Tampere University of Technology

Faculty of Computing and Electrical Engineering

Department of Computer Systems

SDRAM Controller IP for DE2 board Reference Manual

Author:
Antti Alhonen

Updated:
November 9, 2011

Contents

1 REVISION HISTORY

Table 1

Revision	Author	Date	Description
1.0	Antti Alhonen	09.11.2011	First published version

2 DOCUMENT OVERVIEW

2.1 SCOPE

This documentation describes the usage of DCS/TUT made IP to allow simple usage of SDRAM chip from the user-created HW application.

This particular version is designed and tested with the SDRAM chip featured on Altera DE2 board. With minor modifications regarding timing and data width, other types of SDRAM can be used, too.

Users wanting to control an SDRAM directly from an soft-core processor are encouraged to use the controllers provided by the processor vendor.

2.2 AUDIENCE

- SoC developers
- Users of development boards that include an FPGA and A2V64S40CTP or compatible SDRAM.

3 INTRODUCTION

3.1 CONTACT INFORMATION

If you face any problems using the IP offered, please do not hesitate to ask for help or to give suggestions. You can contact Antti Alhonen (*antti.alhonen@tut.fi*).

3.2 INTRODUCTION

Using an SDRAM requires much more attention than using an SRAM chip, due to a need of refresh operations. Also, the timing requirements of any memory can be more complex than simple HW application developers would like to program in their application code. Therefore, a separate controller IP is needed.

3.3 BRIEF DESCRIPTION

This controller gives a simple command bus, direct addressing to memory via an address bus, and data bus with empty—read enable handshake signaling for writing to the memory, and full—write enable handshake signaling for reading from the memory. FIFOs can be used to simplify the task of usage.

The IP supports block reads and writes; your application simply gives the amount to write or read via a data amount bus when issuing the write or read command.

3.4 Designs provided

de2_sdram.vhd implements the whole IP; some of the ports connect to the SDRAM chip via the IO pins on FPGA, some to your application.

de2_sdram_tester.vhd works as a simple testbench and as a usage example. It can work as a self-contained unit showing the status using LEDs, or you can use a logic analyzer tool such as Altera SignalTap to further verify the operation. Please note that only four 16-bit locations on the memory are verified, so this unit primarily verifies the logical operation of the controller, not the condition of the memory chip.

The tester uses fifo.vhd and de2_sdram_pll.vhd to generate a correct clock signal for the chip on DE2 board.

To pass the test, set the switches SW(15...0) as "0101010101010101". The value defined by switches is written alongside other test values, and the result is compared to this value. This way you can verify that the test can indeed fail.

3.5 Clock and reset

The supplied IP automatically calculates the SDRAM timing as you provide the correct clock frequency information via *clk_freq_mhz_g* (in MHz). The operation with the SDRAM chip in synchronous. It may be necessary to include a phase shift to the clock going to the SDRAM, depending on the board. A PLL to perform this is instantiated inside de2_sdram_tester.vhd.

The supplied reset is asynchronous active-low. After releasing the reset, the initialization of the controller takes about 200 microseconds before the operation can start. No data is read by the controller during this time from your application.

3.6 IP-XACT files provided

We have created IP-XACT descriptions in Kactus 2 design software. Bus interfaces include an IO bus for the SDRAM chip, and an application bus to use the SDRAM.

We provide IP-XACT components of the controller and the example application and an example design to connect them together.

TUT.soc.de2_sdram_example is a complete example design which synthesizes in Quartus II.

4 Usage

4.1 Writing

Because the controller relies not only to a "not empty" signal but also for "one data left" signal for the writing operation, it is strongly recommended to use a FIFO buffer to simplify the usage.

When you have the first word (16 bits) of your data ready, do the following:

1. Output the amount of the data you are going to write **in words** to data_amount_in, and the SDRAM address to address_in.
2. Output a write command **"10"** to command_in.
3. Output the data words to the FIFO by inserting the data to the bus and asserting write enable signal to the FIFO, once for every word.
4. The SDRAM controller reads the data from the FIFO instantly when it is not busy.
5. If the amount of data described in data_amount_in during the first word is not satisfied yet, the controller waits for more data.
6. Writing to the SDRAM may stall occasionally during refresh operations. Make sure you use the full signal of your FIFO to stall your application in this case.
7. After all the words are read from the FIFO by the controller, it starts to wait for new commands.

Please see the provided example design, sdram_test.vhd. As you can see, in this example we set the write command after the data is in the FIFO. This can be done only if the FIFO is big enough to hold all the data.

4.2 Reading

A FIFO is suggested for reading, too. Otherwise, you need to stall the reader by asserting output_full_in before you know that you cannot read the data coming from the SDRAM controller when output_we_out becomes high.

When you want to start reading, do the following:

1. Output the amount of the data you are going to read **in words** to data_amount_in, and the SDRAM address where the read starts to address_in.
2. Output the read command **"01"** to command_in.
3. The SDRAM controller reads the data from the SDRAM and inserts it to your application FIFO by asserting output_we_out.
4. After data_amount_in words have been read, the controller starts accepting new commands.

Please see the provided example design, sdram_test.vhd.

5 KNOWN ISSUES

No known issues currently. Please report any problems.