SDRAM Demo Quickstart Guide

REV A

Publication Date: 2012/10/13 XMOS © 2012, All Rights Reserved.



Table of Contents

1	sc_sdram_burst demo : Quick Start Guide			
	1.1	Hardware Setup		
	1.2	Import and Build the Application		
	1.3	Run the Application		
	1.4	Next Steps		
		1.4.1 Look at the Code		
		1.4.2 Try the Benchmark and Regressession Demo		
		1.4.3 Try Other Application Demos which use the SDRAM		

–<mark>X</mark>MOS[°]–

IN THIS CHAPTER

- ► Hardware Setup
- Import and Build the Application
- Run the Application
- Next Steps

This simple demonstration of xTIMEcomposer Studio functionality uses the XA-SK-SDRAM Slice Card together with the xSOFTip module_sdram to demonstrate how the module is used to read and write to the SDRAM.

1.1 Hardware Setup

The XP-SKC-L2 Slicekit Core board has four slots with edge conectors: SQUARE, CIRCLE, "TRIANGLE" and STAR.

To setup up the system:

- 1. Connect XA-SK-SDRAM Slice Card to the XP-SKC-L2 Slicekit Core board using the connector marked with the STAR.
- 2. Connect the XTAG Adapter to Slicekit Core board, and connect XTAG-2 to the adapter.
- 3. Connect the XTAG-2 to host PC. Note that the USB cable is not provided with the Slicekit starter kit.
- 4. Set the XMOS LINK to OFF on the XTAG Adapter.
- 5. Switch on the power supply to the Slicekit Core board.

1.2 Import and Build the Application

- 1. Open xTIMEcomposer and check that it is operating in online mode. Open the edit perspective (Window->Open Perspective->XMOS Edit).
- 2. Locate the 'Slicekit SDRAM Simple Demo' item in the xSOFTip pane on the bottom left of the window and drag it into the Project Explorer window in the xTIMEcomposer. This will also cause the modules on which this application depends (in this case, module_sdram) to be imported as well.
- 3. Click on the app_sdram_demo item in the Explorer pane then click on the build icon (hammer) in xTIMEcomposer. Check the console window to verify that the

application has built successfully. There will be quite a number of warnings that bidirectional buffered port not supported in hardware. These can be safely ignored for this component.

For help in using xTIMEcomposer, try the xTIMEcomposer tutorial.

Note that the Developer Column in the xTIMEcomposer on the right hand side of your screen provides information on the xSOFTip components you are using. Select the module_sdram component in the Project Explorer, and you will see its description together with API documentation. Having done this, click the *back* icon until you return to this quickstart guide within the Developer Column.

1.3 Run the Application

Now that the application has been compiled, the next step is to run it on the Slicekit Core Board using the tools to load the application over JTAG (via the XTAG2 and Xtag Adaptor card) into the xCORE multicore microcontroller.

- 1. Select the file app_sdram_demo.xc in the app_sdram_demo project from the Project Explorer.
- 2. Click on the Run icon (the white arrow in the green circle).
- 3. Select "XCore Application", click "Run".
- 4. The debug console window in xTIMEcomposer should then display the message SDRAM demo complete. This has been generated from the application code via a call to the printf() function.
- 5. If SDRAM demo fail is shown then check that the XMOS LINK is set to OFF then repeat.

1.4 Next Steps

Now that the demo has been run you could try and adjust the BUF_WORDS setting. Note, if you try to allocate more than 64KB of internal XCore memory then the demo will not compile as the Xcore wont be able to hold the image.

1.4.1 Look at the Code

- 1. Examine the application code. In xTIMEcomposer navigate to the src directory under app_sdram_demo and double click on the app_sdram_demo.xc file within it. The file will open in the central editor window.
- 2. Find the main function and note that it runs the application() function on a single logical core.
- 3. Find the sdram_buffer_write function within application(). There are 4 SDRAM I/O commands: sdram_buffer_write, sdram_buffer_read, sdram_full_page_write, sdram_full_page_read. They must all be followed by a sdram_wait_until_idle before another I/O command may be issued. When



the sdram_wait_until_idle returns then the data is now at it destination. This functionality allows the application to be getting on with something else whilst the SDRAM server is busy with the I/O.

4. There is no need to explicitly refresh the SDRAM as this is managed by the sdram_server.

1.4.2 Try the Benchmark and Regressession Demo

- 1. app_sdram_benchmark benchmarks the performance of the module. It does no correctness testing but instead tests the throughput of the SDRAM server.
- 2. app_sdram_regress this demo runs a series of regression tests of increasing difficulty, beginning from using a single core for the sdram_server with one core loaded progressing to all cores being loaded to simulate an XCore under full load.

To try these repeat the procedure in "Import and Build the Application" but with either app_sdram_benchmark or app_sdram_regress.

1.4.3 Try Other Application Demos which use the SDRAM

There are two other siognificant application demos which utilise this component.

- The Display Controller Demo combines the SDRAM Slice with the XA-SK-SCR480 LCD Slice Card to create a fully functioning 480x272 display controller, with the SDRAM actig as the framebuffer.
- ▶ The Audio Reverb Demo utilises the SDRAM component to store enough audio samples to create large audio delay lines, which are a required component of various audio effects. In this case, a reverberation effect is demonstrated.





Copyright © 2012, All Rights Reserved.

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.

XMOS and the XMOS logo are registered trademarks of Xmos Ltd. in the United Kingdom and other countries, and may not be used without written permission. All other trademarks are property of their respective owners. Where those designations appear in this book, and XMOS was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

7/7