Dualport Module for the SDRAM Controller

July 12, 2005 (Version 1.0) Application Note by D. Vanden Bout

Summary

This application note describes a module that adds dualport read/write access to the host-side port of the XESS SDRAM controller.

Dualport Features

The dualport module attaches to the host-side port of the XESS SDRAM controller and splits it into two identical host-side ports. Each of these ports operates identically to the orignal host-side port so no modifications are needed in any applications that used the original SDRAM controller. An application performs memory read/write operations on its port independently of any operations that occur on the other port. The total SDRAM bandwidth can be allocated between the two ports to match the data rates of the attached applications. Dualport modules can be cascaded to build SDRAM interfaces with three or more independent ports.

Dualport Interface

Generic Parameters

Several generic parameters affect the operation of the dualport module:

PIPE_EN: This parameter should be set to `true` if the SDRAM controller has pipelined operations enabled.

PORT_TIME_SLOTS: This parameter allocates the SDRAM bandwidth between the ports by assigning time slots within an interval to each port.

DATA_WIDTH: This parameter should be set to the same width as the host-side databus of the SDRAM controller.

HADDR_WIDTH: This parameter should be set to the same width as the host-side address bus of the SDRAM controller.

I/O Ports

The interface to the dualport module for the XSA Board is shown in Figure 1, while Figure 2 depicts the interface for the XSB Board. The functions of the I/O signals are as follows:

clk: This is the master clock input. It should be driven at the same clock rate as the SDRAM controller to which the dualport module is attached.

The host-side signals for the SDRAM controller are replicated to form two independent ports. The functions and timing for the signals shown below are identical to those on the host-side interface of the SDRAM controller.

Port 0	Port 1	Port to SDRAM Controller
rst0	rst1	rst
rd0	rd1	rd
wr0	wr1	wr
earlyOpBegun0	earlyOpBegun1	earlyOpBegun
opBegun0	opBegun1	opBegun
rdPending0	rdPending1	rdPending
done0	done1	done
rdDone0	rdDone1	rdDone
hAddr0	hAddr1	hAddr
hDIn0	hDIn1	hDIn
hDOut0	hDOut1	hDOut
status0	status1	status

Dualport Application

A simple memory tester application is used to demonstrate the use of the dualport module. The memory tester is composed of two identical modules,

one which tests the lower half of memory while the other tests the upper half. Each memory tester module accesses its section of the memory through one of the ports on the dualport module. (The module that tests the lower half of memory is attached to port 0; the other memory tester module attaches to port 1.) Each memory tester does the following:

- It initializes a pseudo-random number generator (RNG) with a known seed value.
- It writes a sequence of random numbers throughout its range of memory addresses.
- It re-initializes the RNG with the seed value.
- It reads back the contents of memory and compares it to the sequence from the RNG. Any mismatch indicates an error reading or writing the memory.

The source files that describe this application are:

common.vhd: Some functions and definitions useful in many applications are provided in this file.

memtest.vhd: The memory tester state machine is described in this file.

randgen.vhd: This file contains the RNG used by the memory tester.

sdramcntl.vhd: This file describes the SDRAM controller and the associated dualport module.

xsasdramcntl.vhd: This file creates a wrapper around the SDRAM controller core to customize it for the XSA Board.

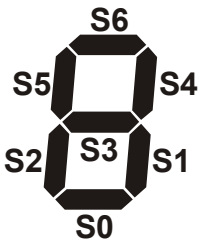
test_dualport_core.vhd: This file combines two memory testers, the dualport module and the SDRAM controller core to make the complete SDRAM tester.

test_dualport.vhd: This file instantiates the test_dualport_core module to create a dualport SDRAM tester for a particular board.

test_dualport.ucf: This file contains the pin assignments for the I/O signals of the dualport SDRAM tester for a particular board.

test_dualport.npl: This file tells the Xilinx WebPACK tools how to combine the source files to create the dualport SDRAM tester application for a particular board.

Once the memory tester application is compiled and downloaded into an XSA Board, you should see the status of the upper and lower SDRAM tests reflected in the activation of the LED segments. The LED segments are labeled as follows:



The meaning of the LED segment activations is as follows:

Lower Memory Test Status	S0	S1	S2
Initialization	ON	OFF	OFF
Writing RNG sequence to the lower half of SDRAM	OFF	ON	OFF
Reading data from the lower half of the SDRAM and comparing to RNG sequence	OFF	OFF	ON
Passed – no mismatches found	OFF	OFF	OFF
Failed – mismatches found	ON	ON	ON

Upper Memory Test Status	S3	S4	S5
Initialization	ON	OFF	OFF
Writing RNG sequence to the upper half of SDRAM	OFF	ON	OFF
Reading data from the upper half of the SDRAM and comparing to RNG sequence	OFF	OFF	ON
Passed – no mismatches found	OFF	OFF	OFF
Failed – mismatches found	ON	ON	ON

The memory test will run repeatedly as long as pushbutton SW2 on the XSA Board is pressed. You should see that the S4-S5 LEDs flash more quickly than the S1-S2 LEDs. Because port 1 is allocated more time slots than port 0, the upper half of memory can be tested in less time so the associated LEDs flash more quickly than those for the lower half.

The dualport memory tester for the XSB Board does not provide as much visual feedback. Once the memory tester application is compiled and downloaded into the XSB Board, you should see the LEDs glow and then either display “OO” (if the dualport SDRAM test completed without errors) or “EE” (if an error occurred).

Allocating SDRAM Bandwidth to Ports

The PORT_TIME_SLOTS generic parameter is used to allocate the SDRAM bandwidth between the ports of the dualport module. PORT_TIME_SLOTS is a 16-bit std_logic_vector where each bit corresponds to a time slot during which a read or write of the SDRAM can occur. Setting a bit to zero assigns the time slot to port 0, and setting it to one assigns the time slot to port 1. For example, the generic parameter assignment

PORT_TIME_SLOTS => “1111000011110000”

assigns eight time slots to each port, with each port getting four contiguous accesses to the SDRAM before the dualport module switches control to the other port. Hence, each port is allocated half the bandwidth of the SDRAM.

The PORT_TIME_SLOTS parameter only affects the operation of the dualport module when applications on both ports try to simultaneously access the SDRAM. An application will get immediate access to the SDRAM on its port if no read or write operation is in progress on the other port. So the assignment

PORT_TIME_SLOTS => “1111111111111111”

allows port 0 to access the SDRAM only when port 1 is not accessing it, but it does not completely block port 0 from ever accessing the SDRAM.

The grouping of bits in the PORT_TIME_SLOTS parameter affects the efficiency and responsiveness of the dualport module. The assignment

PORT_TIME_SLOTS => “1111111100000000”

assigns half the SDRAM bandwidth to each port but may block a port’s access to the SDRAM for up to eight time slots while the other port has priority. To reduce this delay, the assignment

PORT_TIME_SLOTS => “0101010101010101”

alternates access to the SDRAM between the ports but wastes time because the SDRAM controller pipeline must be cleared before each switch. The appropriate grouping of bits is application-specific, but using contiguous groups is generally best.

Expanding the Number of Ports

It is easy to create multi-port SDRAM interfaces just by cascading a number of dualport modules as shown in Figure 3. Be aware that going through the dualport module can increase propagation delays in your application and may decrease its maximum clock frequency.

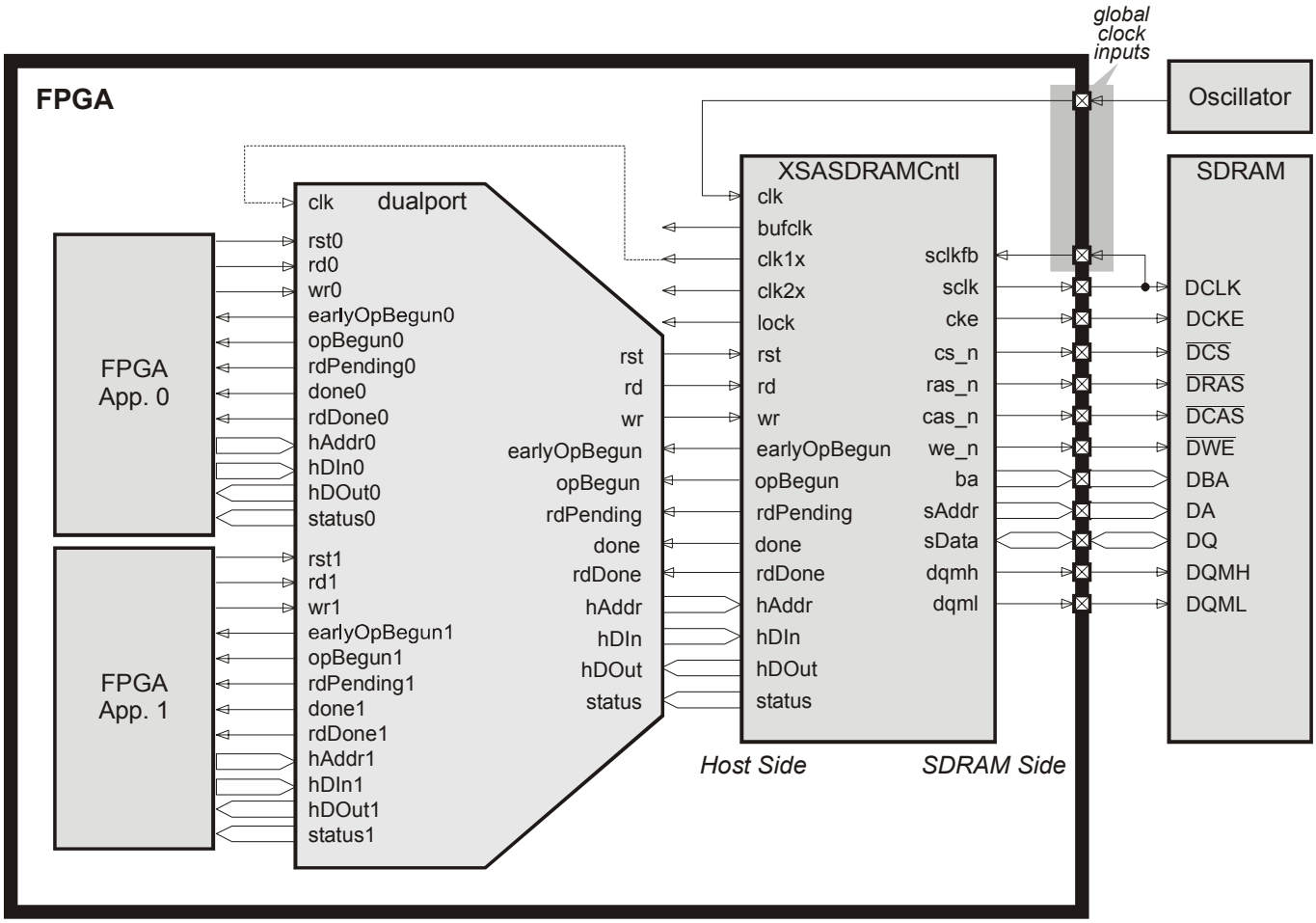


Figure 1: Dualport interface to the XSA Board SDRAM controller.

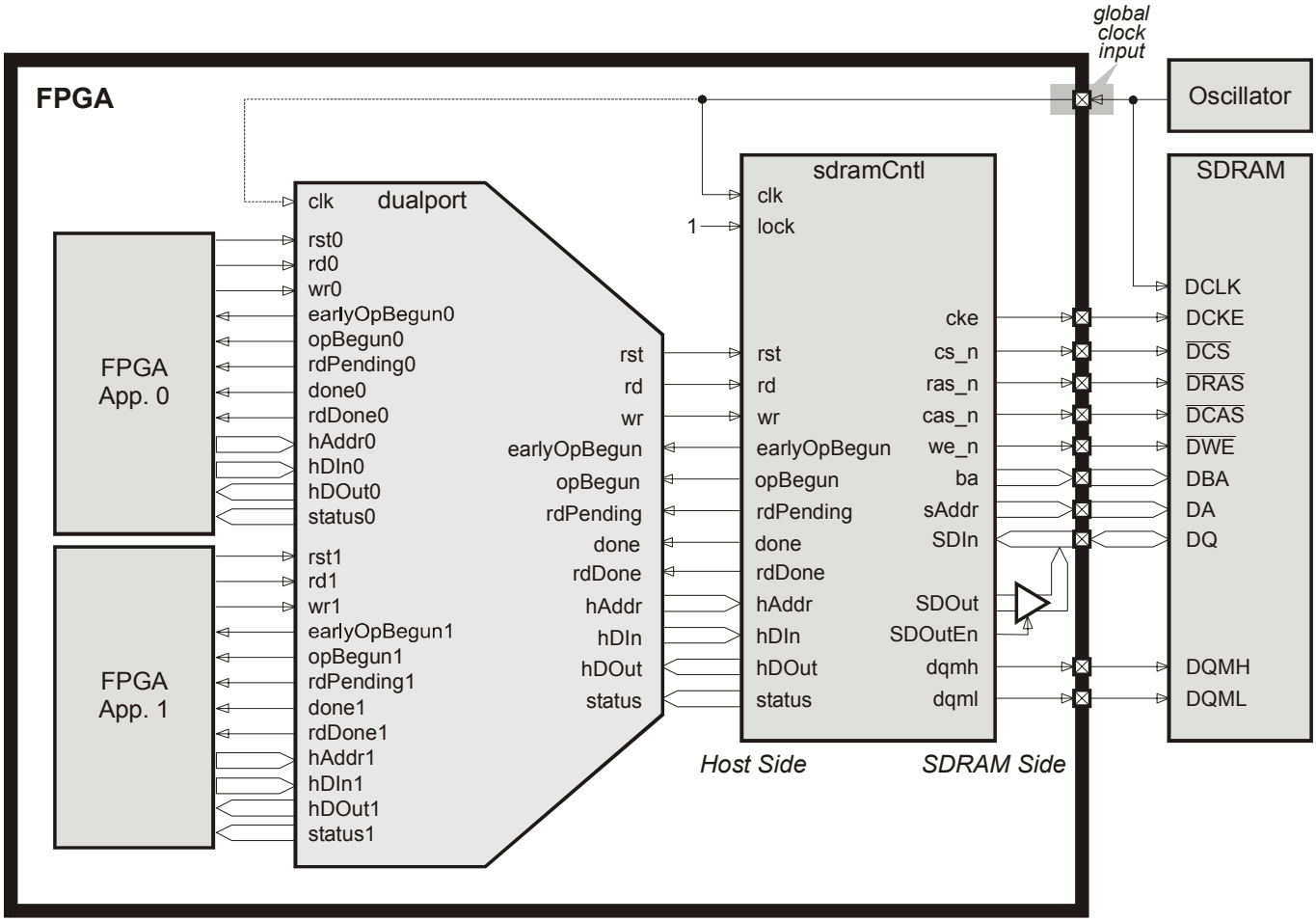


Figure 2: Dualport interface to the XSB Board SDRAM controller.

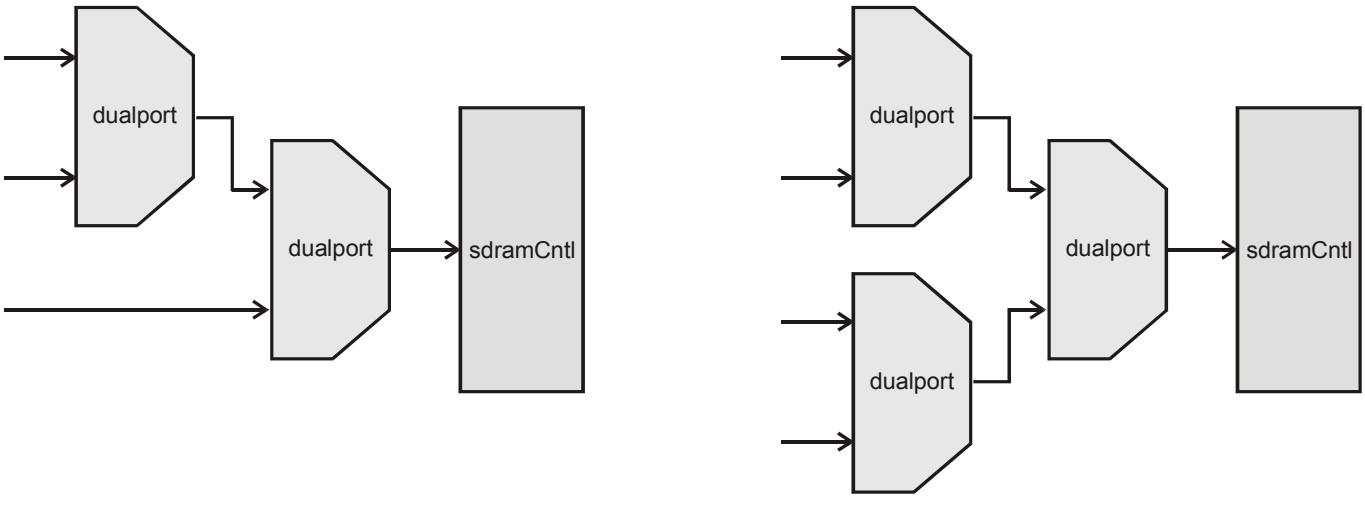


Figure 3: Building three-port and four-port SDRAM interfaces by cascading dualport modules.

Revision	Date	Comments
1.0	07/12/05	Initial version.