



WP260 (v1.0) February 16, 2007

Memory Interfaces Made Easy with Xilinx FPGAs and the Memory Interface Generator

By: Adrian Cosoroaba

As FPGA designers strive to achieve higher performance while meeting critical timing margins, the memory interface design is a consistently difficult and time-consuming challenge. Xilinx FPGAs provide I/O blocks and logic resources that make the interface design easier and more reliable. Nonetheless, the I/O blocks, along with extra logic, must be configured, verified, implemented and properly connected to the rest of the FPGA by the designer in the source RTL code, carefully simulated, and then verified in hardware to ensure a reliable memory interface system.

This white paper discusses the various memory interface controller design challenges and Xilinx solutions. It also describes how to use the Xilinx software tools and hardware-verified reference designs to build a complete memory interface solution for your own application, from low-cost DDR SDRAM applications to higher-performance interfaces like the 667 Mb/s DDR2 SDRAMs.

© 2007 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Memory Interface Trends and Xilinx Solutions

In the late 1990s, memory interfaces evolved from single-data-rate (SDR) SDRAMs to double-data-rate (DDR) SDRAMs, with today's DDR2 SDRAMs running at 667 Mb/s per pin or higher. Present trends indicate that these rates are likely to double every four years, potentially reaching over 1.2 Gb/s per pin by the year 2010 with the upcoming DDR3 SDRAMs. See [Figure 1](#).

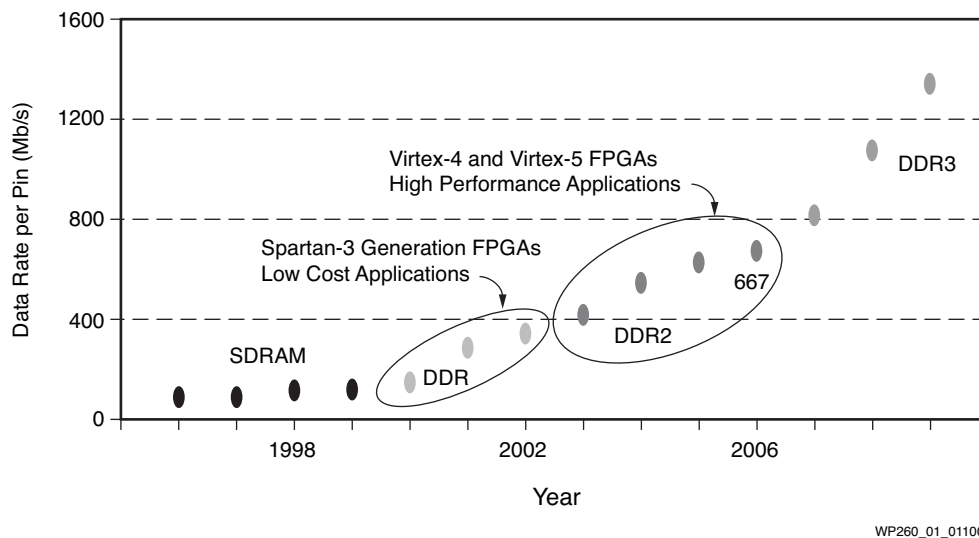


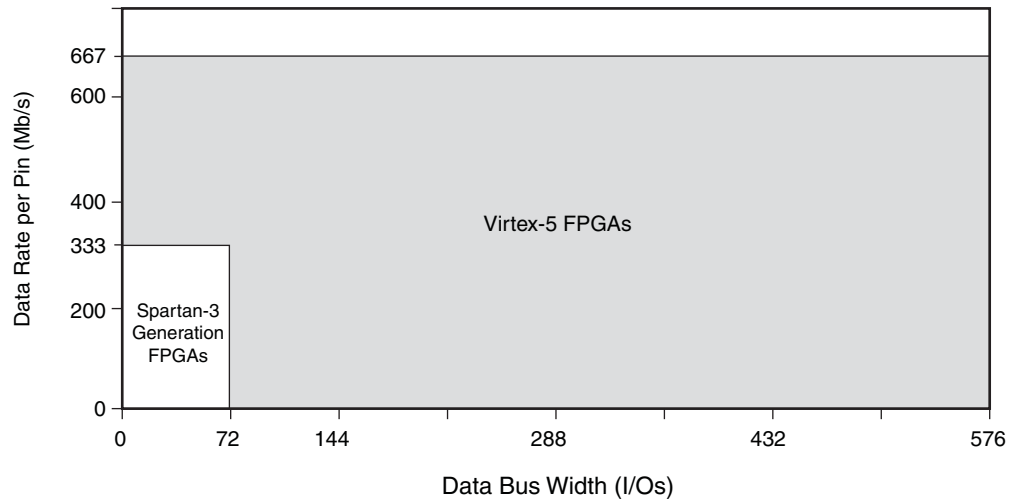
Figure 1: DRAM Data Rates Trends and Xilinx FPGA Solutions

Applications can generally be classified in two categories: low-cost applications, where the cost of the device is most important; and high-performance applications, where getting the highest bandwidth is paramount.

DDR SDRAMs and low-end DDR2 SDRAMs running below 400 Mb/s per pin are adequate to meet most low-cost systems memory bandwidth requirements. For these applications, Xilinx offers the Spartan™-3 Generation FPGAs: Spartan-3, Spartan-3E, and Spartan-3A devices.

For high-performance applications, pushing the limits of the memory interface bandwidth like 533 and 667 Mb/s per pin DDR2 SDRAMs, Xilinx offers the Virtex™-4 and Virtex-5 FPGAs, which are capable of meeting the highest bandwidth requirements of most systems today.

Bandwidth is a factor related to both the data rates per pin and the width of the data bus. Spartan-3 Generation, Virtex-4, and Virtex-5 FPGAs offer distinct options that span from smaller low-cost systems, with data bus widths of less than 72 bits, to 576 bits wide for the larger Virtex-5 packages (see [Figure 2](#)).



WP260_02_020507

Figure 2: Xilinx FPGAs and Memory Interfaces Bandwidth

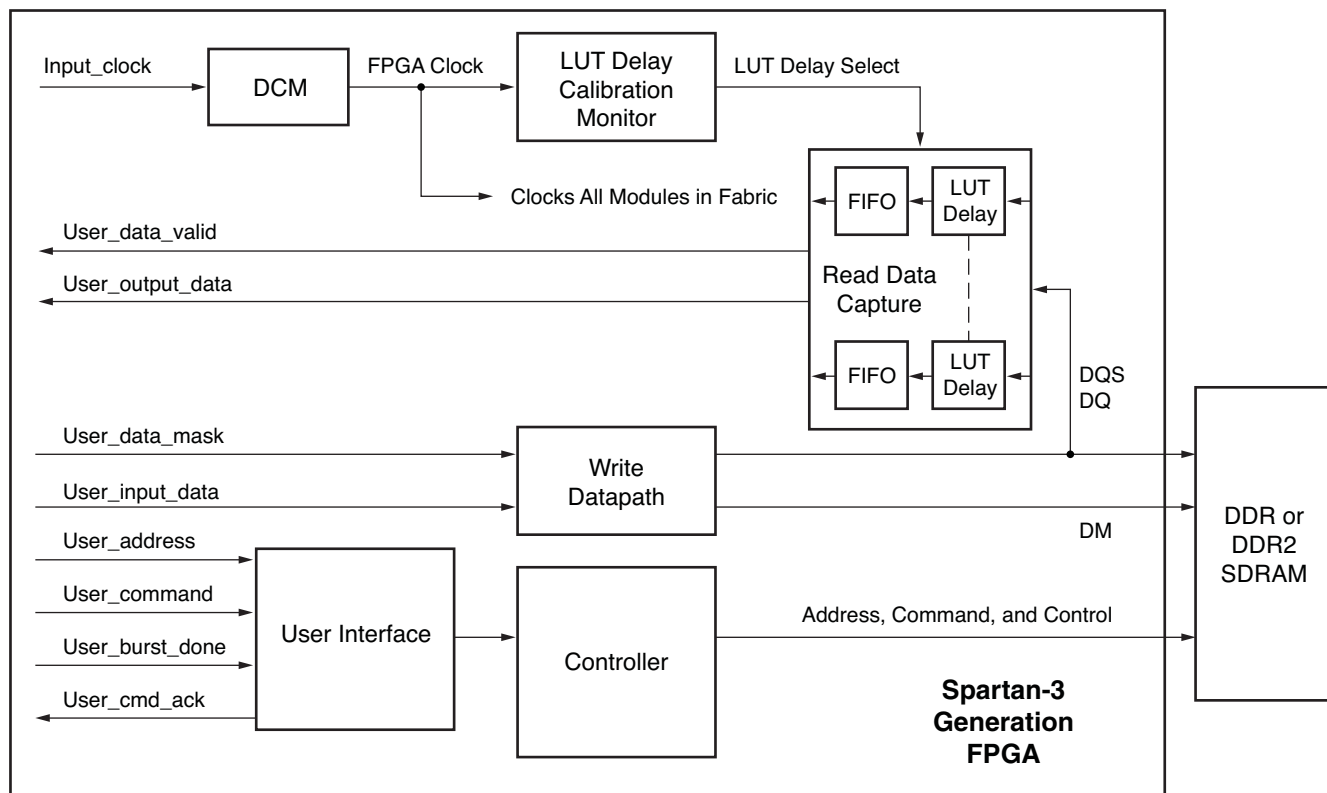
Wider buses at more than 400 Mb/s make the chip-to-chip interfaces all the more challenging. Larger packages and better power- and ground-to-signal ratios are required. Virtex-4 and Virtex-5 FPGAs have been built with advanced Sparse Chevron packaging, which provides superior signal-to-power and ground-pin ratios. Every I/O pin is surrounded by sufficient power and ground pins and planes to ensure proper shielding for minimum crosstalk noise caused by simultaneously switching outputs (SSO).

Low-Cost Memory Interfaces

Not all systems built today push the performance limits for memory interfaces. When low cost is a primary decision driver and memory bit rates per pin of up to 333 Mb/s are sufficient, the Spartan-3 Generation of FPGAs coupled with Xilinx software tools provide an easy-to-implement, low-cost solution.

A memory interface and controller for an FPGA-based design have three fundamental building blocks: the Read and Write Data interface, the memory controller state machine, and the user interface that bridges the memory interface design to the rest of the FPGA design (Figure 3). These blocks, implemented in the fabric, are clocked by the output of the Digital Clock Manager (DCM). In the Spartan-3 Generation implementation, the DCM also drives the Look-Up Table (LUT) delay calibration monitor (a block of logic that ensures proper timing for Read Data capture). The delay calibration circuit is used to select the number of LUT-based delay elements used to delay the strobe lines (DQS) with respect to Read Data. The delay calibration circuit calculates the delay of a circuit that is identical in all respects to the strobe delay circuit.

All aspects of the delay are considered for calibration, including all the component and route delays.



WP260_03_011007

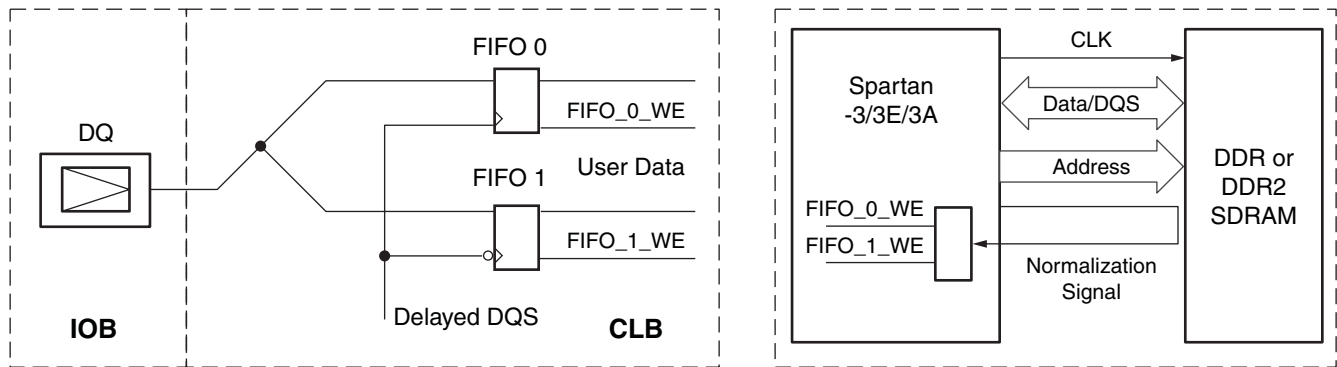
Figure 3: DDR and DDR2 SDRAM Interface Implementation for Spartan-3 Generation FPGAs

The user interface is a handshaking-type interface. The user sends a command, Read or Write, along with the address and data for Write; the user-interface logic responds with the User_cmd-ack signal when the next command can follow.

In the Spartan-3 Generation implementation, the Read Data capture is implemented using the LUTs in the Configurable Logic Blocks (CLBs). During a Read transaction, the DDR or DDR2 SDRAM device sends the Read Data strobe (DQS) and associated data to the FPGA edge aligned with the Read Data (DQ). Capturing the Read Data is a challenging task in source-synchronous interfaces that run at high frequencies because data changes at every edge of the non-free-running DQS strobe. The Read Data capture implementation uses a tap-delay mechanism based on LUTs. The DQS clocking signal is delayed with the proper amount to place it with enough margin in the Read Data valid window for capture inside the FPGA.

The capture of Read Data is done in the LUT-based, dual-port distributed RAM (Table 4). The LUT RAM is configured as a pair of FIFOs, and each data bit is input into

the rising edge (FIFO 0) and falling edge (FIFO 1) FIFOs as shown in [Figure 4](#). These 16-entry-deep FIFOs are asynchronous with independent Read and Write ports.



WP260_04_020707

Figure 4: Read Data Capture FIFO Implementation for Spartan-3 Generation FPGAs

The Read Data from the memory is written into FIFO_0 on the rising edge of the delayed DQS, and into FIFO_1 on the falling edge of the delayed DQS. Transferring the Read Data from the DQS clock domain to the memory controller clock domain is done through these asynchronous FIFOs. Data can be read out of both FIFO_0 and FIFO_1 simultaneously in the memory controller clock domain. FIFO Read pointers are generated in the FPGA internal clock domain. The generation of Write Enable signals (FIFO_0 WE and FIFO1_WE) is done using the DQS and an external loopback or normalization signal. The external normalization signal is driven to an Input/Output Block (IOB) as an output, and it is then taken as an input through the input buffer. This technique compensates for the IOB, device and trace delays between the FPGA, and the memory device. The normalization signal from the input pad of the FPGA uses similar routing resources as the DQS before it enters the LUT delay circuit to match the routing delays. The trace delay of the loop should be the sum of the trace delays of the clock forwarded to the memory and the DQS ([Figure 4](#)).

Write Data commands and timings are generated and controlled through the Write Data interface. The Write Data interface uses IOB flip-flops and the 90, 180, and 270° DCM outputs to transmit the DQS strobe properly aligned to the command and data bits per DDR and DDR2 SDRAM timing requirements.

The implementation of a DDR and DDR2 SDRAM memory interface for Spartan-3 Generation FPGAs has been fully verified in hardware. A reference design example of a low-cost DDR2 SDRAM implementation was developed using the [Spartan-3A starter kit board](#). The design was developed for the onboard, 16-bit-wide, DDR2 SDRAM memory device and uses the XC3S700A-FG484. The reference design utilizes only a small portion of the Spartan-3A FPGA's available resources: 13% of IOBs, 9% of logic slices, 16% of BUFGMUXs, and one of the eight DCMs. This implementation leaves resources available for other functions that are needed in the rest of the FPGA design.

Designers can easily customize Spartan-3 Generation memory interface designs to fit their application using the Memory Interface Generator (MIG) software tool, described later in this white paper.

High-Performance Memory Interfaces

With higher data rates, interface timing requirements become more challenging. Memory interface clocking requirements are typically more difficult to meet when reading from memory, as compared with writing to memory. The trend toward higher data rates presents a serious problem to designers because the Data Valid window (that period within the data period during which Read Data can be reliably obtained) is shrinking faster than the data period itself. The reason for the trend is the various uncertainties associated with system and device performance parameters, which impinge upon the size of the Data Valid window, do not scale down at the same rate as the data period.

This trend is readily apparent when comparing the Data Valid windows of DDR SDRAMs running at 400 Mb/s and DDR2 memory technology, which runs at 667 Mb/s. The DDR device with a 2.5 ns data period has a Data Valid window of 0.7 ns, while the DDR2 device with a 1.5 ns period has a mere 0.14 ns (Figure 5).

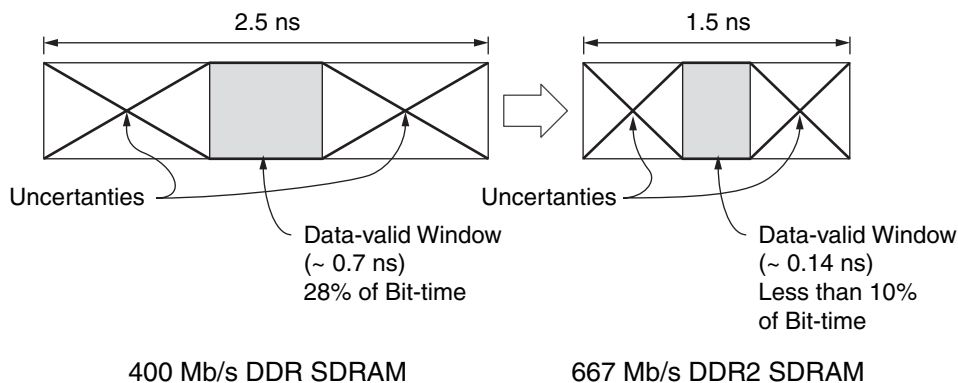
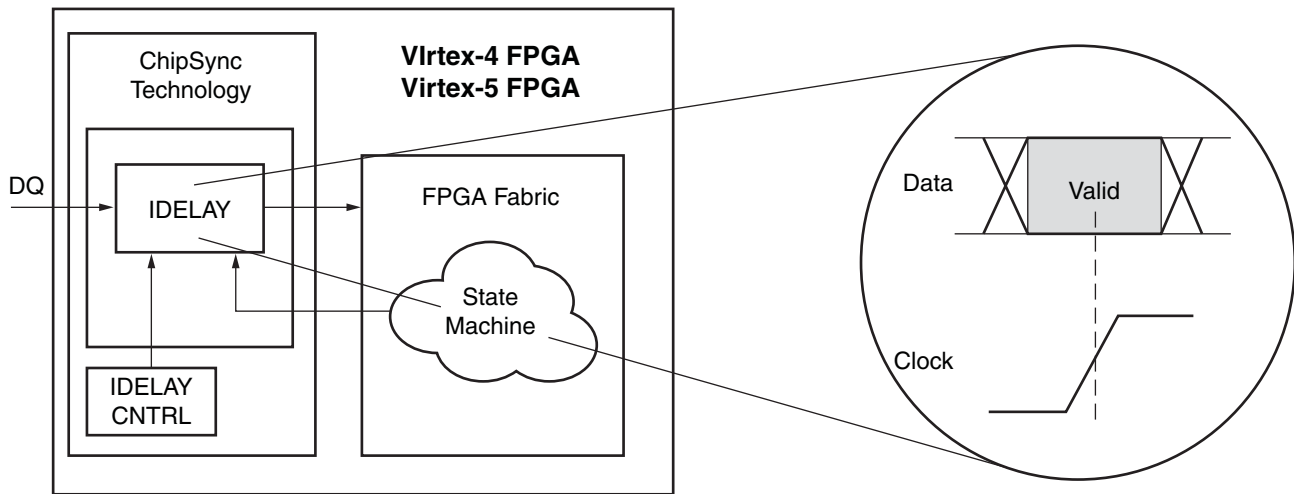


Figure 5: The Shrinking Data Valid Window

Clearly, this accelerated erosion of the Data Valid window introduces a new set of design challenges for the FPGA designer that require a more effective means of establishing and maintaining reliable memory interface performance.

Read Data can be captured into Configurable Logic Blocks (CLBs) using the Read Data strobe (DQS) as implemented in the Spartan-3 Generation FPGAs, but the delay taps needed to center the strobe or clock to the Data Valid window using LUTs are coarse. The delay taps implemented in the CLBs have a resolution on the order of hundreds of picoseconds while the Read capture timings for data rates over 400 Mb/s require resolution of an order of magnitude better than the CLB-based taps. Virtex-4 and Virtex-5 FPGAs answer this challenge with dedicated delay and clocking resources in the I/O blocks - called ChipSync™ technology. The ChipSync block built into every

I/O contains a string of delay elements, tap delays, called IDELAY in Virtex-4 and IODELAY in Virtex-5 FPGAs, with a resolution of 75 ps (see [Figure 6](#)).

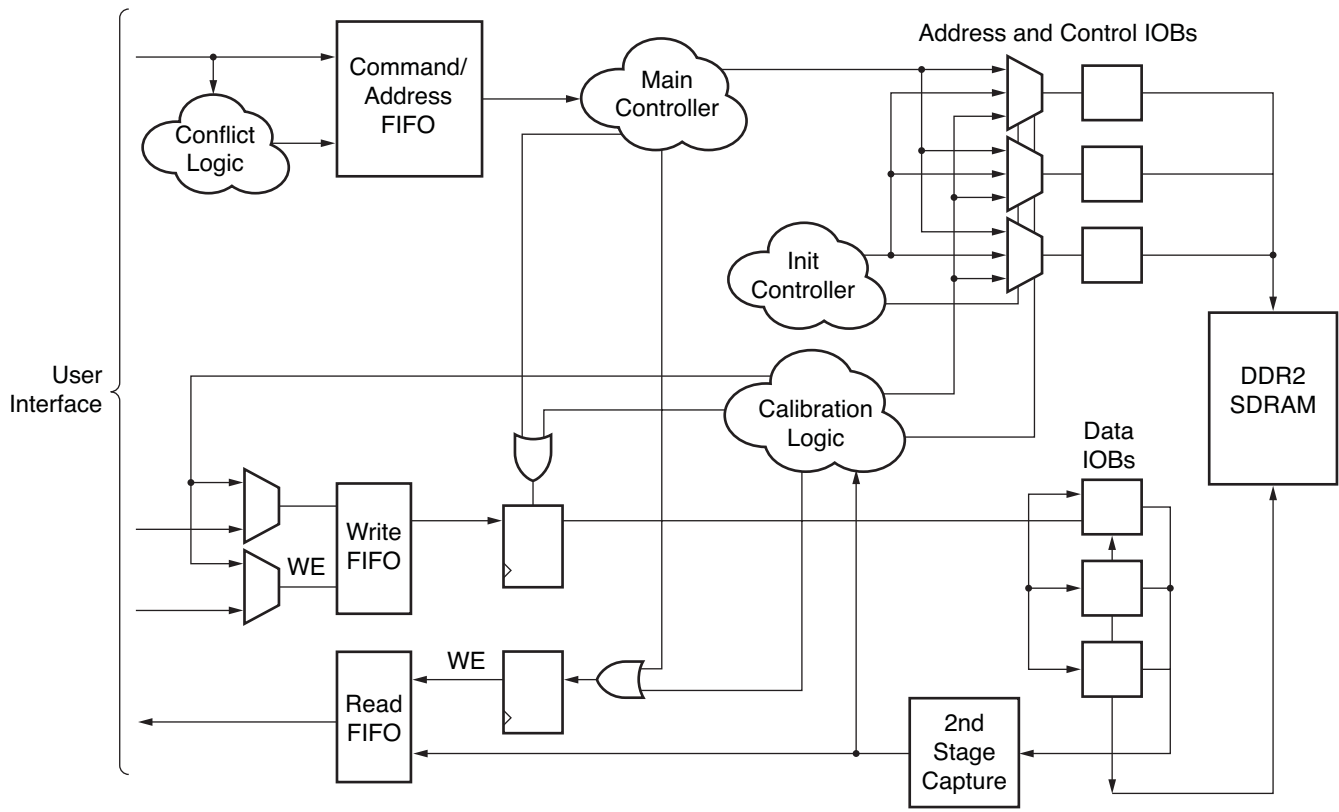


WP260_06_020507

Figure 6: Clock-to-Data Centering using 75 ps Tap Delays in Virtex-4 and Virtex-5 FPGAs

The architecture for the implementation is based on several building blocks. The user interface that bridges the memory controller and physical layer interface to the rest of the FPGA design uses a FIFO architecture ([Figure 7](#)). There are three sets of FIFOs: Command/Address FIFO, Write FIFO, and Read FIFO. The FIFOs hold the command, address, Write Data, and Read Data. The main controller block controls the Read,

Write, and Refresh operations. Two other logic blocks execute the clock-to-data centering for Read operations, the initialization controller and the calibration logic.



WP260_07_011107

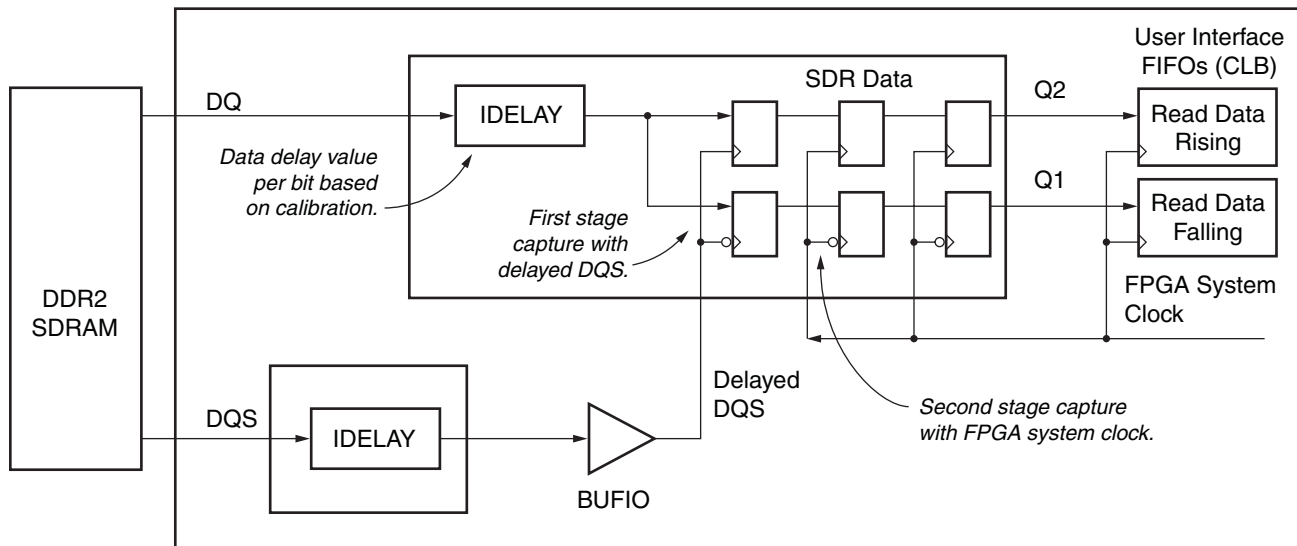
Figure 7: Virtex-5 FPGA Memory Interface Architecture

The physical layer interface for address, control, and data is implemented in the I/O blocks (IOBs). The Read Data is recaptured in a second stage of latches that are also part of the IOBs.

The Virtex-4 and Virtex-5 FPGA memory interface reference designs support two Read Data capture techniques. The *direct-clocking* technique supported by the Virtex-4 FPGAs delays the Read Data so that it can be directly registered using the system clock in the input DDR flip-flop of an IOB. Each Read Data bit is individually calibrated for optimal alignment to the FPGA clock. This technique provides adequate performance for up to 240 MHz clock rates.

The second technique is called the *strobe-based* technique. It is used for even higher clock rates and is supported by both Virtex-4 and Virtex-5 FPGAs. It uses the memory strobe to capture corresponding Read Data and registers it with a delayed version of the strobe distributed through a localized I/O clock buffer (BUFIO). This data is then synchronized to the system clock domain in a second stage of flip-flops. The input serializer/deserializer feature in the IOB is used for Read capture; the first pair of

flip-flops transfer the data from the delayed strobe to the system clock domain (Figure 8).



WP260_08_020507

Figure 8: Virtex-5 Read Data Capture using the Strobe-based Technique

Both techniques involve the use of tap delay (IDELAY) elements that are varied during a calibration routine implemented by the calibration logic. This routine is performed during system initialization to set the optimal phase between strobe, data, and the system clock to maximize timing margins. Calibration removes any uncertainty caused by process-related delays, compensating for components of the path delay that are static to any one board. These components include PCB trace delays, package delays, and process-related components of propagation delays (both in the memory and FPGA) as well as setup/hold times of capture flip-flops in the FPGA I/O blocks. Calibration accounts for variation in delays that are process, voltage, and temperature dependent at the system initialization stage.

During calibration, the delay taps for strobe and data are incremented to perform edge detection by continuously reading back from memory and by sampling either a prewritten training pattern or the memory strobe itself until either the leading edge or both edges of the Data Strobe (DQS) are determined. The number of taps for data or strobe is then set to provide the maximum timing margin. For *strobe-based* capture, the strobe and data can have different tap delay values because there are essentially two stages of synchronization: one to first capture the data in the strobe domain and another to transfer this data to the system clock domain.

The *strobe-based* capture method becomes necessary at higher clock frequencies. Its two-stage approach offers better capture timing margins because the DDR timing uncertainties are mainly restricted to the first rank of flip-flops in the IOBs. Furthermore, because the strobe is used to register the data, timing uncertainty is smaller for the strobe-to-data variation when compared to the clock-to-data (T_{ac}) variation. For example, in the case of DDR2, these uncertainties are given by the tDQSQ and tQHS parameters of the part.

Write timing for Virtex-4 and Virtex-5 FPGAs, as in the Spartan-3 Generation FPGA implementation, is supported by the DCM that generates two phase outputs of the system clock. The memory strobe is forwarded using an output DDR register clocked by an in-phase copy of the system clock. The Write Data is clocked by a DCM clock

output that is 90° ahead of the system clock. This technique ensures that the strobe is center-aligned to the data on a Write at the outputs of the FPGA.

Other aspects of the design include the overall controller state machine logic generation and the user interface. In order to make the complete design easier for the FPGA designer, Xilinx has developed the Memory Interface Generator (MIG) tool.

Controller Design and Integration

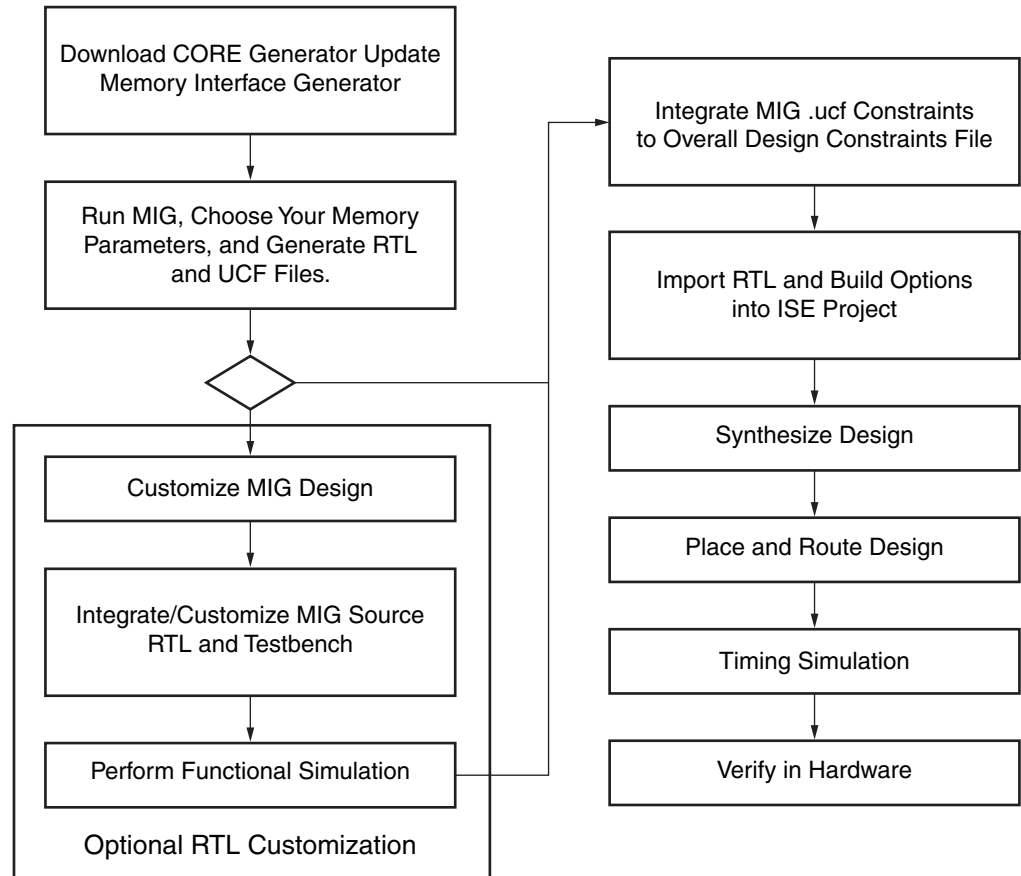
The complexities and intricacies of creating memory controllers pose a wide assortment of challenges, which for the FPGA designer suggest the need for a new level of integration support from the tools that accompany the FPGA.

Integrating all the building blocks including the memory controller state machine is essential for the completeness of a design. Controller state machines vary with the memory architecture and system parameters. State machine code can also be complicated as well as a function of many variables, such as:

- Architecture (DDR, DDR2, QDR II, RLDRAM, etc.)
- Number of banks (either external or internal to the memory device)
- Data bus width
- Memory device width and depth
- Bank and row access algorithms

Finally, parameters like Data-to-Strobe ratios (DQ/DQS) can add further complexity to the design. The controller state machine must issue the commands in the correct order while considering the timing requirements of the memory device.

The complete design can be generated with the MIG tool, a software tool freely available from Xilinx as part of the CORE Generator™ suite of reference designs and IP. The MIG design flow (Figure 9) is similar to the traditional FPGA design flow. The benefit of the MIG tool is that there is no need to generate the RTL code from scratch for the physical layer interface or the memory controller.

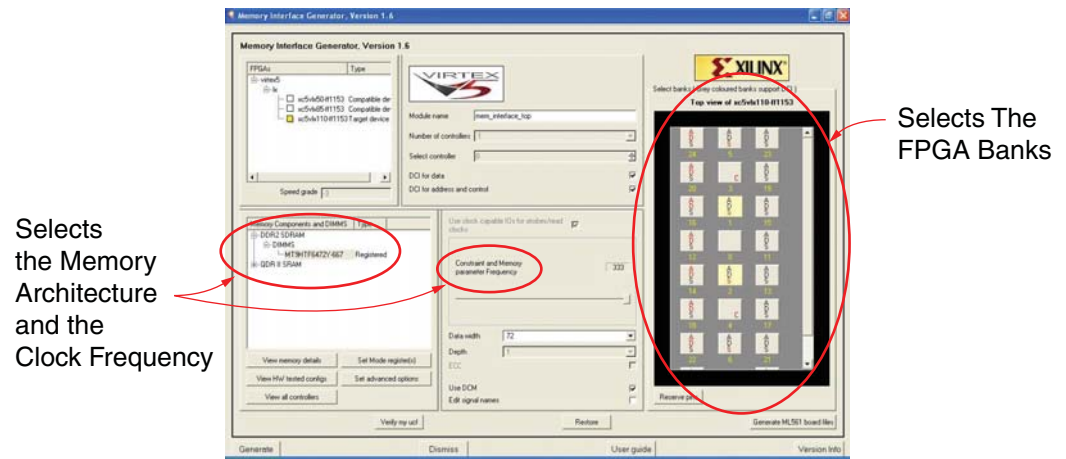


WP260_08_020707

Figure 9: MIG Design Flow

The MIG Graphic User Interface (GUI) is used to set system and memory parameters (Figure 10). For example, after selecting the FPGA device, package, and speed grade, the designer can select the memory architecture and pick the actual memory device or DIMM. The same GUI provides a selection of bus widths and clock frequencies. It also offers for some FPGA devices the option to have more than one controller for multiple

memory bus interfaces. Other options provide control of the clocking method, CAS latency, burst length, and the pin assignments.



WP260_10_020707

Figure 10: MIG Graphic User Interface (GUI)

In less than a minute, the MIG tool can generate the RTL and UCF files, which are the HDL code and constraints files, respectively. These files are generated using a library of hardware-verified reference designs, with modifications based on the user's inputs.

The designer has complete flexibility to further modify the RTL code. Unlike other solutions that offer *black box* implementations, the code is not encrypted, providing complete flexibility to change and further customize a design. The output files are categorized in modules that apply to different building blocks of the design: user interface, physical layer, controller state machine, etc. Consequently, the designer could choose to customize the state machine that controls the bank access algorithm. The Virtex-4 and Virtex-5 DDR2 bank access algorithms generated by the MIG tool are different. The Virtex-5 design employs a least-recently-used (LRU) algorithm that keeps one row in up to four banks always open to reduce the overhead associated with opening and closing rows. If a row in a new bank needs to be opened, the controller closes the row in the least recently used bank and opens a row in the new bank. In the Virtex-4 controller implementation, only a single bank has an opened row at any time. Each application may require its own access algorithm to maximize throughput and the designer can modify the algorithm by changing the RTL code to better fit the access patterns of his application.

After the optional code change, the designer can perform additional simulations to verify the functionality of the overall design. The MIG tool also generates a synthesizable testbench with memory checker capability. The testbench is a design example used in the functional simulation and the hardware verification of the Xilinx base design. The testbench issues a series of writes and readbacks to the memory controller. It can also be used as a template to generate a custom testbench.

The final stage of the design is to import the MIG files into the ISE project, merge them with the rest of the FPGA design files followed by synthesis, place and route, run additional timing simulations if needed, and finally verify the design in hardware. The MIG software also generates a batch file with the appropriate synthesis, map, and place and route options to help optimally generate the final bit file.

High-Performance System Design

Implementing high-performance memory interfaces goes beyond the FPGA on-chip design implementation and requires meeting chip-to-chip challenges like signal integrity requirements and board design challenges.

The signal integrity challenge is to gain control of crosstalk, ground bounce, ringing, noise margins, impedance matching, and decoupling to ensure reliable signal valid windows. The column-based architecture used in the Virtex-4 and Virtex-5 FPGA enables I/O, clock, power, and ground pins to be located anywhere on the silicon chip, rather than just along the periphery. This architecture alleviates the problems associated with I/O and array dependency, power and ground distribution, and hard-IP scaling. Additionally, Sparse Chevron packaging technology used in Virtex-4 and Virtex-5 FPGAs enables distribution of power and ground pins evenly across the package. These packages offer better immunity against crosstalk with the benefit of improved signal integrity for high-performance designs. Figure 11 shows the Virtex-5 FPGA package pinouts. The dots represent power and ground pins, and the crosses represent pins available for the user; thus, the I/O signals are surrounded by sufficient power and ground to ensure good shielding for SSO noise.

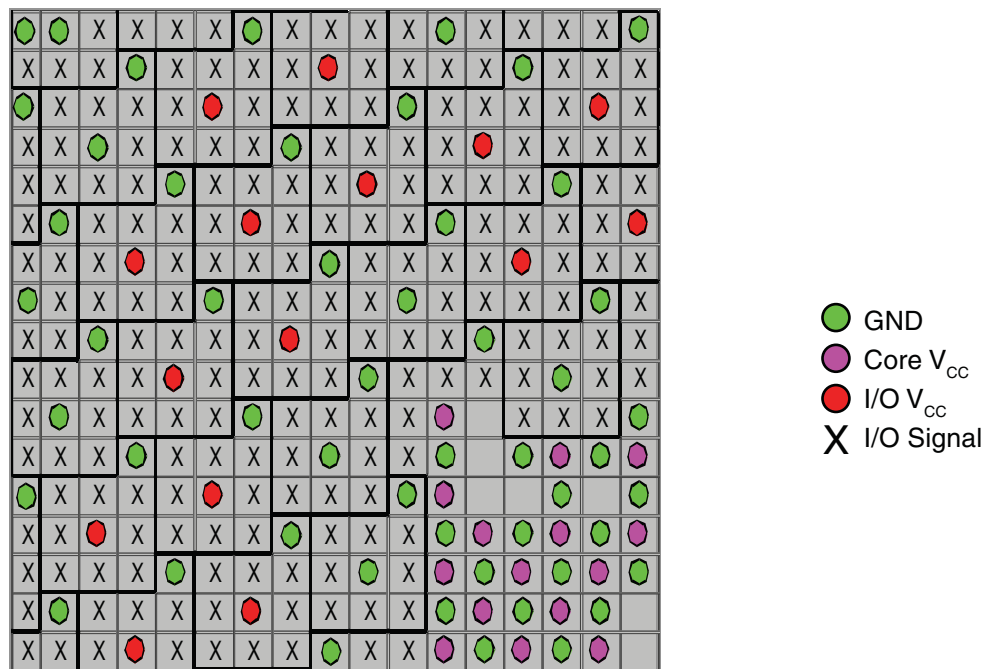


Figure 11: Virtex-5 FPGA Package with Distributed Power and Ground Pins

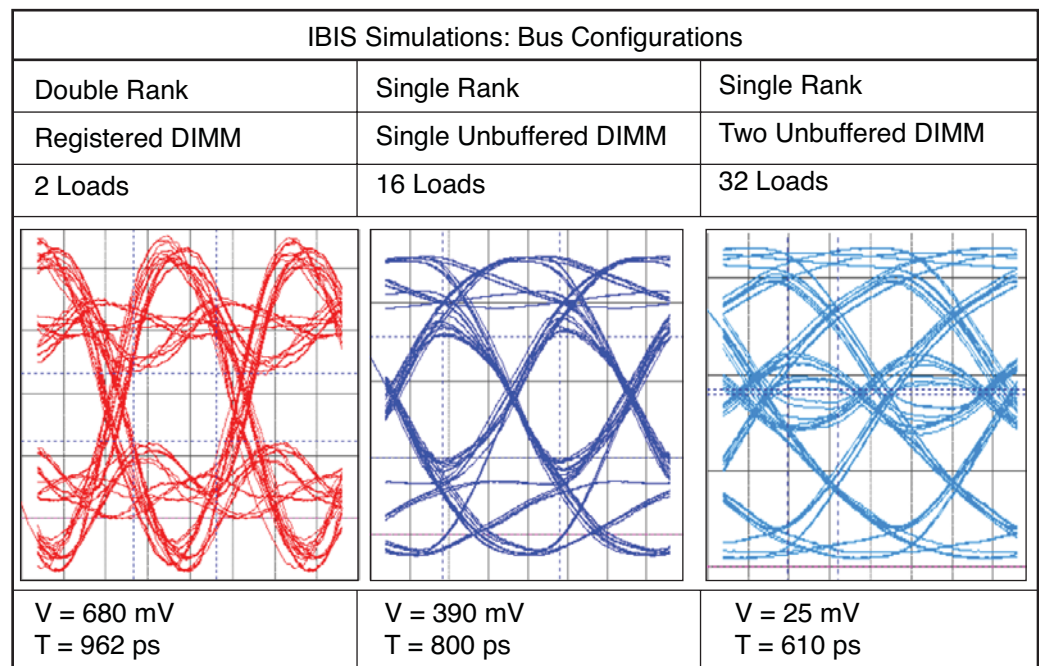
Increasing the data rate is not always sufficient for high-performance memory systems; wider data buses are needed to achieve the desired bandwidth. Interfaces of 144 or 288 bits are not uncommon today. Numerous bits switching simultaneously can create signal integrity problems. The SSO limit is specified by the device vendor and represents the number of signal pins that the user can use simultaneously per bank in the device. With the Sparse Chevron package advantage of good SSO noise shielding and the homogeneous I/O structure, wide data bus interfaces are quite feasible.

Table 1 shows the Virtex-5 LX devices and the maximum data bus width that meets the SSO requirements for 600 Mb/s data rates.

Table 1: Maximum Data-bus Width Meeting SSO requirements at 600 Mb/s rates for Virtex-5 LX Devices

Virtex-5 Device	Package	Maximum Data Width
LX30, LX50	FF324	108
LX30, LX50, LX85, LX110	FF676	243
LX50, LX85, LX110	FF1153	432
LX110, LX220, LX330	FF1760	648

Another system design challenge when designing a large or dense memory system is capacitive loading. High-performance memory systems can require multiple memory devices driven by a common bus for address and command signals. This corresponds, for example, to the case of a dense unbuffered DIMM interface. One interface with two 72-bit unbuffered DIMMs can have a load of up to 36 receivers on the address and command buses, assuming that each single-rank DIMM has 18 components. The maximum load recommended by JEDEC standards and encountered in common systems is two unbuffered DIMMs. The resulting capacitive loading on the bus is extremely large, causing signals to have edges that take more than one clock period to rise and fall, thereby resulting in setup and hold violations at the memory device. Figure 12 shows the eye diagrams obtained by IBIS simulations using different configurations: one registered DIMM, one unbuffered DIMM, and two single-rank unbuffered DIMMs. The capacitive loads range from 2 for the registered DIMM to 36 for the unbuffered DIMM.



WP60_13_011107

Figure 12: Address and Command Signal Eye Openings Obtained with IBIS Simulations of DIMM Interfaces with Different Loads on the Address Bus

These eye diagrams clearly show the effect of capacitive loading on the address bus; the registered DIMMs offer a wide open valid window on the address and command bus. The eye opening for one DIMM appears to still be good at 267 MHz. However, with 32 loads, the address and command signal valid window has collapsed, and the conventional implementation is no longer sufficient to interface reliably to the two unbuffered DIMMs.

This simple test case illustrates that the loading causes the edges to slow down significantly and the eye to close itself for higher frequencies. In systems where the load on the bus cannot be reduced, lowering the clock frequency of operation is one way to keep the integrity of the signals acceptable. There are ways, however, to resolve the capacitive loading issue without a decrease in clock frequency:

- In applications where adding one clock cycle of latency on the interface is applicable, using Registered DIMMs can be a good option. These DIMMs use a register to buffer signals like address and command, reducing the capacitive loading.
- Using the design technique based on two clock periods (called 2T timing) on address and command signals, the address and command signals are transmitted at half the system clock frequency.

Managing the cost of a memory system is as much of a challenge as achieving the required performance. One way to reduce the board design complexity and bill of materials is to use on-chip terminations rather than resistors on the board. The Virtex-4 and Virtex-5 family of FPGAs offer a feature called Digitally Controlled Impedance (DCI) that can be implemented in the design to reduce the number of resistors on the board. The MIG tool has a built-in option to implement the memory interface design with this feature for address, control, or data bus. One tradeoff to consider in this case is the on-chip versus off-chip power consumption when terminations are implemented on-chip.

Development Boards for Memory Interfaces

Hardware verification of reference designs is an important final step to ensure a robust and reliable solution. Xilinx has verified the memory interface designs for both Spartan-3 Generation and Virtex-4 and Virtex-5 FPGAs. [Table 2](#) shows memory interfaces supported for each of the development boards.

Table 2: Development Boards for Memory Interfaces

Xilinx FPGA	Spartan-3	Spartan-3E	Spartan-3A	Virtex-4	Virtex-5
Development Board	SL-361	Starter Kit	Starter Kit	ML-461	ML-561
Memory Interfaces Supported	DDR	DDR	DDR2	DDR DDR2 QDR-II RLDRAM II	DDR DDR2 QDR-II RLDRAM II

The development boards range from low-cost Spartan-3 Generation FPGA implementations to the high-performance solutions offered by the Virtex-4 and Virtex-5 FPGA family of devices.

Conclusion

Equipped with the right FPGA, software tools, and development boards, the memory interface controller design can be a fast and trouble-free process from low-cost applications to high-performance designs using the 667 Mb/s DDR2 SDRAMs.

Please refer to www.xilinx.com/memory for more information and details on these memory interfaces solutions.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/16/07	1.0	Initial Xilinx release.