Interfacing Micron DDR2 Memories to Xilinx Spartan-3A/AN FPGAs: A Step-By-Step Guide

APPLICATION NOTE

AN086 February 18, 2008 (Version 1.0)

Introduction

Many modern applications use FPGAs to implement complex system level building blocks. These building blocks can contain processors, DSPs, bus interface and even video and audio functions. On-FPGA memory is available for use in small FIFOs and scratch pad memories, but many applications require a significant amount of off-chip memory to support advanced processing algorithms or data buffering. Video applications, for example, may need to have a large frame buffer to store images during processing. External to the FPGA memory devices, like the Micron MT47H32M16 DDR2 memory, can provide the additional storage required by these types of applications.

Interfacing an off-chip DDR2 memory device to a Xilinx Spartan-3A/3AN FPGA involves a variety of design considerations. This application note provides a step-by-step guide for a design engineer who needs to quickly and efficiently create an FPGA design using the Spartan-3A/3AN FPGA and a Micron DDR2 memory device.

Topics in this application note are organized by design task (Device Selection, FPGA Design, and Board Design) and each topic is a stand-alone section, with a short introduction or overview, followed by the step-by-step design guidelines. The reader may skip over sections and go directly to the topic of interest, or start at the beginning and follow each topic in order to quickly go through the entire design process, step-by-step. The topics covered in this application note include the following:

1) Device Selection
   a. Micron DDR2 Memory
      i. Memory Functional Overview
      ii. Pin Descriptions
      iii. Initialization
      iv. Devices supported by Spartan-3A/AN Memory Interface Generator
   b. Xilinx Spartan-3A/AN FPGAs
      i. Spartan-3A/AN Functional Overview
      ii. Performance Targets for Spartan-3A/AN Memory Interface Generator
2) FPGA Design
   a. Using the Xilinx Memory Interface Generator (MIG)
      i. MIG Overview
      ii. Example Design- Step-by-Step (Included in Appendix 1)
      iii. Results of MIG (Included in appendix 1)
   b. Memory Controller Interface
      i. Memory Controller Overview
      ii. Functional Description
      iii. Pin Descriptions
3) Board Design Considerations
   a. Pin Assignment Considerations
   b. Signal Termination Considerations
   c. Timing Related Layout Considerations
   d. Simultaneous Switching Output Considerations

A significant amount of supporting documentation is referenced in this application note and the reference section at the end of this application note provides links to on-line versions of each reference, making it easy to locate additional information.

Device Selection

The first phase of the design project is device selection. Here the designer decides on the key components to be used in the design. In this application note we will review some of the options when selecting a DDR2 memory device and will, in particular, select between a Micron DDR2 Memory module and a single chip solution from the Micron Technologies MT47H Family. We will also look at device options within the Xilinx Spartan-3A/AN Family for a target device that fits our requirements. This combination of a Xilinx Spartan-3A/AN FPGA and a Micron DDR2 memory device is excellent for a wide range of applications in the consumer, industrial, communications and networking industries. The following sections provide a quick overview of device options and will determine specific devices for our example application using the performance of the memory interface as the key criteria.

Micron DDR2 Memory Solutions

Micron offers an exceptionally wide range of DDR2 memory options that include memory modules and components. Memory modules offer a significant increase in memory capacity- both width and depth- since typically 8 or more devices are mounted on a single module. For high capacity requirements a module usually is a significant savings in terms of board-space as well. Because module form factor and functionality are standard it is easy to replace a module with one of higher or lower capacity as well. This makes it easy to create a common platform for a design that can be used in both high-end and low-end applications by increasing memory size and speed.

Individual components are a good choice when the depth and width of the memory are constrained, board space is at a premium, low power is important and FPGA pins are limited. As long as the performance and memory capacity are satisfied by an individual component it is usually a good choice. Memory components are pin compatible over capacity and performance ranges, so a single design can target multiple applications by creating manufacturing options. Because components must be soldered to the board, unlike modules which are easily plugged into or removed from a board, build options based on components is a bit more complex and inventory intensive, but is still a benefit in many cases.

In our target application we will use a single device because our capacity requirement is limited and we have a limited number of FPGA pins available. A single device will best satisfy these requirements. We will select our device from the Micron MT47H Family since it has the high-performance and low-power requirements that are also desirable in our application.

Micron DDR2 Memory

The Micron Technologies MT47H Family of Double Data Rate 2 (DDR2) SDRAM are high-speed CMOS, dynamic random access memory containing from 256Mb to 2Gb memory bits, internally configured as 4-bank or 8-bank DRAM. They use a double data rate architecture to achieve high-speed operation. A double data rate interface is designed to transfer two data words per clock cycle at the device outputs. A single read or write access for the DDR2 SDRAM effectively consists of a single 4n-bit-wide (where n is the width of the data word on the device outputs), one-clock-cycle data transfer at the internal DRAM core and four corresponding n-bit-wide, one-half-clock-cycle data transfers at the device outputs.

Micron DDR2 Memory Functional Overview

A bidirectional data strobe (DQS, DQS#) is transmitted externally, along with data, for use in data capture at the receiver. DQS is a strobe transmitted by the DDR2 SDRAM during READs and by the memory controller during WRITEs. DQS is edge-aligned with data for READs and center-aligned with data for WRITEs. The x16 offering has two data strobes, one for the lower byte (LDQS, LDQS#) and one for the upper byte (UDQS, UDQS#).

MT47H Family SDRAM operates from a differential clock (CK and CK#); the crossing of CK going HIGH and CK# going LOW will be referred to as the positive edge of CK. Commands (address and control signals) are registered at every positive edge of CK. Input data is registered on both edges of DQS, and output data is referenced to both edges of DQS as well as to both edges of CK. Read and write accesses to the DDR2 SDRAM are burst-oriented; accesses start at a selected location and continue for a programmed number of locations in a programmed sequence. Accesses begin with the registration of an ACTIVE command, which is then followed by a READ or WRITE command. The address bits registered coincident with the ACTIVE command are used to select the bank and row to be accessed. The address bits registered coincident with the READ or WRITE command are used to select the bank and the starting column location for the burst access.

The MT47H SDRAM Family provides for programmable read or write burst lengths of four or eight locations. It supports interrupting a burst read of eight with another read or a burst write of eight with another write. An auto precharge function may be enabled to provide a self-timed row precharge that is initiated at the end of the burst access. As with standard DDR SDRAMs, the pipelined, multibank architecture of DDR2 SDRAMs allows for concurrent operation, thereby providing high, effective bandwidth by hiding row precharge and activation time.

All inputs are compatible with the JEDEC standard for SSTL_18 and all full drive-strength outputs are SSTL_18-compatible.

Input and Output Descriptions

Descriptions of the device input and output signals for the MT47H32M16 are listed in Table 1 along with a detailed description of the operation of each signal. (Descriptions for other family devices are the same except for the number of address, data and associated signals. We will use the MT47H32M16 IO assignments because that is the device we will eventually select for our example design).This information is provided for review and for reference when determining the specifics of the FPGA memory interface during the FPGA Design step. A more complete description of the operation of the MT47H32M16 is provided in the datasheet referenced at the end of this application note.

Name	Type	Description
ODT	Input	On-die termination: ODT (registered HIGH) enables termination resistance internal to the DDR2 SDRAM. When enabled, ODT is only applied to each of the following balls: DQ0–DQ15, LDM, UDM, LDQS, LDQS#, UDQS, and UDQS#. The ODT input will be ignored if disabled via the LOAD MODE command.
CK, CK#	Input	Clock: CK and CK# are differential clock inputs. All address and control input signals are sampled on the crossing of the positive edge of CK and negative edge of CK#. Output data (DQs and DQS/ DQS#) is referenced to the crossings of CK and CK#.
CKE	Input	Clock enable: CKE (registered HIGH) activates and CKE (registered LOW) deactivates clocking circuitry on the DDR2 SDRAM. The specific circuitry that is enabled/disabled is dependent on the DDR2 SDRAM configuration and operating mode. CKE LOW provides pre-charge power-down mode and SELF REFRESH operation (all banks idle), or active power-down (row active in any bank). CKE is synchronous for power-down entry, power-down exit, output disable, and for self refresh entry. CKE is asynchronous for SELF REFRESH exit. Input buffers (excluding CK, CK#, CKE, and ODT) are disabled during power-down. Input buffers (excluding CKE) are disabled during self-refresh. CKE is an SSTL_18 input but will detect a LVCMOS LOW level once VDD is applied during first power-up. After VREF has become stable during the power on and initialization sequence, it must be maintained for proper operation of the CKE receiver. For proper SELF REFRESH operation, VREF must be maintained.
CS#	Input	Chip select: CS# enables (registered LOW) and disables (registered HIGH) the command decoder. All commands are masked when CS# is registered HIGH. CS# provides for external bank selection on systems with multiple ranks. CS# is considered part of the command code.
RAS#, CAS#, WE#	Input	Command inputs: RAS#, CAS#, and WE# (along with CS#) define the command being entered.
LDM, UDM (DM)	Input	Input data mask: DM is an input mask signal for write data. Input data is masked when DM is concurrently sampled HIGH during a WRITE access. DM is sampled on both edges of DQS. Although DM balls are input-only, the DM loading is designed to match that of DQ and DQS balls. LDM is DM for lower byte DQ0–DQ7 and UDM is DM for upper byte DQ8–DQ15.
BA0–BA1	Input	Bank address inputs: BA0–BA1 indicate to which bank an ACTIVE, READ, WRITE, or PRECHARGE command is being applied. BA0–BA1define which mode register including MR, EMR, EMR(2), and EMR(3) is loaded during the LOAD MODE command.
A0–A3 A4–A7 A8–A11 A12	Input	Address inputs: Provide the row address for ACTIVE commands, and the column address and auto pre-charge bit (A10) for READ/WRITE commands, to select one location out of the memory array in the respective bank. A10 sampled during a PRECHARGE command determines whether the PRECHARGE applies to one bank (A10 LOW, bank selected by BA1–BA0) or all banks (A10 HIGH). The address inputs also provide the op-code during a LOAD MODE command.
DQ0–DQ3 DQ4–DQ7 DQ8–DQ11 DQ12–DQ15	I/O	Data input/output: Bidirectional data bus.

UDQS, UDQS#	I/O	Data strobe for upper byte: Output with read data, input with write data for source synchronous operation. Edge-aligned with read data, center-aligned with write data. UDQS# is only used when differential data strobe mode is enabled via the LOAD MODE command.
LDQS, LDQS#	I/O	Data strobe for lower byte: Output with read data, input with write data for source synchronous operation. Edge-aligned with read data, center-aligned with write data. LDQS# is only used when differential data strobe mode is enabled via the LOAD MODE command.
DQS, DQS#	I/O	Data strobe: Output with read data, input with write data for source synchronous operation. Edge-aligned with read data, center aligned with write data. DQS# is only used when differential data strobe mode is enabled via the LOAD MODE command.
V	Supply	Power supply: 1.8V ±0.1V.
VDDL	Supply	DLL power supply: 1.8V ±0.1V.
VDDQ	Supply	DQ power supply: 1.8V ±0.1V. Isolated on the device for improved noise immunity.
VREF	Supply	SSTL_18 reference voltage.
VSS	Supply	Ground.
VSSQ	Supply	DQ ground. Isolated on the device for improved noise immunity.

Table 1: MT47H32M16 Pin Descriptions

Initialization

The initialization of the MT47H32M16 requires a 16 step process that includes applying power in the correct sequence and within specified timing parameters, precharging the memory array, initializing the mode register, issuing refresh commands and waiting for the specified DDL initialization time. (The exact specification of these steps can be found in the MT47H32M16 data sheet referenced at the end of this document). The Spartan-3A/AN FPGA DDR2 Memory Controller IP Core is designed to implement the specifics of these initialization requirements.

Xilinx Spartan-3 Generation of FPGAs

The Xilinx Spartan-3 generation of FPGAs are optimized for a variety of applications. Spartan-3 devices are targeted for high density, high pin count designs and are ideal for data processing applications. Spartan-3E devices are optimized for logic intensive designs and are ideal for co-processing and embedded control applications. Spartan-3A DSP devices are optimizes for DSP designs and are ideal for ultrasound, consumer video and video surveillance applications. Spartan-3A devices are optimized for IO intensive designs and are ideal for bridging, differential signaling and memory interface applications. Spartan-3AN devices are non-volatile and are ideal for any application with design security, low cost and high-volume requirements.

Xilinx Spartan-3A/AN FPGA Family Overview

Xilinx Spartan-3A/AN devices are excellent selections for memory interface solutions because of the high IO counts supported. Memory interfaces are typically IO 'bound' (they tend to run out of IOs prior to running out of logic resources) and as such fit nicely into the Spartan-3A/AN devices. If the target application is security conscious, the Spartan-3AN devices are a good fit with their non-volatile characteristics. Reverse engineering, cloning and other security risks are mitigated with the Spartan-3AN devices. The logic capabilities, IO capabilities and other 'standard' FPGA features are the same between the Spartan-3A and -3AN devices so for the purpose of this application note we will select a Spartan-3A device as our target, but the process of generating a memory interface design would be the same for either type of device (just select a Spartan-3AN device instead of a Spartan-3A in the Core Generator tool).

The selection of the target Spartan-3A device depends on both the IO requirements and the logic requirements. For our target application let's assume we need over 350 IOs and over 10K logic cells. Table 2 below shows the range of capabilities of the Spartan-3A devices. The XC3S700A device is the smallest that meets both of our key requirements. (During the memory interface generation process we will have the option to design the interface in such a way that we can migrate to another device with the same package footprint, with a different amount of logic capacity. This can be useful if the design requirements might change late in the design process. This is covered in the memory interface generation section of this app note.)

Device	Logic Cells	IOs
XC3S50A	1584	144
XC3S200A	4032	248
XC3S400A	8064	311
XC3S700A	13,248	372
XC3S1400A	25,344	502

Table 2: Spartan-3A Logic Cell and IO Capabilities

Spartan-3 Memory Interface Performance Targets

Spartan-3 generation FPGA devices can support a variety of interface timing and memory device speed grades. The interface performance level is determined by both the memory device speed grade and the Spartan-3 device speed grade. Typically the selection of the device family and speed grade will determine the minimum and maximum memory interface performance. Table 3 below lists the various Xilinx Spartan-3 generation speed grades and the minimum and maximum performance of the memory interface for a DDR2 memory subsystem. (These ranges assume a CAS latency of 3). Assuming the target for our memory interface frequency is 133MHz we see that the -4 speed grade will work for a Spartan-3A device. We will use the xcS3700A-4 device in our example application.

Device/Speed Grade	Minimum Frequency	Maximum Frequency
Spartan-3/-4 Speed Grade	125	133MHz
Spartan-3/-5 Speed Grade	125	166MHz
Spartan-3A/AN/A DSP -4 Speed Grade	125	133MHz
Spartan-3E	Not Supported	Not Supported

Table 3: Spartan-3 Device Memory Interface Minimum and Maximum Design Frequency Ranges

Micron Technologies DDR2 Device Support

A wide range of Micron DDR2 Memory devices are supported in the Xilinx Memory Interface Generator (MIG) tool. (MIG is described in detail in the FPGA Design section of this document) . Table 4 below identifies each of the supported devices, the storage size of the device, the speed grades supported and the data rate of each speed grade. For the purpose of this application note we will use the MT47H32M16 device. The memory interface frequency we are targeting, 133MHz, allows us to use the 5E speed grade (400MHz maximum data rate) of the MT47H32M16 with CAS Latency of 3.

Device Part Number	Size (Mb)	Speed Grades	Data Rate (MHz), (CL=4)	Data Rate (MHz), (CL=3)
MT47H64M4XX	256	3,37E,5E	533, 533, 400	400
MT47H128M4XX	512	3,37E,5E	533, 533, 400	400
MT47H256M4XX	1024	3,37E,5E	533, 533, 400	400
MT47H32M8XX	256	3,37E,5E	533, 533, 400	400
MT47H64M8XX	512	3,37E,5E	533, 533, 400	400
MT47H128M8XX	1024	3,37E,5E	533, 533, 400	400
MT47H16M16XX	256	3,37E,5E	533, 533, 400	400
MT47H32M16XX	512	3,37E,5E	533, 533, 400	400
MT47H64M16XX	1024	3,37E,5E	533, 533, 400	400

Table 4: Micron Technologies DDR2 Memory Devices Supported in MIG

FPGA Design

Once the key devices have been selected the FPGA stage of the design can be started. During this phase the designer will need to create a memory interface design that can be targeted to the selected FPGA. Xilinx has significantly simplified this stage of the design by creating a Memory Interface Generator (MIG) tool that guides the designer thru the process of creating a memory interface optimized for the target FPGA and memory devices. MIG creates the RTL code (VHDL or Verilog), the constraints used to guide logic cell placement, timing, and IO assignments, documentation to help support the design, and a test bench for use during design simulation. The balance of this section of this application note will go through a step-by-step guide of using the MIG to create a memory interface for our target devices.

Using MIG

The MIG is a tool used to generate memory interfaces for Xilinx FPGAs. MIG generates Verilog or VHDL RTL design files, UCF constraints, and script files. The script files are used to run synthesis, map, and place and route (par) for the selected configuration. Creating memory interfaces is very easy when using the MIG tool. The designer can specify all the important parameters for a specific memory interface (address range, etc) and the MIG tool automatically creates the required design information used in conjunction with a Xilinx DDR2 Memory Controller IP Core.

Once MIG has been run it creates a complete RTL memory controller and interface. Shown below is an example of the various parameters selected by the user in MIG and produced as an output report by MIG. Many of the parameters will be familiar to the designer and are self explanatory. For the more interested reader a detailed description and a step-by-step example of a complete design using the MIG is given at the appendix at the end of this document. The step-by-step guide takes the reader through the complete design process and after completing it the reader will be able to successfully complete their own memory interface design using a variety of Xilinx FPGA and Micron DDR2 memory devices.

```
Datasheet
Generated by mig Version 2.0 on Thu January 17 12:8 2008

*************   INPUTS GIVEN   ***********
 PART                      : xc3s700afg484
 Frequency in MHz          : 133
 Speed grade               : -4
 No of controllers         : 1
 Synthesis tool            : XST
 HDL                       : vhdl
 Implementation Options:
 DCM used                  : 1
 Add test bench            : 0
 Number of write pipelines: 4

************************************
 Generating interface for controller 0
 Memory type               : DDR2_SDRAM/Components/
                             MT47H32M16XX-3
 Bits per strobe           : X8
 Banks for Data            : 1
 Data bits                 : 8
 Banks for addr & cntrl    : 3
 Banks for System Control  : 0,2
 Banks for System Clock    : 0,2
 Row address bits          : 13
 Column address bits       : 10
 Bank address bits         : 2
*************************************************
Design Parameters :
       Mode Register :
            Burst Length           : 4(010)
            Burst Type             : sequential(0)
            CAS Latency            : 3(011)
            Mode                   : normal(0)
            DLL Reset              : yes(1)
            PD Mode                : fast exit(0)
            Write Recovery         : 3(010)
      Extended Mode Register :
            DLL Enable             : Enable-Normal(0)
            Output Drive Strength  : Fullstrength(0)
            RTT (nominal) - ODT    : 75ohms(01)
            Additive Latency (AL)  : 0(000)
            OCD Operation          : OCD Exit(000)
            DQS# Enable            : Enable(0)
            RDQS Enable            : Disable(0)
            Outputs                : Enable(0)
```

For the reader interested in how to use the MIG for a variety of memory architectures (DDR SDRAM, DDRII SRAM, DDR2 SDRAM, QDRII SRAM, and RLDRAM II) and for different Xilinx FPGA families (Spartan-3/3E/3A/3AN/3A DSP FPGA Families and Virtex-4/5 FPGA Families) is given in the MIG Users Guide referenced at the end of this document.

Memory Controller Interface

The memory controller created by MIG must be integrated into the rest of the design. Typically this involves connecting the memory controllers user interface to the portion of the design that 'drives' the memory controller. This could be a simple as a processor bus or more complicated and involve significant optimization of the data access sequence to memory. Any user logic will communicate with the memory controller over the same set of signals however. The balance of this section provides a quick overview of these interface signals and a short functional description of the operation of the controller. More details of the controller are available in the Xilinx xapp454 application note listed in the reference section of this application note.

Signal Descriptions

The interface signals used to interact with the memory controller are given in Table 5 below. The signal name is listed along with the type of signal (input or output) and a short description of each signal. A short description of the operation of the memory controller follows the table.

Name	Type	Description
SYS_CLK, SYS_CLKb	Input	Differential system clock inputs.
reset_in_n	Input	Active low system reset signal
Burst_done	Input	Indicates that the current burst transaction is done
User_command_register	Input	Memory Command
User_data_mask	Input	Data mask for partial write operations
User_input_data	Input	Data to write to memory
User_input_address	Input	Address of memory location
Init_done	Output	Indicates memory initialization is complete
Ar_done	Output	Indicates memory auto refresh is complete
Auto_ref_req	Output	Indicates memory refresh is required
User_cmd_ack	Output	Command acknowledge for the user read or write command
Clk_tb	Output	Provided to the user for synchronization
Clk90_tb	Output	Provided to the user for synchronization
Sys_rst_tb	Output	Provided to the user for synchronization
Sys_rst90_tb	Output	Provided to the user for synchronization
Sys_rs180_tb	Output	Provided to the user for synchronization
User_data_valid	Output	Indicates data read from memory is available
User_output_data	Output	Data read from memory

Table 5: Memory Controller User Interface Signals

Functional Description

The memory controller user interface simplifies the process or interacting with the off-chip DDR2 memory device. Low level tasks like initialization, refresh timing, read and write data capture and other details are simplified by the memory controller and the user can focus on higher level actions like reading and writing.

There are commands supported by the controller and these are specified on the user command signals. The commands are Nop, memory initialization, auto-refresh, write, read and load mode.

Nop:	No operation.
Memory Initialization:	Used to initialize the memory interface.
Auto Refresh:	Used to execute a refresh command.
Write:	Executes a sequential block write to memory.
Read:	Executes a sequential read from memory.
Load Mode:	Loads user data to the memory mode register.

The user indicates the operation desired on the user_command_ register signals. When clocked by the SYS_CLK (rising edge) the command is executed by the controller. The Memory Initialization command is a multiple cycle command and is complete when the controller asserts the Int_done signal. The Auto Refresh command is a multi-cycle command and is complete when the Ar_done signal is asserted.

The Write command begins with the user indicating a write on the user_command_register. The controller acknowledges receipt of the command when the user_cmd_ack signal is asserted. After the acknowledge is received the user can place the address and data on the controller inputs. The user then asserts burst-done to indicate the write cycle is complete. (The exact timing of each of these signals is given in detail in xapp454. For the purposes of this overview these details will be skipped).

The Read command begins with the user indicating a read on the user_command_register. The controller acknowledges receipt of the command when the user_cmd_ack signal is asserted. After the acknowledge is received the user can place the address on the controller inputs. The user then asserts burst-done to indicate the address burst is complete. The controller responds with the user_ valid_data signal being asserted and the read data is presented on the data outputs. The user_valid_data signal is de-asserted by the controller when the last data word is presented to the user interface. (The exact timing of each of these signals is given in detail in xapp454. For the purposes of this overview these details will be skipped).

After the user connects the memory interface to their design and simulates the application successfully the FPGA design portion of the project is completed. The next step is the Board Design (In many cases some of the FPGA and Board Design activities can be done in parallel and this is usually advantageous. For the purposes of this application note we will assume that these phases are done sequentially however).

Board Design

During the board design portion of the memory subsystem design process there are several key design considerations that need to be addressed. These considerations can be grouped as follows: Pin Assignment Considerations, Signal Termination Considerations, Timing Related Layout Considerations, and Simultaneous Switching Output Considerations. Following the discussion of these considerations is an example memory subsystem design showing details of the device placement, layout and signal routing used in the design. This illustrates some of the trade-offs involved in implementing a real-world system and shows how the designer can use the board layout considerations given in this document to help simplify the process of evaluating various design trade-offs.

Pin Assignment Considerations

The Xilinx MIG tool generates pin assignments for a memory interface based on certain rules depending on the design technique, but does not provide the best possible pin assignment for every board implementation. For example, during layout it might be necessary to swap pin locations depending on the number of layers available and the interface topology.

The best way to change the pin assignment is to first apply changes on a byte basis then swap bits within a byte. Calculate the PCB loopback length, if required, after pin swapping and trace matching. The following rules of thumb are provided to help designers determine how pins can be swapped. Any changes to the pin assignments require modifications to the UCF provided by MIG and might require changes to the source code depending on the changes made.

1) For all MIG Spartan designs, the address and control pins can be swapped with each other as needed to avoid crossing nets on the printed circuit board.

2) Data lines used to read data from a memory are placed in the same bank as their associated strobe. The strobe is used to capture the associated data in the FPGA and needs to be located close to the data to improve timing.

3) For memory interfaces that do not provide a data valid signal to indicate when the read data is valid, a data valid signal is to be provided on the PCB. This loopback is used as a write-enable signal for the Read Data FIFOs. A strobe is used to latch the data. Two pins are needed per design: one to output the signal and one to input the return signal. The length of the loopback is:
    a. PCB loopback = CLK delay to memory + strobe delay

4) Address and control signals are to be placed together in the same bank or placed in banks near each other to minimize the route delays for these signals inside the FPGA.

5) Spartan-3/3E/3A designs have specific pin placement rules that are followed by MIG to generate the pin assignments. A byte can be swapped with another byte as long as all the necessary signals associated with that byte are changed (strobe, data, and data mask).

6) Within a byte, only even-numbered bits can be swapped with other even-numbered bits (with the same rule applying for odd-numbered bits) because two copies of the DQS strobe are internally generated: one copy for even-numbered bits and one for odd-numbered bits. Each copy is delayed a specific amount relative to the placement of the even (or odd) Read Data FIFOs. As an example, in a byte bits 0 and 2 can be swapped but bits 0 and 1 cannot be swapped. The UCF provided by MIG contains LOC constraints that must be changed to match the swapped pin assignments.

7) The local clocking scheme used to capture data in all MIG Spartan-3 family memory designs requires place and route (PAR) template routes to properly place the delayed strobe and data bits. Template routing is required to properly route the delayed strobe (dqs*_delayed_col*), as well as the data (dq bits). For the data bits to be routed properly, the environment variable XIL_ROUTE_ENABLE_DATA_CAPTURE must be enabled when PAR is run. This environment variable is set in the implementation script file ise_flow.bat provided in the /par MIG output directory. The user must set this environment variable when running the design using the GUI mode from create_ise.bat.

8) The DQ and DM bits of a byte are to be placed in the same bank as the associated DQS. The DQ bits must be kept close together for better routing.

9) Address and control signals are to be placed in the same bank or placed in banks near each other. If all control signals cannot fit in one bank, CK, ODT, and CKE should be selected first for placement in another bank.

10) Each bank that contains DQ/DQS/DM signals needs a loopback signal. If a bank is pin-limited and there is a need to free up a few pins, the following actions can be considered:
    a. The CKE signals can be tied together for multiple devices.
    b. For DIMMs, non-critical features need not be implemented, such as PAR_IN/PAR_OUT and the SPD interface (SA, SPD, SCL).

Signal Termination Considerations

For the proper operation of the memory interface specific signal termination guidelines need to be followed. The following guidelines apply to termination for Spartan-3A/AN FPGAs when interfacing to DDR2 SDRAM:

1) For DIMMs, the CK signals are to be terminated by a 5 pF capacitor between the two legs of the differential signal instead of the 100Ω resistor termination because these signals are already terminated on each DIMM.
2) The ODT and CKE signals are not terminated. These signals are required to be pulled down during memory initialization with a 4.7 kΩ resistor connected to GND.
3) ODT, which terminates a signal at the memory, applies to the DQ/DQS/DM signals only. If ODT is used, the Mode register must be set appropriately in the FPGA design.
4) To save board space, DCI at the FPGA and ODT at the memory can be used to minimize the number of external resistors on the board

Timing Related Board Layout Considerations

Once signals have been assigned and trace geometry has been optimized the trace lengths of critical signals need to be examined to make sure timing constraints are met. For DDR2 memory interface designs with Spartan-3A/AN FPGAs running at 333MHz the following maximum electrical delays need to be met:

1) ± 25 ps maximum electrical delay between any DQ and its associated DQS/DQS#
2) ± 50 ps maximum electrical delay between any address and control signals and the corresponding CK/CK#
3) ± 100 ps maximum electrical delay between any DQS/DQS# and CK/CK#.

Simultaneous Switching Output Considerations

Ground bounce must be controlled to ensure proper operation of high-performance FPGA devices. When multiple output drivers change state at the same time, power supply disturbance occurs. These disturbances can cause undesired transient behavior in output drivers, input receivers, or in internal logic. These disturbances are

often referred to as Simultaneous Switching Output (SSO) noise. The SSO limits govern the number and type of I/O output drivers that can be switched simultaneously while maintaining a safe level of SSO noise. SSO of an individual FPGA IO bank is calculated by summing the SSO contributions of the individual I/O standards in the bank. The SSO contribution is the percentage of full utilization of any one I/O standard in any one bank. The Weighted Average SSO (WASSO) calculation is the done by combining the SSO contributions of all I/O in a bank into a single figure.

When using MIG the bank assignment step helps restrict memory interface SSOs to within the guidelines required for controlling ground bounce. After the designer includes all signals in the design the WASSO calculation should be done to insure the final WASSO calculation is within the device requirements. When your signal assignment is complete follow the steps in the balance of this section to check your design meets the device WASSO constraints.

A Microsoft Excel-based spreadsheet entitled "WASSO Calculator" has been created by Xilinx to automate these calculations. The WASSO calculator uses PCB geometry, such as board thickness, via diameter, and breakout trace width and length, to determine board inductance. It determines the smallest undershoot and logic-Low threshold voltage among all input devices, calculates the average output capacitance, and determines the SSO allowance by taking into account all of the board-level design parameters mentioned in this document. In addition, the WASSO calculator performs checks to ensure the overall design does not exceed the SSO allowance.

The WASSO Calculator with support for Spartan-3A/AN (Rev 1.0) is available as part of xapp689 (rev 1.2). Simply download the calculator, refer to xapp689 (rev 1.2) for a description of each entry and enter the specifics for your PCB Design Parameters, the devices being driven by the FPGA, the parameters for the FPGA output loading and the FPGA bank allocation of resources. The calculator will determine what percentage of the WASSO limit is being utilized for each bank. If your design meets the WASSO requirements of the target device the green OK cells will be highlighted in the spreadsheet.

Example Board Layout for a Sample Memory Subsystem

In this example design we will use a Spartan-3A and a Micron MT47H32M16 DDR2 memory device (this design is used in the Xilinx Spartan-3A DDR2 SDRAM Interface Development Kit User Board, available for purchase from Nu Horizons as ordering part number HW-SPAR3ADDR2-DK-UNI-G. More detailed documentation is available in the kits user guide and design example files). Figure 1 shows the board level schematic of the signal interconnects between the target devices. A detailed description of these signals is given in Table 3, earlier in this document, and will not be repeated here. The main address and data signals between the FPGA and are shown at the top of the diagram. The interface signals below the address and data signals are the various control and timing signals. The power supply and termination voltages are provided to the memory subsystem from the top left and the termination network is shown between the two main devices.

Example Signal Termination Scheme

The termination scheme in use on the Spartan-3A DDR2 SDRAM Interface Development Kit is a good compromise that yields respectable performance while reducing the component count and board complexity. It is suitable for a point-to-point connection between devices where the signal length is low and the signal loading is light:

- SSTL18_I is selected for the Spartan-3A FPGA SelectIO mode. The memory device uses full strength drivers with on-die termination disabled.
- For single ended, unidirectional and bidirectional signals: One 50Ω termination to VTT, in the middle of the trace, which is roughly 1 inch from both devices.
- For differential, unidirectional and bidirectional signals: One 50Ω termination to VTT on each signal in the pair, in the middle of the trace, which is roughly 1 inch from both devices. Differential signals are effectively treated as two single-ended signals.

The Revision A prototype of the Spartan-3A DDR2 SDRAM Interface Development Kit had additional terminator component footprints to enable the designer to experiment with the termination scheme. Several termination schemes were initially validated through simulation using IBIS models. Subsequent experimentation with the prototype confirmed that the less expensive termination scheme described above produced satisfactory results.

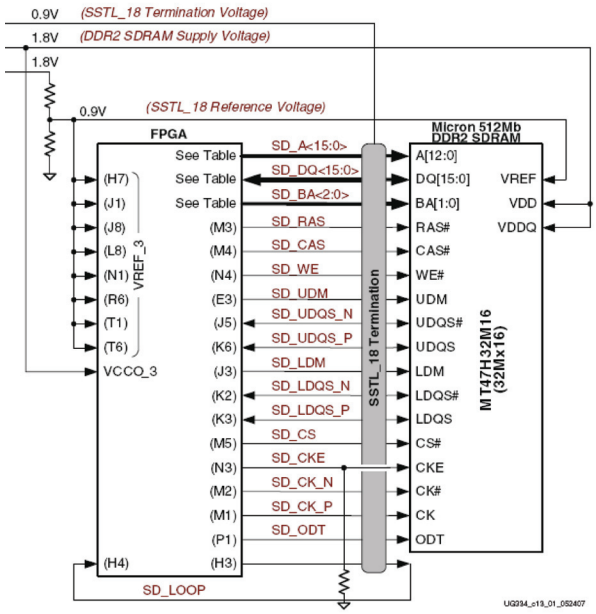

Figure 1: Memory Subsystem Schematic

Example Component Placement

Placement of memory subsystem devices is shown in Figure 2. The pinout for the memory interface in the Spartan-3A FPGA (IC1, The large device in the middle of Figure 2) is located in I/O Bank 3, which is on the left side of IC1. The termination resistors are located in a vertical strip between IC1 and the memory device (IC23, the smaller device on the left side of the figure) as described in the previous section. The relative placement of the key components makes it easy to route the printed circuit board, given the pinout generated by MIG. Notice that it helps to understand the orientation of the devices when assigning I/O banks to the FPGA in MIG. If the orientation is changed later, MIG can easily be re-run with new I/O bank assignments.

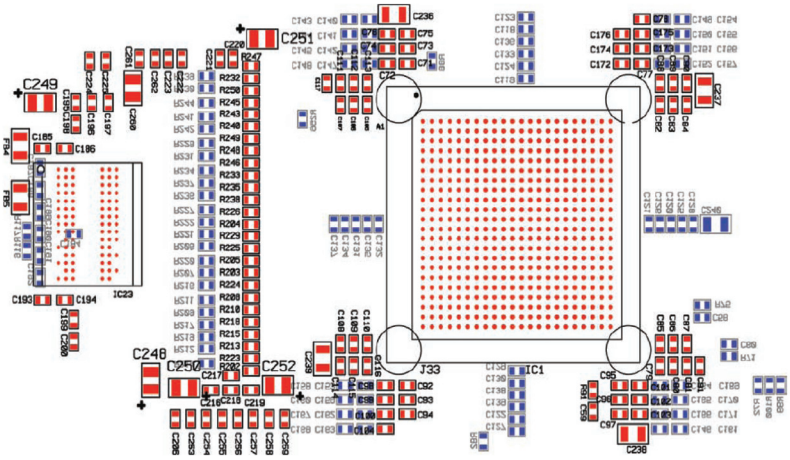

Figure 2: Memory Subsystem Example Component Placement

Example Signal Routing

Some space is needed to accommodate serpentine traces for delay matching of the routes. However, what is shown in Figure 2 is far from an aggressive placement and could be further optimized. The Revision A prototype of the Spartan-3A DDR2 SDRAM Interface Development Kit had additional terminator component footprints that were removed after evaluating the signal integrity on the prototypes, freeing up considerable space. However, the placement of the memory device and the Spartan-3A FPGA was not changed. In theory, this placement could be compressed with further space savings achieved using resistor packs, smaller resistor packages, or staggering the placement of the termination devices to reduce average trace length.

Figure 3 and Figure 4 show the signal routing. The data and strobe signals form a source synchronous bus (governed by the read and write timing budgets) and are routed with an average length of 2.5 inches and less than 50 ps of skew. The address and control signals create another bus and are routed similarly.

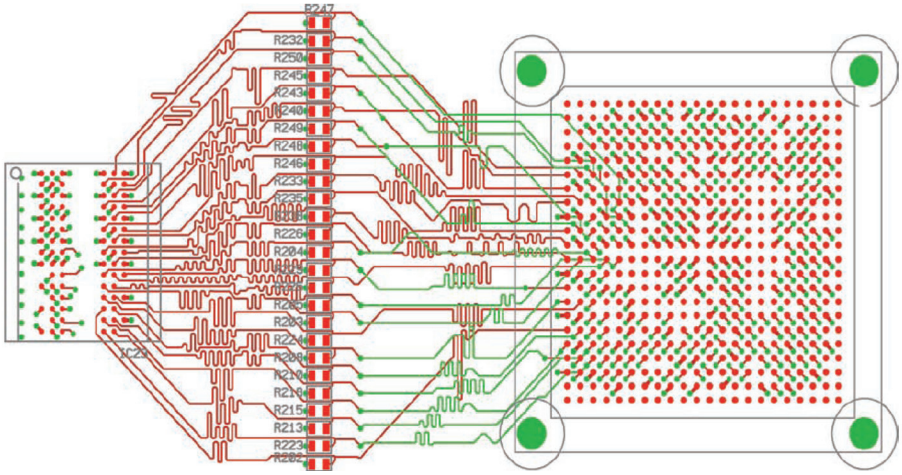

Figure 3: Top and Mid Layer Routing

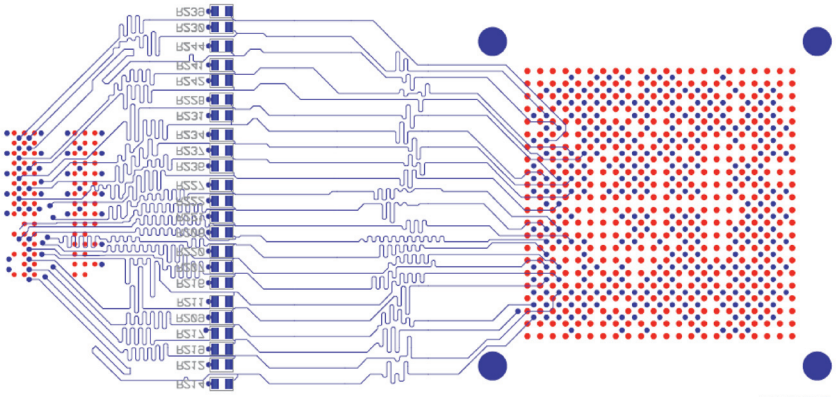

Figure 4: Bottom Layer Routing

The MIG-based memory interface in the Spartan-3A FPGA also requires a properly tuned loopback signal, shown at the bottom of Figure 1. This signal is used by the memory interface to generate an enable for the input capture scheme. The trace delay of the loop should be the sum of the trace delays of the clock forwarded to the memory and the average DQS trace delay. For more information on this topic, refer to UG086, Xilinx Memory Interface Generator User Guide.

Verification of the Example Design

The example memory system design given here has been verified on the Xilinx Spartan-3A DDR2 SDRAM Interface Development Kit. Complete documentation including schematics, layout and example designs as well as a complete user guide are available with the purchase of a kit from Nu Horizons.

Conclusion

This application note has provided a step-by-step guide to designing a memory interface using Xilinx Spartan-3A/AN devices and the Micron Technologies MT47H family of DDR2 memories. The reader should be able to more easily implement these types of interfaces and better understand the major design considerations and concerns present in these types of designs. It is hoped that the reader can also extending the concepts presented in this app note to other Xilinx FPGAs and other Micron Technology memories and simplify those designs as well.

For the reader who wants additional levels of detail on the Xilinx and Micron Technology products used in this app note a list of valuable documents is given in the Reference section at the end of this document. For a more hands on approach to learning more the reader is referred to the Nu Horizons Spartan-3A/3AN Starter Kit Board. This kit is a complete design development platform and features a Xilinx xc3S700A FPGA and a Micron Technologies MT47H32M16 DDR2 memory, the same parts used in this example design. You can use this platform to quickly implement and test your application and leverage a wide range of example designs, application notes and IP included with the kit. Point your browser to http://www.nuhorizons.com/appnotes/AN086/index.asp to learn more about this powerful development platform, purchase parts, or see other useful web links relating to Xilinx and Micron.

References

1) Micron Technology MT47H32M16 Data Sheet (Rev K 8/06)
2) Xilinx Spartan-3A/AN Starter Board Rev D User Guide (ug334, v1.0- May 28, 2007)
3) Xilinx Memory Interface Generator (MIG) (UG086, V2.0)
4) Xilinx OPB DDR2 Memory Controller IP Core Data Sheet (DS532, V1.2)
5) Xilinx Spartan3-A Data Sheet (DS529, 7/10/2007)
6) Xilinx WASSO Calculator and application note (xapp689, Rev 1.2)
7) Xilinx DDR2 Memory Interface for Spartan-3 FPGAs (xapp454, V1.1.1)

Appendix: A Step-By-Step Guide to Using MIG

This section of the document includes a step-by-step guide for creating a memory interface using the Xilinx Memory Interface Generator (MIG). The reader can follow the guide by running the Xilinx Core Generator tool to create a project and then running the MIG tool to create the RTL and testbench for the specified FPGA and memory device. Even if the designers actual target FPGA and/or memory device are different than the current example, it will simplify the design process by first running through the example design to better understand the key steps and then making the necessary modifications on a second pass through the MIG tool. Following the Step-by-Step guide is a quick review of the various outputs of the MIG tool and a listing of the Design Notes reported by MIG for the example design. After completing this section of the document the reader should have a detailed enough understanding of MIG and the accompanying design flow to easily create their own MIG-based design.

Step-By-Step Guide to Creating a Memory Interface for the MT47H32M16

Prior to running the MIG tool you must first create a project in the Xilinx Core Generator (MIG is a tool that runs within Core Generator). Follow these steps to create the required Core Generator project.

1) Run Core Generator

2) Create a new project by clicking on the Create new project option. Give the project a name (TestController for our example) and you will be presented with a window from which you can make the project device selections. Make the following selections:
   a. Select Spartan-3A and Spartan-3AN for the Family
   b. Select xc3s700a for the Device
   c. Select fg484 for the Package
   d. Select -4 for the Speed Grade
   e. Click OK

3) The project file is now initialized. To run the MIG tool expand the Memories & Storage Elements folder by clicking on the '+' sign.

4) Next expend the Memory Interface Generator folder by clicking on the '+' sign.

5) Click on the MIG 2.0 Icon and the MIG Reference Design will come up in the right hand window.

6) To run the MIG click on the Customize entry in the right hand window.

Once the MIG window is open it immediately displays the Core Generator project options previously defined. These must be correct or you will need to exit out of the program and begin again.

1) *Selected CORE Generator Project Options*- If the settings are correct click Next to proceed to the next step.

2) *MIG Output Options*- MIG has three main output options- Create Design, Create Design for Xilinx Reference Boards or Create Preset Configuration. We will select the Create Design option for this example. (Refer to the MIG user guide referenced at the end of this application note for more information on these other options).

3) *Component Name*- Select a component name for the memory interface. Type in 'MicronController' and click Next.

4) *Compatible FPGAs*- In this step the user selects compatible FPGAs (FPGAs with the same package) that can be used with the design. Selecting a compatible device allows the design to be migrated by limiting pins to common locations. (For this example we will skip this step, but in practice this can be a useful feature to allow the design to migrate to a different device.)
   - Click Next

5) *Memory Selection*- In this step the user selects the type of memory device used in the design.
   - Select the DDR2 memory type and click Next

6) *Controller Options*- In this step the user selects various controller options for use in the design.
   - Frequency: We will use the default frequency of 166MHz in our design.
   - Memory Type: Since we have already selected a component this selection is not available and is grayed out.
   - Memory Part: Select the MT47H32M16XX-3.
   - Data Width: Select 16 as the Data Width for the memory device.
   - Write Pipe Stages: Select the default of 4 as the number of pipeline stages.
   - Click Next

7) *Set Mode Registers*- In this step the values loaded into the MT47H32M16 mode register during initialization are defined by the designer. Some settings have been automatically selected due to the selections of the speed grade and frequency specified previously. For example, the CAS Latency of 3 is based on the frequency of operation specified on the Controller Options page.
   - Burst Length: Select the default of 4
   - Burst Type: Select the default of sequential
   - The rest of the settings are already determined
   - Click Next

8) *Set Extended Mode Registers*- In this step the values loaded into the MT47H32M16 extended mode register are defined by the designer. Only some settings are supported by the controller.
   - Output Drive Strength: Select Fullstrength
   - RTT (nominal)- ODT: Select 75ohms(01)
   - DQS# Enable: Select Enable(0)
   - The rest of the settings are already specified

9) *DCM Option*- In this step the option of including the use of a DCM or excluding it is selected. For the example design we will include the use of the DCM.
   - Use DCM- Click the check box.

10) *SSTL Class Option*- In this step Class I or Class II operation is specified. In general Class II is preferred.
   - Class for Address and Control- Use the default setting of Class II.

11) *Reserve Pins*- In this step any pins required by the FPGA design are selected so that the memory interface generator will not select those for the memory interface.
   - Selected Pins- Use this table to select the pins that are required by the non-interface portion of the design by selecting on the left hand list and clicking Reserve. You may read in already specified pins by clicking on the Read UCF button. Once all pins are selected by may click the Save as button record your settings. Selecting no reserved pins allows the memory interface to take priority. For the example design we will not select any reserved pins. Click next.

12) *Bank Selection*- In this step the memory interface signals are assigned to FPGA I/O banks. We will use the default settings to provide the maximum flexibility to the place and route tool.
   - Click Next

13) *Micron Model License*- The MIG creates a Micron memory model for the MT47H32M16 for use in simulation. You must read and agree to the Micron License Agreement during this step to create the model.
   - Select Accept License Agreement by clicking on the check box.

14) *Summary*- This step displays a summary of the information entered into the MIG so far.
   - Click the Generate button to create the design files.

15) *Creating PCBs for MIG Designs*- This step references the appendix of the MIG User Guide which contains detailed information about PCB design guidelines. Review the information on these guidelines. (The PCB guidelines for the MT47H32M16 are also included in this document so you can skip a detailed review within the MIG program if you prefer).
   - Click Next

16) *Finish*- This step indicates if generation was successful and displays the design notes associated with the generated design.
   - Review the Design Notes (These notes are included at the end of this application note for your easy reference. Be sure to read and understand these notes prior to completing the FPGA Design phase of your project.)
   - Click the Close Button when finished.

Results of Running MIG

After running MIG as described in the previous step-by-step guide you have a complete memory interface controller IP Core that can be used with the Micron MT47H32M16. The IP Core is contained in several output and support files. Under the Project Directory (in our example, TestControlelr) the component name directory (in our example , MicronController) a .tcl file and three folders (example_design, user_design and doc) are created. The .tcl file is used for the CORE Generator application. The example_design directory contains a design with a testbench. The user_design directory does not contain a testbench. The user_design will be instantiated in the final application. The example_design can be used to insure the simulation tool flow is working correctly and to test the features of the memory interface.

The example_design and user_design folders contain several useful folders and files. They are:

- rtl: Contains all the .rtl files (either VHDL or Verilog design files).
- par: Contains the UCF with constraints for the design. Two scripts files are generated:
  ise_flow.bat: Double-click the ise_flow.bat script file to run the design through synthesis, build, map, and par. This script file sets all the required options.
  create_ise.bat: Double-click the create_ise.bat file to create an ISE project. To run the project in GUI mode, the user double-clicks the ISE project file to open up ISE in GUI mode with all project settings.
- synth: Contains the SDC file which has design constraints for Synplify Pro synthesis tool. This folder also has the script files, which set various tool options.

- sim: Contains the testbench files that are needed to simulate the design. It also has an executable and a .do file. If sim.exe is double-clicked, the design is automatically simulated using the ModelSim simulator. There is a simulation_help.chm file in the sim folder that helps you to understand the simulation environment provided. For the user_design folder, a synthesizable testbench module is also present in the sim folder.

The docs directory contains all relevant documents, including application notes, timing analysis spreadsheets, and user guides.

MIG Generated Design Notes

An on screen report is generated by MIG that displays the Design Notes specific to a particular design. Make sure you review these Design Notes prior to completing the FPGA Design phase of your design.

Listed below are the Design Notes generated by MIG for the target devices in this example design.

1. This design is tested with ISE 9.2.03i and Synplify Pro 8.8.0.4.

2. The reset polarity is set to active low by default. You can change this by editing the parameter file.

3. It is recommended not to change any other parameters directly. All other changes should be done through MIG GUI.

4. We recommend using adjacent banks to get better timing. Also "Area Grouping" may improve timing results.

5. The user's application must have a DCM primitive instantiated in the design for designs generated without DCM. All user clocks should be driven through BUFGMUXs.

6. Users may want to change default IO standards of SSTL18_I signals to SSTL18_II for DDR2 and SSTL2_I to SSTL2_II for DDR in UCF file.

7. "Verify UCF" is not supported for Spartan-3/3A/3E/3A DSP.

8. The Load Mode Register command is not supported on the User Interface.

9. Use x8 Memory Components for better timing.

10. The controller sets bit A10 (Auto pre-charge) to zero for read and write commands. The controller ignores the A10 bit from the user interface. This results in the column address from the user interface to be non-contiguous.

11. Synplicity Attributes may show up warning messages in XST runs and vice versa.
12. Added "FROM TO" constraints to the UCF to get rid of the majority of unconstraint paths. The other remaining paths are either OFFSET/IN where the source is either clock pad or reset pad (so they are not real timing paths), or they are OFFSET/OUT for the source synchronous interface to the memory (so again, timing is irrelevant). You can ignore those unconstrained paths due to the above reasons.

Notice Of Disclaimer