Version for EDK 8.2.02i as of February 1, 2007

1 Goals

You will be learning how to connect external memory to the OPB bus controller. If you wish to use a controller that does not have the OPB bus interface you should still go through this module to become familiar with the clocking issues and then look at Section 7.

This module has not yet been updated for the XUPV2P board. If someone would like to submit changes for doing something equivalent with the DDR on that board, please do so.

2 Background

In previous designs you have used internal memory structures instantiated within the FPGA. This memory has been connected to the Local Memory Bus (LMB). The external memory on the Multimedia Board is zero-bus turnaround (ZBT) memory and will be connected to the system via the On-chip Peripheral Bus (OPB).

External memory can be connected to an EDK system using Xilinx's External Memory Controller (EMC) core. The EMC is an OPB peripheral that can connect up to four external banks into the address space of the MicroBlaze.

The Base System Builder wizard has the ability to create a design that contains a connection to a single ZBT bank. The design provided by the Base System Builder does work, though it does not properly handle the memory clocking. In particular, clock synchronization becomes a major concern in complex or high-speed designs. This document describes how to modify the design provided by the Base System Builder so that the clock to the external memory is properly synchronized to the internal clock.

For high-speed interfaces, such as the ZBT memory, it is necessary to account for delays through the FPGA and on the traces of the PCB. The Digital Clock Manager (DCM) available on the Xilinx Virtex-II FPGA can be used to account for these delays.

A clock feedback wire, with delay equal to the clock lines to the five memory chips according to the Multimedia User Guide, is available to align the phase of the internal clocks to the clock arriving at the memory chips using a DCM.

The Multimedia board contains a total of five banks of ZBT memory, providing 10MB of space. After learning how to use the EMC to connect to a single bank, this document will address how to connect four of the five banks to a single memory controller (the fifth bank can be connected to a second memory controller).

3 Requirements

Module m01: Building a MicroBlaze System in XPS
Module m02: Adding IP and Device Drivers — GPIO and Polling

4 Preparation

- Summaries of how to connect external memories are given in the "Connecting to Memory" section of the documentation for the OPB External Memory Controller Core.
- You might also want to look at the documentation describing the Digital Clock Manager (DCM) found in the Virtex II User's Guide and the Libraries Guide.

5 Connecting a Single ZBT Bank

5.1 Step-by-step

1. Unzip the ZIP file that accompanies this module.

2. Create a new system with Base System Builder in the `lab8/` directory you extracted from the ZIP file. Ensure that in ZBT_512Kx32 is selected with the OPB EMC option in the Configure Additional IO Interfaces screen. You may also wish to select Generate Sample Application in the Software Configuration screen as the application includes a test of the ZBT memory.

3. The created project should include one DCM module already (`dcm_0`). By selecting IP Catalog tab, add another `dcm_module` instance. The two DCM modules will be used to create the clocking scheme shown in Figure 1.

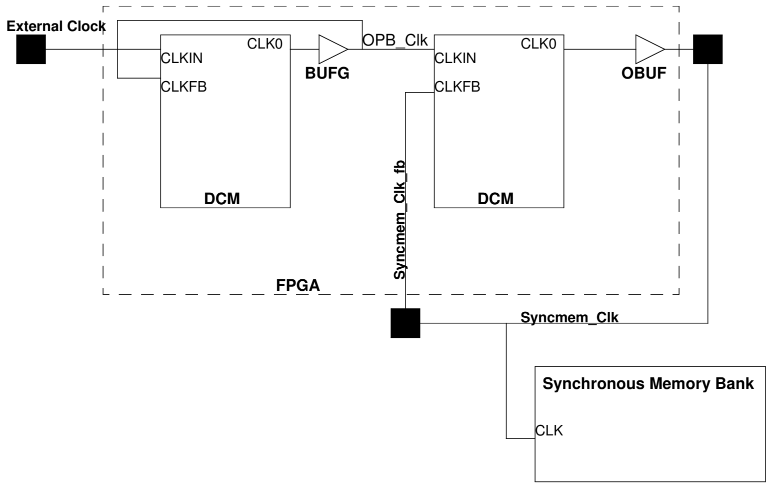


Figure 1: The clocking scheme required for external ZBT memory with clock feedback (from Xilinx's OPB EMC datasheet).

Connect the ports of the first DCM module (`dcm_0`) to the following nets:

RST	dcm_0_rst
CLKIN	dcm_clk_s
CLKFB	sys_clk_s
CLK0	sys_clk_s
LOCKED	dcm_0_lock

Connect the ports of the second DCM module (`dcm_module_0`) to the following nets:

RST	zbt_dcm_rst
CLKIN	sys_clk_s
CLKFB	zbt_dcm_clk_fb
CLK0	zbt_dcm_clk
LOCKED	zbt_dcm_clk_lock

4. Using the Add External Port button of the Ports tab of the System Assembly View, create an output port named `fpga_0_ZBT_512Kx32_Mem_CLOCK_FB_OUT` and an input port `fpga_0_ZBT_512Kx32_Mem_CLOCK_FB_IN`. It is not necessary to classify the ports as CLK signals, but don't forget to specify the port directions. Connect the two new ports to the `zbt_dcm_clk` and `zbt_dcm_clk_fb` nets respectively. Change the `C_CLKIN_PERIOD` parameter for each DCM to 37.037037. The default values for the other parameters should be appropriate. The DCM will lock when the edges of its `CLKFB` and `CLKIN` inputs are aligned.

5. Locate the port `fpga_0_ZBT_CLOCK` and rename it to `fpga_0_ZBT_CLOCK_pin` in order to make it consistent with the external port naming scheme. This was generated by the Base System Builder wizard. Notice that it is by default connected to `sys_clk_s`. Connect it to `zbt_dcm_clk` instead.

6. Edit the `system.ucf` file to connect the two new ports to appropriate pins. This can be accomplished by adding the following lines.

   ```
   Net fpga_0_ZBT_512Kx32_Mem_CLOCK_FB_IN_pin LOC=AE15;
   Net fpga_0_ZBT_512Kx32_Mem_CLOCK_FB_IN_pin FAST;
   Net fpga_0_ZBT_512Kx32_Mem_CLOCK_FB_OUT_pin LOC=AH14;
   Net fpga_0_ZBT_512Kx32_Mem_CLOCK_FB_OUT_pin FAST;
   ```

   You should also take this opportunity to update the UCF to reflect your renamed `fpga_0_ZBT_CLOCK_pin` port.

7. The DCM takes time to synchronize the clocks and lock. During this time, it is undesirable for the MicroBlaze or other components to be operating. The `LOCKED` signal of the DCM can be used to keep other components in a reset state until the DCM is ready. A custom core, `clk_align_v1_00_a`, has been created that helps accomplish this.

8. Add an instance of the `clk_align` core to the system and add all of its ports. Connect the ports with the internal connections that follow:

`external_clk`	`dcm_clk_s`
`external_reset`	`sys_rst_ext`
`dcm0_locked`	`dcm_0_lock`
`dcm1_locked`	`zbt_dcm_clk_lock`
`dcm0_reset`	`dcm_0_rst`
`dcm1_reset`	`zbt_dcm_rst`
`system_reset`	`sys_rst_s`

9. Locate the system port that connects the external `sys_rst_pin` pin to the `sys_rst_s` net. Alter it so that it connects to the `sys_rst_ext` net.

10. The system is now ready to be built and downloaded.

11. Do a quick sanity check by running XMD and doing a memory read and a memory write to the address space associated with the ZBT. The address space associated with the ZBT is not the same as the address space associated with the EMC. To find the address space, check the Parameters tab of Add/Edit Cores and select the instance of the `opb_emc`.

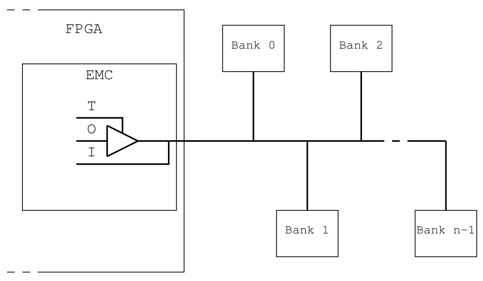
12. Run the automatically-generated memory test application, TestApp_Memory, for another quick test of the ZBT memory communications. You'll need to compile the project (try right-clicking on it and selecting Build Project), launch HyperTerminal or something equivalent to monitor the program output, download the program via XMD, and run it.

6 Connecting Multiple ZBT banks

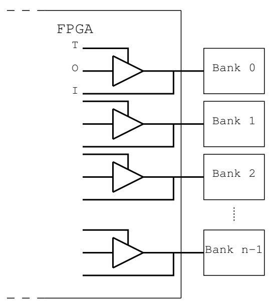
One solution to connect multiple external memory banks is to use multiple EMCs. However, this is a significant waste of FPGA logic since, as mentioned earlier, an EMC is capable of controlling up to four external banks. We will now use an EMC to connect to four of the five ZBT memory banks available on the Multimedia board. The fifth bank can either be added by creating a second EMC instance to control it or by modifying the `opb_emc` peripheral to support more than four banks of memory. Contributing such a modified `opb_emc` peripheral would be much appreciated. It also bears noting that older verions of the `opb_emc` pcore supported up to eight banks of memory.

With the exception of the Chip Enable signals, the EMC was designed such that all the external banks share the same signals (such as the address and data signals). The Multimedia board's design does not accomodate this as the pins for each ZBT bank are connected to separate pads on the FPGA. This is fine for most signals, such as the address lines, since that data can easily be distributed to many of the FPGA pins. However, complications arise for the data signals since they are bidirectional and involve tristate buffering.

The EMC is designed to be used with the memory connected via a shared bus, as shown in Figure 2(a). However, on the Multimedia board, each ZBT has its own set of pins allocated for it on the FPGA. Thus, if more than one ZBT bank is present, the EMC must be adapted to supporte the connection scheme shown in Figure 2(b) wher each ZBT has its own set of tristate buffers for the data lines. Thus, the task of connecting multiple ZBT banks largely consists of adjusting the EMC so that it can adhere to the required connection scheme.



(a) The data signal connection scheme intended for use with the EMC



(b) The required signal connection scheme on the multimedia board

Figure 2: The expected and the necessary connection schemes for external memory

6.1 Step-by-step

1. When sending data out, it is sufficient to send the data to all four banks since only the bank with its Write Enable signal asserted (low) will latch the data. However, when reading data from the ZBTs, the data lines from each bank must be multiplexed using the Output Enable signal to select which bank's data is valid and should be sent to the EMC. A custom core, `zbtio_v1_00_a` has been created to do this.

   Previous verions of this core used the Chip Enable signal to select which data should be routed to the EMC. However, careful reading of the ZBT RAM datasheets shows that the CE signal need not necessarily be asserted during the read cycle, only during the addressing cycle. Take a quick look at the datasheets to confirm this for yourself.

2. Use the IP Catalog tab to add a `zbtio_v1_00_a` instance.

3. Change the parameters of the `opb_emc` instance to include four memory banks and set the address spaces of those banks to be contiguous and non-overlapping. Make sure all four banks are being accessed synchronously and that the Execute Multiple Memory Accesses option is disabled.

4. Make the following connections between the `zbtio` core and the `opb_emc` core. All the connections should be internal in scope. Make sure you also disconnect the bidirectional `Mem_DQ` port on the `opb_emc` peripheral (recall that either the bidirectional port or the three component unidirectional ports should be connected — never both).

zbtio	opb_emc	Net Name
emc_CEN	Mem_CEN	emc_CEN
emc_OEN	Mem_OEN	emc_OEN
emc_DQ_I_out	Mem_DQ_I	emc_DQ_I
emc_DQ_O_in	Mem_DQ_O	emc_DQ_O
emc_DQ_T_in	Mem_DQ_T	emc_DQ_T

5. Connect the following ports of the zbtio core to the external ports indicated. The UCF file will be edited later to map these nets to pins on the FPGA.

zbtio port	External Port
bank0data	fpga_0_ZBT_512Kx32_Mem_Bank0_DQ_pin
bank1data	fpga_0_ZBT_512Kx32_Mem_Bank1_DQ_pin
bank2data	fpga_0_ZBT_512Kx32_Mem_Bank2_DQ_pin
bank3data	fpga_0_ZBT_512Kx32_Mem_Bank3_DQ_pin
bank0ce	fpga_0_ZBT_512Kx32_Mem_Bank0_CEN_pin
bank1ce	fpga_0_ZBT_512Kx32_Mem_Bank1_CEN_pin
bank2ce	fpga_0_ZBT_512Kx32_Mem_Bank2_CEN_pin
bank3ce	fpga_0_ZBT_512Kx32_Mem_Bank3_CEN_pin
bank0oen	fpga_0_ZBT_512Kx32_Mem_Bank0_OEN_pin
bank1oen	fpga_0_ZBT_512Kx32_Mem_Bank1_OEN_pin
bank2oen	fpga_0_ZBT_512Kx32_Mem_Bank2_OEN_pin
bank3oen	fpga_0_ZBT_512Kx32_Mem_Bank3_OEN_pin

6. As you might be noticing, setting up these all these ports is repetitive and tedious using the EDK GUI. You may find it useful to add ports by directly editing the system's MHS file. The file can quickly be accessed via the Project tab in the Project Information Area. Create some of the connections below using the GUI. Using the connections made in the GUI as an example, create the remaining by directly modifying the MHS file. Check the results using the GUI.

   The following ports are the signals shared by all five ZBT banks. Use the Add External Port button to create a system port. The system ports name will refer to a definition in the UCF file. Connect the system port to the indicated port on the opb_emc by ensuring that they share the same net in the Net Name column (do not just type the information in the table).

   Since we are changing the signal names of all four connected banks, you should also take this opportunity to delete the external ports created by the Base System Builder to connect the obp_emc instance to Bank 0 of the ZBT memory.

System port	EMC Port
fpga_0_ZBT_512Kx32_Mem_Bank0_ADV_LDN_pin	Mem_ADV_LDN
fpga_0_ZBT_512Kx32_Mem_Bank0_BEN_pin	Mem_BEN
fpga_0_ZBT_512Kx32_Mem_Bank0_CKEN_pin	Mem_CKEN
fpga_0_ZBT_512Kx32_Mem_Bank0_WEN_pin	Mem_WEN
fpga_0_ZBT_512Kx32_Mem_Bank1_ADV_LDN_pin	Mem_ADV_LDN
fpga_0_ZBT_512Kx32_Mem_Bank1_BEN_pin	Mem_BEN
fpga_0_ZBT_512Kx32_Mem_Bank1_CKEN_pin	Mem_CKEN
fpga_0_ZBT_512Kx32_Mem_Bank1_WEN_pin	Mem_WEN
fpga_0_ZBT_512Kx32_Mem_Bank2_ADV_LDN_pin	Mem_ADV_LDN
fpga_0_ZBT_512Kx32_Mem_Bank2_BEN_pin	Mem_BEN
fpga_0_ZBT_512Kx32_Mem_Bank2_CKEN_pin	Mem_CKEN
fpga_0_ZBT_512Kx32_Mem_Bank2_WEN_pin	Mem_WEN
fpga_0_ZBT_512Kx32_Mem_Bank3_ADV_LDN_pin	Mem_ADV_LDN
fpga_0_ZBT_512Kx32_Mem_Bank3_BEN_pin	Mem_BEN
fpga_0_ZBT_512Kx32_Mem_Bank3_CKEN_pin	Mem_CKEN
fpga_0_ZBT_512Kx32_Mem_Bank3_WEN_pin	Mem_WEN

7. The address port can also be shared by all ZBT banks; however it is passed through a util_bus_split peripheral instance between the opb_emc instance and the physical pins. Connect the Out1 bus of the util_bus_split instance to the following ports:

```
fpga_0_ZBT_512Kx32_Mem_Bank0_A
fpga_0_ZBT_512Kx32_Mem_Bank1_A
fpga_0_ZBT_512Kx32_Mem_Bank2_A
fpga_0_ZBT_512Kx32_Mem_Bank3_A
```

8. Your system should still be configured for the clocking scheme from the single-bank portion of the lab (see Figure 1). Double-check that this is so and connect the clock signals to all four memory banks by creating the following external ports:

Port	Net
fpga_0_ZBT_512Kx32_Mem_Bank0_CLOCK	zbt_clk
fpga_0_ZBT_512Kx32_Mem_Bank1_CLOCK	zbt_clk
fpga_0_ZBT_512Kx32_Mem_Bank2_CLOCK	zbt_clk
fpga_0_ZBT_512Kx32_Mem_Bank3_CLOCK	zbt_clk

9. Open the system's `system.ucf` file. Like the MHS file, it can be accessed from the System tab. Find the file `zbt.ucf` that was extracted into the `lab8\` directory from the ZIP file. Copy the relevant lines of this file to `system.ucf`. This file contains the pin mappings from the Multimedia Board's user guide. (**Note:** the User Guide contains an error — MEMORY_BANK3_ADDR4 should be connected to AF3, **not AF4**. The file `zbt.ucf` contains this correction.)

10. Save the file, build and download the system to the FPGA.

11. Do a quick sanity check with XMD to ensure that the ZBT are being accessed. This can be done with memory write and memory read commands on an address allocated for a ZBT.

12. Write a simple program to test writing and reading to the ZBTs. The easiest way to do this is likely by modifying the memory test application you generated automatically in the first part of this lab. A sample memory test program, Testi_Mem_Four_ZBT, is included in the ZIP file for this module.

13. External memory is often useful for programs which are too large to fit within the internal block RAM. Create a new software project named dhrystone in the system and add the dhrystone code found in the ZIP file for this module. From Wikipedia:

    > Dhrystone is a synthetic benchmark program developed in 1984 by Reinhold P. Weicker intended to be representative of system (integer) programming. The Dhrystone grew to become representative of general processor (CPU) performance until it was superseded by the CPU89 benchmark suite from the Standard Performance Evaluation Corporation, today known as the "SPECint" suite.

14. Double click on the dhrystone software project to modify its Compiler Options. Change the Program Start Address to `0x20200000`. Compile the program.

15. Using XMD, download the program to the ZBTs by typing `dow dhrystone/executable.elf`. Then run the program by typing `run`. Dhrystone should send text to the standard output if it is running.

6.2 Other Examples of Multiple Memory Banks

The previous example is intended to run at 27MHz. There are two other examples of designs that run at 50MHz and 100MHz. The ZIP files are available on the web site as is a diagram of the DCM configurations. Note that the DCMs are configured differently from the example above.

No timing analysis has been done to determine the actual timing margins of these configurations. This is an exercise left for the reader.

No DCM phase shifting is used. With some timing analysis, some DCM shifting might make the design more robust. Experimentally these designs work and are offered as examples.

7 Using the ZBT Memories Without an OPB Bus Interface

If you want to connect hardware directly to the ZBT memories without using the OPB bus, then you will need to use a standalone ZBT controller. There are some links on the EDK Tutorials web page below m08. You should look at Xilinx Application Note XAPP136 and the accompanying to see an overview of how these memories work.

These controllers have not been packaged as cores for EDK.

8 Simulation

If you wish to do simulations that include the ZBT memories, a Verilog behavioural model, `MMboard_ZBT_behmod.v` is included in the ZIP file for this module.

9 Look At Next

Module 13: MBlaze MP3 Decoder
XAPP 136