The demo design implements scrub logic, which monitors the 1-bit correction flag and updates the LSRAM with the corrected data if a single bit error occurs.

The host PC application injects 1-bit or 2-bit error into LSRAM data through SmartDebug.

Figure 1, page 3 shows the top-level block diagram of RTG4 LSRAM EDAC demo design.

Figure 1 • Top-Level Block Diagram

Following are the demo design configurations:

1. The LSRAM is configured for ×18 mode and EDAC is enabled by connecting LSRAMs ECC_EN signal to high.

   **Note:** The LSRAM EDAC is supported for only ×18 and ×36 modes.

2. The CoreUART IP is configured to communicate with host PC application at 115200 baud rate.

3. The CCC is configured to clock the CoreUART and other fabric logic at 80 MHz.
2.2.1 Features

Following are the demo design features:

• Read and write to LSRAM
• Inject 1-bit and 2-bit error using SmartDebug
• Display 1-bit and 2-bit error count values
• Provision to clear the error count values
• Enable or disable the memory scrubbing logic

2.2.2 Description

This demo design involves implementation of following tasks:

• Initializing and accessing LSRAM
  The memory interface logic implemented in the fabric logic receives the initialization command from GUI and initializes the first 100 memory locations of LSRAM with the incremental data. It also performs the read and write operations to the LSRAM by receiving the address and data from the GUI. For read operation, the design fetches the data from LSRAM and provides it to GUI for display. The expectation is that the design will not induce errors before using SmartDebug.

• Injecting 1-bit or 2-bit errors (Embedded SmartDebug exe)
  To inject the 1-bit or 2-bit errors into specified memory location of LSRAM, the demo GUI application invokes the SmartDebug (sdebug.exe) in batch mode. The following operations are performed using SmartDebug to inject 1-bit and 2-bit errors to LSRAM:
  a. Read the data from LSRAM
  b. Modify one or two LSB bits of data in the SmartDebug GUI.
  c. Write the modified data to the same LSRAM memory location
  During the LSRAM read and write operation through the SmartDebug (JTAG) interface, the EDAC controller is bypassed and does not compute the ECC bits for the write operation in step c.

• Error Counting
  8-bit counters are used to provide an error count and is design into the fabric logic to count 1-bit or 2-bit errors. The command decoder logic provides the count values to the GUI when receiving commands from the GUI.

2.2.3 Test Modes

Loop test and manual test are implemented in this demo, which are applicable to both 1-bit and 2-bit errors.

2.2.3.1 Loop Test

In loop test, the GUI executes these steps. Initially, all the error counters and EDAC related registers are placed in the RESET state.

1. Initialize the LSRAM memory.
2. Inject the 1-bit or 2-bit error to the same LSRAM memory locations through JTAG interface.
3. Read the data from the same LSRAM memory locations.
4. Send the 1-bit or 2-bit error detection and 1-bit error correction data in case of 1-bit error to the GUI.
2.2.3.2 Manual Test

This method allows manual testing of EDAC functionality, using which 1-bit or 2-bit errors can be introduced to any location within the LSRAM.

Following steps describes how to test EDAC functionality manually:

1. Write data to the specified address using the GUI fields.
2. Inject the 1-bit or 2-bit corrupted data to the same address location using the inject error options provided by GUI.
3. Read the data from the same address location then the LED on the board must toggle to notify the detection and correction of errors.
4. The corresponding error counter is displayed in the GUI. The GUI serial console logs all the actions performed in RTG4.

2.3 Clocking Structure

In this demo design, there is one clock domain. The internal 50 MHz oscillator drives the RTG4FCCC. The RTG4FCCC generates 80 MHz clock that provides clock source to reset_synchronizer, COREUART, cmd_decoder, TPSRAM_ECC and RAM_RW modules. The following figure shows the clocking structure of the demo design.

![Clocking Structure Diagram]
2.4 Reset Structure

In this demo design, the reset signal of COREUART, cmd_decoder and RAM_RW blocks are issued using reset_synchronizer module. The reset_synchronizer module releases active low reset when the DEVRST_N and RTG4FCCC lock are asserted. The following figure shows the reset structure of demo design.

Figure 3 • Reset Structure

2.5 Setting Up the Demo Design

The following sections describe how to setup the RTG4 Development Kit and GUI to run the demo design.

2.5.1 Jumper Settings

1. Connect the jumpers on the RTG4 Development Kit (see Table 2, page 6).

Table 2 • Jumper Settings

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Pin (From)</th>
<th>Pin (To)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>J11, J17, J19, J21, J23, J26, J27, J28</td>
<td>1</td>
<td>2</td>
<td>Default</td>
</tr>
<tr>
<td>J16</td>
<td>2</td>
<td>3</td>
<td>Default</td>
</tr>
<tr>
<td>J32</td>
<td>2</td>
<td>3</td>
<td>–</td>
</tr>
<tr>
<td>J33</td>
<td>1</td>
<td>3</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Appendix: Silk Screen, page 19 provides the RTG4 development kit silk screen to identify the jumper locations on board.

Note: Switch off the power supply switch, SW6, while connecting the jumpers.

2. Connect the USB cable (mini USB to Type A USB cable) to J47 of the RTG4 Development Kit and other end of the cable to the USB port of the host PC.

3. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the device manager of the host PC.
Figure 4, page 7 shows the USB 2.0 serial port properties and the connected COM95 and USB serial converter C.

**Figure 4 • USB to UART Bridge Drivers**

![USB to UART Bridge Drivers](image)

**Note:** If the USB to UART bridge drivers are not installed, download and install the drivers from [www.microsemi.com/documents/CDM_2.08.24_WHQL_Certified.zip](http://www.microsemi.com/documents/CDM_2.08.24_WHQL_Certified.zip)
4. Disable the on-board FlashPro5 interface as shown in Figure 5, page 8. This is required to avoid the RTG4 device reset while injecting errors.

**Figure 5 • On-board FlashPro5 Interface**

5. Connect the external FlashPro5 programmer to J22 header.

**Figure 6 • Board Set Up**
2.5.2 Programming the Demo Design

The following steps describe how to program the demo design:

1. Download the demo design from:
   \[http://soc.microsemi.com/download/rsc/?f=rt4g_dg0703_liberov11p9sp1_df\]
2. Launch the FlashPro software.
3. Click **New Project**. In the **New Project** window, enter the project name.
4. Click **Browse** and navigate to the location to save the project.
5. Select **Single device** as the **Programming mode**.
6. Click **OK**.

**Figure 7 • FlashPro - New Project**

7. Click **Configure Device**.
8. Click **Browse** and navigate to the location where the \[LSRAM_EDAC.stp\] file is located. The default location is: \[<download_folder>\rtg4_dg0703_liberov11p9sp1_df\ProgrammingFiles\]
9. Click **View Programmers**.
10. Enable the external FlashPro5 programmer and disable the internal FlashPro5 Programmer.
11. Click **PROGRAM** to start programming the device. Wait until the **Programmer Status** is changed to **RUN PASSED**.

**Figure 8 • FlashPro Programmer Status**

2.5.3 **Installing the EDAC Demo GUI**

The EDAC demo is provided with a user-friendly GUI that runs on the host PC, which communicates with the RTG4 Development Kit. The UART is used as the underlying communication protocol between the host PC and RTG4 Development Kit.

The following steps describe how to install the EDAC demo GUI:

1. Go to `<Download_folder>\rtg4_dg0703_liberov11p9sp1_df\GUI\` and double-click **setup.exe**.
2. Do not change the default options and click **Next**.

**Figure 9 • EDAC GUI Installation**

3. Click **Finish** to complete the installation.
2.5.4  **Graphical User Interface**

The EDAC demo GUI provides simple user interface to communicate with the RTG4 FPGA through UART interface (see Figure 10, page 11).

**Figure 10**  
EDAC Demo GUI

The GUI contains the following sections:

1. **SmartDebug EXE Path**: To browse the SmartDebug (sdebug.exe) path. The GUI requires SmartDebug to inject the errors through JTAG.
2. COM port selection to establish the UART connection to RTG4 FPGA with the 115200 baud rate.
3. Selection of 1-bit error correction and 2-bit error detection tabs.
4. **LSRAM Memory Write**: To write the 8-bit data to the specified LSRAM memory address.
5. **Error Inject**: To inject the error to the specified LSRAM memory address.
6. **Memory Scrubbing**: To enable or disable the scrubbing logic.
7. **LSRAM Memory Read**: To read the 8-bit data from the specified LSRAM memory address.
8. **Error Count**: Displays the error count and provides an option to clear the counter value to zero.
9. **Loop Test**: Performs the error injection and error correction or detection operations to 100 LSRAM memory locations.
10. **Log Data**: Provides the status information for every operation performed using the GUI.
2.6 Running the Demo

Following steps describe how to run the demo:

1. Invoke the EDAC GUI from All Programs > EDAC Demo > EDAC Demo GUI (see Figure 11, page 12).
2. Click Browse and navigate to the location where the Libero software is installed, and select the sdebug.exe path.
3. Select the COM95 port from the list and click Connect.

Figure 11 • EDAC GUI

4. Select the 1 Bit Error Correction tab or 2 Bit Error Detection tab, as shown in Figure 12 on page 13 and Figure 13 on page 13.
5. Two types of tests can be performed: Manual and Loop.
2.6.1 Performing Loop Test

Click Loop Test. It runs in loop mode where continuous correction and detection of errors is done. The loop runs for 100 iterations. All actions performed in RTG4 are logged in the Serial Console section of the GUI. The GUI prints the following values in the serial console for every iteration:

- Data read from LSRAM for respective address
- Expected data
- Error count

Figure 12 • 1 Bit Error Correction Tab - Loop Test

Figure 13 • 2 Bit Error Detection Tab - Loop Test
2.6.2 Performing Manual Test

In this method, errors are introduced manually using the inject error option in GUI. Use the following steps to execute 1-bit error correction or 2-bit error detection.

1. Enter **Address** and **Write Data** fields (use 8-bit Hexadecimal values) in the LSRAM Memory Write section.
2. Click **Write** (see Figure 14, page 14).

*Figure 14 • Writing to LSRAM Memory Location*

3. Enter **Address** field in the **Error Inject** section for injecting 1-bit error.
4. Click **Inject Error** (see Figure 15, page 14). The GUI invokes the sdebug.exe (SmartDebug) in batch mode and injects error.

*Figure 15 • Injecting Error to LSRAM Memory Location*
5. Enter **Address** field in the **LSRAM Memory Read** section.

6. Click **Read** (see **Figure 16**, page 15).

**Figure 16** • Reading from LSRAM Memory Location

![Error Detection and Correction Demo GUI](image)

7. Observe **Error Count** and **Read Data** fields in the GUI. The error count value increases by 1.
   a. In case of 1-bit error injection, the **Read Data** field displays the correct data as the EDAC corrects the error bit.
      If memory scrubbing is not enabled, then the error count is incremented for every read from same LSRAM address as it causes the 1-bit error.
   
   b. In case of 2-bit error injection, the **Read Data** field displays the corrupted data.

All the actions performed in RTG4 are logged in Serial Console section of GUI.

### 2.7 Conclusion

This demo highlights the EDAC capabilities of the RTG4 LSRAM memories. The 1-bit error or 2-bit error are introduced manually and 1-bit error correction and 2-bit error detection is observed using a GUI.