Version for EDK 8.1i, May 10 2006

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.

2 Background

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

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 8 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 works. However, it does not properly handle the memory clocking. In more complicated designs, the clock synchronization becomes very relevant. 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 all five banks.

3 Requirements

Module 1: Building a Base System
Module 2: Adding Drivers and IP

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 Xilinx DCM (Digital Clock Manager) documentation found in the Virtex II FPGA data sheet.

5 Connecting a Single ZBT Bank

5.1 Step-by-step

1. Create a new system with Base System Builder as you normally would. 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.

2. The created project should include one dcm_0 DCM module already. By selecting IP Catalog tab from Project Information Area, add another instance of dcm_module. The two DCM modules will be used to create the clocking scheme shown in Figure 1. Add the following to the ports table for each of the DCM modules: CLKIN, CLKFB, RST, DSSEN, PSEN, CLK0, LOCKED. Set them all as internal in scope. The default setup should be fine.

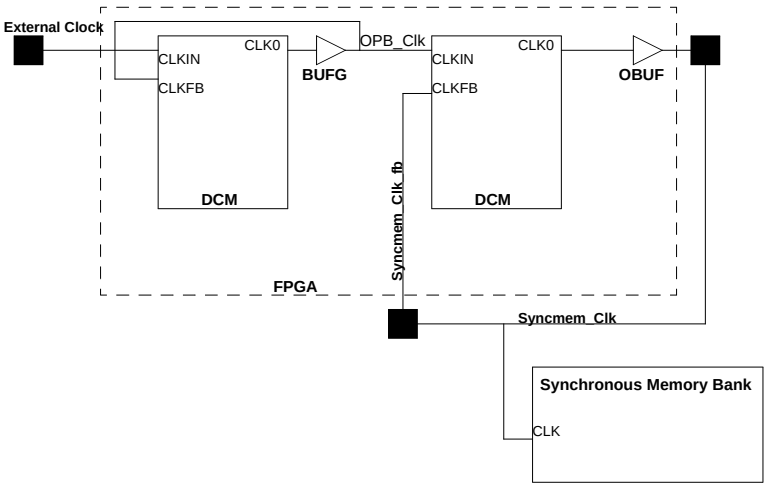


Figure 1: The clocking scheme required for external ZBT memory with clock feedback (Taken from Xilinx's OPB EMC datasheet).

Connect the ports of the first DCM module to the following nets.

RST	dcm_rst
CLKIN	sys_clk_predcm
CLKFB	sys_clk_s
PSEN	net_gnd
DSSEN	net_gnd
CLK0	sys_clk_s
LOCKED	opb_dcm_locked

Connect the ports of the second DCM modules to the following nets.

RST	dcm_rst
CLKIN	sys_clk_s
CLKFB	zbt_dcm_feedback
PSEN	net_gnd
DSSEN	net_gnd
CLK0	zbt_dcm_clk
LOCKED	zbt_dcm_locked

3. Locate the system port that connects the external sys_clk pin to the net sys_clk_s. Change the net to sys_clk_predcm.

4. Using the Add External Port button from System Assembly View, create an output port named ZBT_512Kx32_EMC_CLK_FEEDBACK_OUT and an input port ZBT_512Kx32_EMC_CLK_FEEDBACK_IN. It is not necessary to classify the ports as CLK signals. Connect the two new ports to the zbt_dcm_clk and zbt_dcm_feedback nets respectively. Change the C_CLKIN_PERIOD parameter for each DCM to 37.037. The default values for the other parameters should be sufficient. The DCM will lock when the edges of its CLKFB and CLKIN inputs are aligned.

5. Locate the generated port ZBT_512Kx32_EMC_CLK_OUT. This was generated by System Builder. 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 ZBT_512Kx32_EMC_CLK_FEEDBACK_IN LOC= AE15;
Net ZBT_512Kx32_EMC_CLK_FEEDBACK_IN FAST;
Net ZBT_512Kx32_EMC_CLK_FEEDBACK_OUT LOC= AH14;
Net ZBT_512Kx32_EMC_CLK_FEEDBACK_OUT FAST;
```

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. It can be found in the zip file for this module.

   Copy the `clk_align` core into the system's `pcores` directory. Restart the EDK and find the core in IP Catalog tab.

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	sys_clk_predcm
extend_dcm_reset	sys_rst_preclkalign
dcm0_locked	zbt_dcm_locked
dcm1_locked	opb_dcm_locked
fpga_reset	sys_rst_s
dcm_reset	dcm_rst

9. Locate the system port that connects the external sys_rst pin to the sys_rst_s net. Alter it so that it connects to the sys_rst_preclkalign net.

10. The system is now ready to be built and downloaded. Be aware that Base System Builder connected the system reset signal to User Switch SW0 on the Multimedia board. It is best to download the system with the switch in the reset state. However, the signal must be low in order for XMD to connect to the MicroBlaze.

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 sample application for another quick test of the ZBT memory communications.

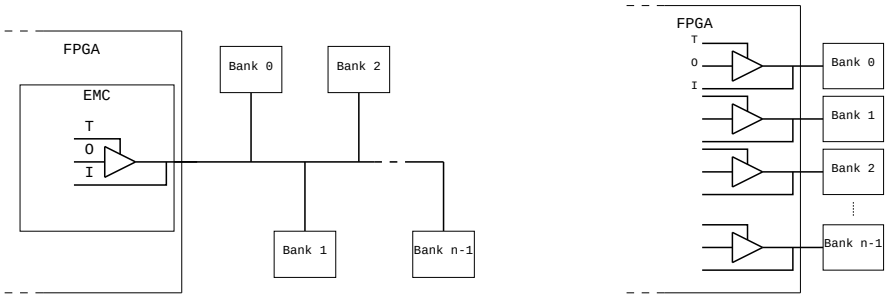
6 Connecting Multiple ZBT banks

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 8 external banks. We will now use an EMC to connect to the 5 ZBT memory banks available on the Multimedia board.

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 is not ideal for this. 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 connection scheme intended to be used with the EMC is shown in Figure 2a. For this scheme to work, the data pins from each ZBT must be connected together externally—outside of the FPGA. However, on the Multimedia board, this is not the case. Each ZBT has its own set of pins allocated for it on the FPGA. This is a non-issue if only one ZBT is being used by the EMC (as we have done thus far). However, if more than one ZBT bank is present the connection scheme shown in Figure 2b is required. Each ZBT bank requires its own set of tristate buffers.

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. Since an altered version of the EMC will be used to connect the external ZBT banks, we will not be using the EMC used in the first part of this lab. Create a new system using Base System Builder that does not contain an EMC. This will avoid any conflict in the .ucf file.

2. Locate the opb_emc core in the hardware library found in the directory within which EDK is installed. It can be found in `EDK/hw/XilinxProcessorIPLib/pcores`. Copy the `opb_emc_v1_10_b` directory into your design's `pcores` directory. Change the version number in the directory's name so that when the altered EMC core is added, it can be distinguished from the original. For example, rename the directory to `opb_emc_v1_10_c`.

3. Examine the files in the core's `hdl/vhdl` directory. Although you may be more familiar with Verilog, you should be able to decipher that according to the HDL, the EMC has 3 ports for the data: Mem_DQ_I, Mem_DQ_O and Mem_DQ_T. The core is configured so that the EDK generates tristate buffers based on these signals. We will be altering the configuration so that these signals are exposed instead.

4. Open `opb_emc_v2_1_0.mpd` in the core's `data` directory. Within the Ports section locate the Mem_DQ port. Notice that it has its DIR option set to INOUT. This causes the EDK to look for the three signals mentioned above and form a set of tristate buffers. Replace the single port declaration with the following three port declarations.

```
PORT Mem_DQ_I = "", DIR = IN, VEC = [0:C_MAX_MEM_WIDTH-1], PERMIT = BASE_USER
PORT Mem_DQ_O = "", DIR = OUT, VEC = [0:C_MAX_MEM_WIDTH-1], PERMIT = BASE_USER
PORT Mem_DQ_T = "", DIR = OUT, VEC = [0:C_MAX_MEM_WIDTH-1], PERMIT = BASE_USER
```

5. When sending data out, it is sufficient to send the data to all 5 banks since only the bank with its Chip Enable signal high will register the data. However, when reading data from the ZBTs, the data lines from each bank must be multiplexed using the Chip Enable signal to select which bank's data is valid and should be sent to the EMC. A custom core has been created to do this.

   Locate the `zbtio` core in the zip file and copy it into your design's `pcores` directory. The zbtio core multiplexes the data input. It also provides the tristate buffers for each chip and splits up some vectored signals into individual lines for convenience.

6. If the system is open in the EDK, close and reopen it so that the newly added cores are available. Use IP Catalog tab to add an instance of opb_emc and zbtio.

   Verify that the opb_emc that was added is in fact the modified one by checking the IP version column of the System Assembly View tab. It should be the version that was changed in the core's directory name (1.10.c if you followed this text's example). Provide the EMC an address space of 0x1F. This is the address space of the controller; not the memory being controlled. Connect the EMC as a slave on the OPB.

7. Add the following ports of the EMC: OPB_Clk, Mem_WEN, Mem_OEN, Mem_DQ_T, Mem_DQ_O, Mem_DQ_I, Mem_CKEN, Mem_CEN, Mem_BEN, Mem_ADV_LDN, and Mem_A. Connect OPB_Clk to sys_clk_s. All the ports should be internal in scope.

8. Add all the ports of the zbtio core. Make the following connections between the zbtio core and the opb_emc core. (Note that for them to be connected, they must share the same net in the Net Name column. The information in the table below cannot be simply typed in.) All the connections should be internal in scope.

zbtio	opb_emc
emcchipselect	Mem_CEN
emcoen	Mem_OEN
emc2zbt_data	Mem_DQ_0
zbt2emc_data	Mem_DQ_I
tribuff_enable	Mem_DQ_T

9. Connect the following ports of the zbtio core to the external ports indicated. The connections should be external in scope as they refer to pins that will be listed in the .ucf file (it will be edited later). Note that a value in the range column is required for the multi-bit data signals.

zbtio port	External Port	Range
bank0data	MEMORY_BANK0_DATA	[0:31]
bank0ce	MEMORY_BANK0_CEN	
bank0oen	MEMORY_BANK0_OEN	
bank1data	MEMORY_BANK1_DATA	[0:31]
bank1ce	MEMORY_BANK1_CEN	
bank1oen	MEMORY_BANK1_OEN	
bank2data	MEMORY_BANK2_DATA	[0:31]
bank2ce	MEMORY_BANK2_CEN	
bank2oen	MEMORY_BANK2_OEN	
bank3data	MEMORY_BANK3_DATA	[0:31]
bank3ce	MEMORY_BANK3_CEN	
bank3oen	MEMORY_BANK3_OEN	
bank4data	MEMORY_BANK4_DATA	[0:31]
bank4ce	MEMORY_BANK4_CEN	
bank4oen	MEMORY_BANK4_OEN	

10. 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 5 ZBT banks. Use the Add 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).

System port	EMC Port	Range
MEMORY_BANK0_ADDR	Mem_A	[0:31]
MEMORY_BANK0_ADV_LDZ	Mem_ADV_LDN	
MEMORY_BANK0_BEN	Mem_BEN	[0:3]
MEMORY_BANK0_CLKEN	Mem_CKEN	
MEMORY_BANK0_WEN	Mem_WEN	
MEMORY_BANK1_ADDR	Mem_A	[0:31]
MEMORY_BANK1_ADV_LDZ	Mem_ADV_LDN	
MEMORY_BANK1_BEN	Mem_BEN	[0:3]
MEMORY_BANK1_CLKEN	Mem_CKEN	
MEMORY_BANK1_WEN	Mem_WEN	
MEMORY_BANK2_ADDR	Mem_A	[0:31]
MEMORY_BANK2_ADV_LDZ	Mem_ADV_LDN	
MEMORY_BANK2_BEN	Mem_BEN	[0:3]
MEMORY_BANK2_CLKEN	Mem_CKEN	
MEMORY_BANK2_WEN	Mem_WEN	
MEMORY_BANK3_ADDR	Mem_A	[0:31]
MEMORY_BANK3_ADV_LDZ	Mem_ADV_LDN	
MEMORY_BANK3_BEN	Mem_BEN	[0:3]
MEMORY_BANK3_CLKEN	Mem_CKEN	
MEMORY_BANK3_WEN	Mem_WEN	
MEMORY_BANK4_ADDR	Mem_A	[0:31]
MEMORY_BANK4_ADV_LDZ	Mem_ADV_LDN	
MEMORY_BANK4_BEN	Mem_BEN	[0:3]
MEMORY_BANK4_CLKEN	Mem_CKEN	
MEMORY_BANK4_WEN	Mem_WEN	

11. Since the opb_emc core is able to control several types of external memories, parameters must be used to ensure proper commutation with the ZBTs. Customize the following parameters of the opb_emc.

C_NUM_BANKS_MEM	5
C_INCLUDE_DATAWIDTH_MATCHING_0	0
C_INCLUDE_DATAWIDTH_MATCHING_1	0
C_INCLUDE_DATAWIDTH_MATCHING_2	0
C_INCLUDE_DATAWIDTH_MATCHING_3	0
C_INCLUDE_DATAWIDTH_MATCHING_4	0
C_SYNCH_MEM_0	1
C_SYNCH_MEM_1	1
C_SYNCH_MEM_2	1
C_SYNCH_MEM_3	1
C_SYNCH_MEM_4	1
C_OPB_CLK_PERIOD_PS	37037
C_DEV_MIR_ENABLE	0
C_MEM0_BASEADDR	0x80600000
C_MEM0_HIGHADDR	0x807fffff
C_MEM1_BASEADDR	0x80800000
C_MEM1_HIGHADDR	0x809fffff
C_MEM2_BASEADDR	0x80a00000
C_MEM2_HIGHADDR	0x80bfffff
C_MEM3_BASEADDR	0x80c00000
C_MEM3_HIGHADDR	0x80dfffff
C_MEM4_BASEADDR	0x80e00000
C_MEM4_HIGHADDR	0x80ffffff

The BASEADDR and HIGHADDR parameters define the address space associated with each bank.

12. As was done when connecting a single external bank, the ZBTs must be driven by the clocking scheme shown in Figure 1.

    Follow the directions in "Connecting a Single ZBT Bank" to implement the clocking scheme. However, the .ucf file that will be used will have a different naming convention. Create the following external ports with the Add External Port button in the System Assembly View. These ports will replace the ports beginning with ZBT_512Kx32 that were used when connecting a single bank.

Port	Net
MEMORY_CLOCK_FB_IN	zbt_feedback
MEMORY_CLOCK_FB_OUT	zbt_clk
MEMORY_BANK0_CLK	zbt_clk
MEMORY_BANK1_CLK	zbt_clk
MEMORY_BANK2_CLK	zbt_clk
MEMORY_BANK3_CLK	zbt_clk
MEMORY_BANK4_CLK	zbt_clk

13. Open the system's `data\system.ucf` file. Like the MHS file, it can be accessed from the System tab. Find the file `zbt.ucf` in the zip file. Append the contents of this file to `data\system.ucf`. This file contains the pin mappings written in the Multimedia Board's user guide. (For future reference, the User guide contains an error at Memory_Bank3_Addr4 assigning it to AF4 rather than AF3. The file `zbt.ucf` contains this correction.)

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

15. 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.

16. Write a simple program to test writing and reading to the ZBTs. Below is an example program. It writes each word's (MicroBlaze) address to the ZBT and reads it back. Check `xparameters.h` for the definitions suitable for your system.

```
#include <stdio.h>
#include "xparameters.h"

#define UINT unsigned int

int main()
{
  UINT * mem = (UINT*)XPAR_OPB_EMC_0_MEM0_BASEADDR;

  xil_printf("Writing to Mem\n\r");
  while(mem <= (UINT*)XPAR_OPB_EMC_0_MEM4_HIGHADDR)
    {
      *mem = ((UINT) mem );
      mem++;
    }

  xil_printf("Reading from mem\n\r");
  mem = (UINT*)XPAR_OPB_EMC_0_MEM0_BASEADDR;

  while(mem <= (UINT*)XPAR_OPB_EMC_0_MEM4_HIGHADDR)
    {
      if(*mem != ((UINT) mem))
xil_printf("Error at 0x%X = 0x%X\n\r",mem,*mem);
      mem++;
    }
  xil_printf("Done\n\r");
}
```

17. 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. The dhrystone program is a large program which can be run off the ZBTs.

18. Double click on the new software project's Compiler Options. Change the Program Start Address to 0x80600000. Compile the program.

19. 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 xapp136 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