

Issue 1
March 2006



Memory Interfaces

Solution Guide

Overcoming Memory Interface Bottlenecks

INSIDE

ARTICLES

Implementing
High-Performance
Memory Interfaces
with Virtex-4 FPGAs

Meeting Signal Integrity
Requirements in FPGAs

How to Detect Potential
Memory Problems Early
in FPGA Designs

APPLICATION NOTES

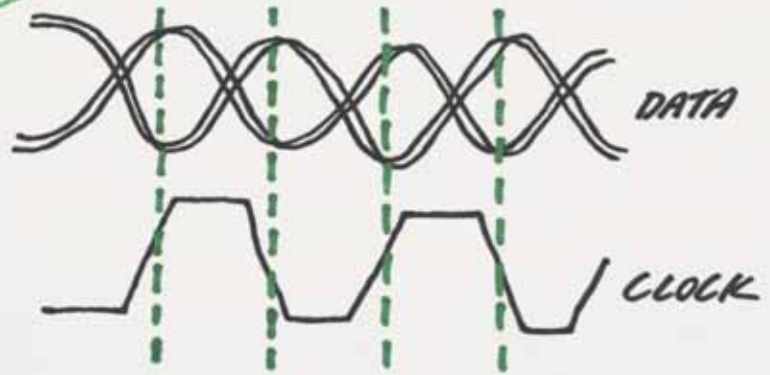
667 Mbps DDR2 SDRAM
Interface Solution
Using Virtex-4 FPGAs



Dr. Howard Johnson
The world's foremost
authority on
signal integrity

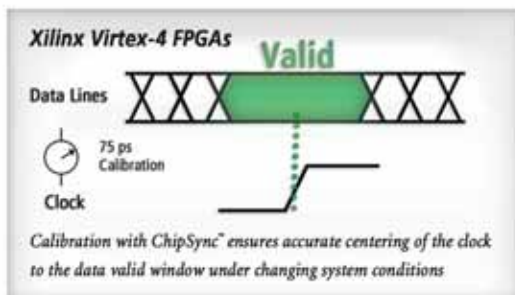
*THE VIRTEx
ADVANTAGE*

DDR2 SDRAM



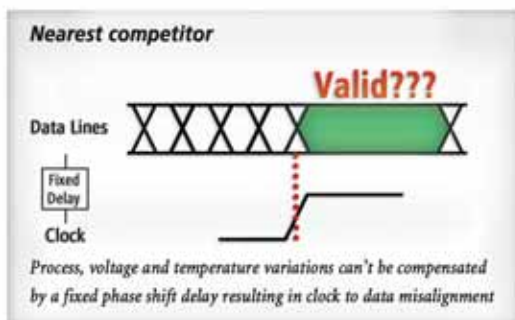
1. GOOD DATA EYE
2. CLOCK TO DATA CENTERING

**"A RELIABLE FPGA
MUST ADAPT TO YOUR
MEMORY TIMING."**



Fastest Memory Interfaces: 75 ps adaptive calibration

Supporting **667 Mbps DDR2 SDRAM** interfaces, Virtex-4 FPGAs achieve the highest bandwidth benchmark in the industry. Based on our unique ChipSync™ technology—built into every I/O—the Virtex-4 family provides adaptive centering of the clock to the data valid window. By providing reliable data capture (critical to high-performance memory interfacing), and 75 ps resolution for maximum design margins, your memory design can now adapt to changing system conditions.



HIGH SIGNAL INTEGRITY MEANS HIGH-BANDWIDTH MEMORY INTERFACES

Virtex-4 FPGAs deliver the industry's best signal integrity for high-speed and wide data bus designs—7x less Simultaneous Switching Output (SSO) noise, a critical factor in ensuring reliable interfaces. And Xilinx provides hardware verified interface solutions for popular memories such as DDR2 SDRAM, QDR II SRAM, and more. The lab results speak for themselves. As measured by industry expert Dr. Howard Johnson, no competing FPGA comes close to achieving the signal integrity of Virtex-4 devices.

Visit www.xilinx.com/virtex4/memory today, and start your next memory interface design for Virtex-4 FPGAs with the easy-to-use Memory Interface Generator software.



Dr. Howard Johnson, author of *High-Speed Digital Design*, frequently conducts technical workshops for digital engineers at Oxford University and other sites worldwide. Visit www.sigcon.com to register.



www.xilinx.com/virtex4/memory



View The
TechOnline
Seminar Today

BREAKTHROUGH PERFORMANCE AT THE LOWEST COST

Memory Interfaces Solution Guide

PUBLISHER
Forrest Couch
forrest.couch@xilinx.com
408-879-5270

EDITOR
Charmaine Cooper Hussain

ART DIRECTOR
Scott Blair

ADVERTISING SALES
Dan Teie
1-800-493-5551



Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3400
Phone: 408-559-7778
FAX: 408-879-4780

© 2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx Logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.

The articles, information, and other materials included in this issue are provided solely for the convenience of our readers. Xilinx makes no warranties, express, implied, statutory, or otherwise, and accepts no liability with respect to any such articles, information, or other materials or their use, and any use thereof is solely at the risk of the user. Any person or entity using such information in any way releases and waives any claim it might have against Xilinx for any loss, damage, or expense caused thereby.

Faster, But More Challenging

Welcome to the *Memory Interfaces Solution Guide*, an educational journal of memory interface design and implementation solutions from Xilinx. Engineers in the semiconductor and electronics design community tasked to create high-performance system-level designs know well the growing challenge of overcoming memory interface bottlenecks. This guide seeks to bring light to current memory interface issues, challenges, and solutions, especially as they relate to extracting maximum performance in FPGA designs.

Toward the latter half of the 1990s, memory interfaces evolved from single-data-rate SDRAMs to double-data-rate (DDR) SDRAMs, the fastest of which is currently the DDR2 SDRAM, running at 667 Mbps/pin. Present trends indicate that these rates are likely to double every four years, potentially reaching 1.6 Gbps/pin by 2010. These trends present a serious problem to designers in that the time period during which you can reliably obtain read data – the data valid window – is shrinking faster than the data period itself.

This erosion of the data valid window introduces a new set of design challenges that require a more effective means of establishing and maintaining reliable memory interface performance.

Along with the performance issues that attend the new breed of high-performance memories, designers face a new set of memory controller design issues as well. The complexities and intricacies of creating memory controllers for these devices pose a wide assortment of challenges that suggest a need for a new level of integration support from the tools accompanying the FPGA.

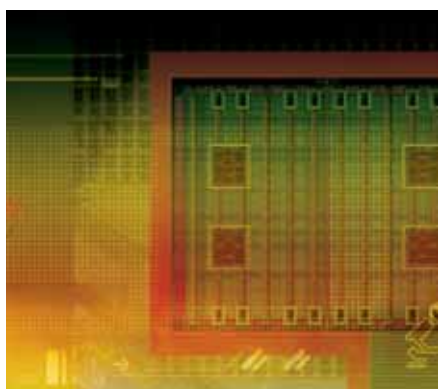
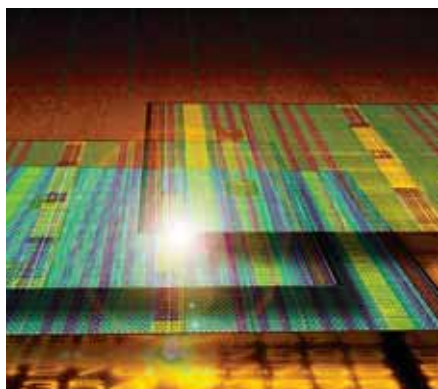
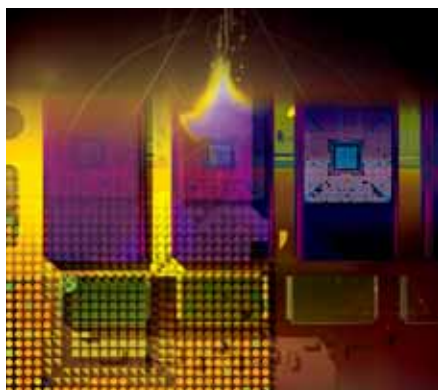
In this guide, we offer a foundational set of articles covering the broad selection of resources and solutions Xilinx offers, including the latest silicon features in the VirtexTM-4 FPGA family that address the shrinking data valid window, the availability of hardware reference designs to accelerate your design efforts, and two application notes that discuss the latest technology advances enabling the design of DDR2 SDRAM interfaces running at 667 Mbps/pin.

Enjoy!



Adrian Cosoroaba
Marketing Manager,
Product Solutions Marketing
Xilinx, Inc.

C O N T E N T S



ARTICLES

Implementing High-Performance Memory Interfaces with Virtex-4 FPGAs.....	5
Successful DDR2 Design.....	9
Designing a Spartan-3 FPGA DDR Memory Interface	14
Xilinx/Micron Partner to Provide High-Speed Memory Interfaces	16
Meeting Signal Integrity Requirements in FPGAs with High-End Memory Interfaces.....	18
How to Detect Potential Memory Problems Early in FPGA Designs	22
Interfacing QDR II SRAM with Virtex-4 FPGAs	25

APPLICATION NOTES

Memory Interfaces Reference Designs.....	27
DDR2 SDRAM Memory Interface for Spartan-3 FPGAs	30
DDR2 Controller (267 MHz and Above) Using Virtex-4 Devices.....	42
High-Performance DDR2 SDRAM Interface Data Capture Using ISERDES and OSERDES.....	55

EDUCATION

Signal Integrity for High-Speed Memory and Processor I/O	67
--	----

BOARDS

Virtex-4 Memory Interfaces	68
----------------------------------	----

Learn more about memory interfaces solutions
from Xilinx at: www.xilinx.com/xcell/memory1/

Implementing High-Performance Memory Interfaces with Virtex-4 FPGAs

You can center-align clock-to-read data at “run time” with ChipSync technology.

by Adrian Cosoroaba
Marketing Manager
Xilinx, Inc.
adrian.cosoroaba@xilinx.com

As designers of high-performance systems labor to achieve higher bandwidth while meeting critical timing margins, one consistently vexing performance bottleneck is the memory interface. Whether you are designing for an ASIC, ASSP, or FPGA, capturing source-synchronous read data at transfer rates exceeding 500 Mbps may well be the toughest challenge.

Source-Synchronous Memory Interfaces

Double-data rate (DDR) SDRAM and quad-data-rate (QDR) SRAM memories utilize source-synchronous interfaces through which the data and clock (or strobe) are sent from the transmitter to the receiver. The clock is used within the receiver interface to latch the data. This eliminates interface control issues such as the time of signal flight between the memory and the FPGA, but raises new challenges that you must address.

One of these issues is how to meet the various read data capture requirements to implement a high-speed source-synchronous interface. For instance, the receiver must ensure that the clock or strobe is routed to all data loads while meeting the required input setup and hold timing. But source-synchronous devices often limit the loading of the forwarded clock. Also, as the data-valid window becomes smaller at higher frequencies, it becomes more important (and simultaneously more challenging) to align the received clock with the center of the data.

Traditional Read Data Capture Method

Source-synchronous clocking requirements are typically more difficult to meet when reading from memory compared with writing to memory. This is because the DDR and DDR2 SDRAM devices send the data edge aligned with a non-continuous strobe signal instead of a continuous clock. For low-frequency interfaces up to 100 MHz, DCM phase-shifted outputs can be used to capture read data.

Capturing read data becomes more challenging at higher frequencies. Read data can be captured into configurable logic blocks (CLBs) using the memory read strobe, but the strobe must first be delayed so that its edge coincides with the center of the data valid window. Finding the correct phase-shift value is further complicated by process, voltage, and temperature (PVT) variations. The delayed strobe must also be routed onto low-skew FPGA clock resources to maintain the accuracy of the delay.

The traditional method used by FPGA, ASIC, and ASSP controller-based designs employs a phase-locked loop (PLL) or delay-locked loop (DLL) circuit that guarantees a fixed phase shift or delay between the source clock and the clock used for capturing data (Figure 1). You can insert this phase shift to accommodate estimated process, voltage, and temperature variations. The obvious drawback with this method is that it fixes the delay to a single value predetermined during the design phase. Thus, hard-to-predict variations within the system itself – caused by different routing to different memory devices, variations between FPGA or ASIC devices, and ambient system condi-

tions (voltage, temperature) – can easily create skew whereby the predetermined phase shift is ineffectual.

These techniques have allowed FPGA designers to implement DDR SDRAM memory interfaces. But very high-speed 267

also cause data and address timing problems at the input to the RAM and the FPGA's I/O blocks (IOB) flip-flop. Furthermore, as a bidirectional and non-free-running signal, the data strobe has an increased jitter component, unlike the clock signal.

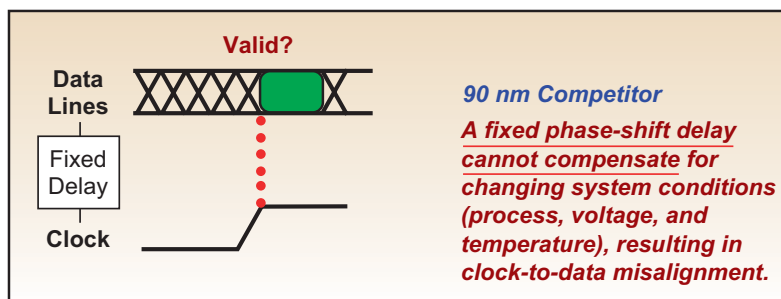


Figure 1 – Traditional fixed-delay read data capture method

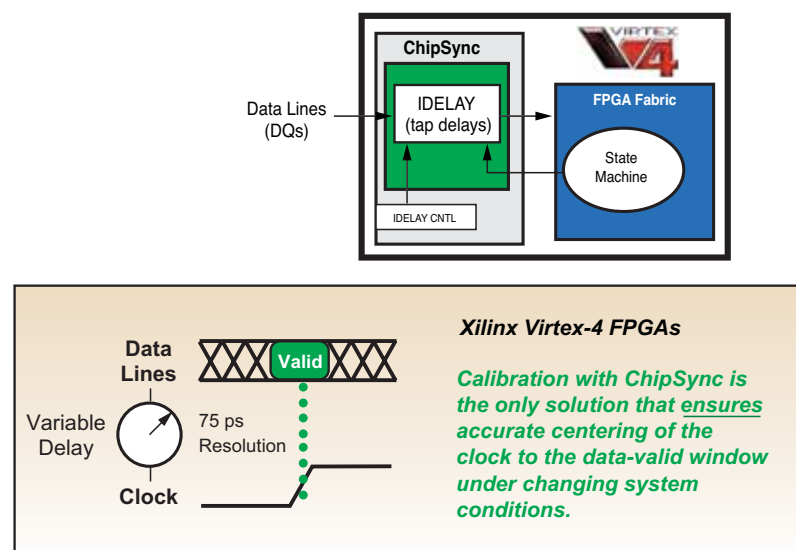


Figure 2 – Clock-to-data centering using ChipSync tap delays

MHz DDR2 SDRAM and 300 MHz QDR II SRAM interfaces demand much tighter control over the clock or strobe delay.

System timing issues associated with setup (leading edge) and hold (trailing edge) uncertainties further minimize the valid window available for reliable read data capture. For example, 267 MHz (533 Mbps) DDR2 read interface timings require FPGA clock alignment within a .33 ns window.

Other issues also demand your attention, including chip-to-chip signal integrity, simultaneous switching constraints, and board layout constraints. Pulse-width distortion and jitter on clock or data strobe signals

Clock-to-Data Centering Built into Every I/O

Xilinx® Virtex™-4 FPGAs with dedicated delay and clocking resources in the I/O blocks – called ChipSync™ technology – answer these challenges. These devices make memory interface design significantly easier and free up the FPGA fabric for other purposes. Moreover, Xilinx offers a reference design for memory interface solutions that center-aligns the clock to the read data at “run time” upon system initialization. This proven methodology ensures optimum performance, reduces engineering costs, and increases design reliability.

ChipSync features are built into every I/O. This capability provides additional flexibility if you are looking to alleviate board layout constraints and improve signal integrity.

ChipSync technology enables clock-to-data centering without consuming CLB resources. Designers can use the memory read strobe purely to determine the phase relationship between the FPGA's own DCM clock output and the read data. The read data is then delayed to center-align the

determine the phase relationship between the FPGA clock and the read data received at the FPGA. This is done using the memory read strobe. Based on this phase relationship, the next step is to delay read data to center it with respect to the FPGA clock. The delayed read data is then captured

transition (second-edge taps) in the FPGA clock domain.

Having determined the values for first-edge taps and second-edge taps, the state machine logic can compute the required data delay. The pulse center is computed with these recorded values as $(\text{second-edge taps} - \text{first-edge taps})/2$. The required data delay is the sum of the first-edge taps and the pulse center. Using this delay value, the data-valid window is centered with respect to the FPGA clock.

ChipSync features are built into every I/O. This capability provides additional flexibility if you are looking to alleviate board layout constraints and improve signal integrity.

Each I/O also has input DDR flip-flops required for read data capture either in the delayed memory read strobe domain or in the system (FPGA) clock domain. With these modes you can achieve higher design performance by avoiding half-clock-cycle data paths in the FPGA fabric.

Instead of capturing the data into a CLB-configured FIFO, the architecture provides dedicated 500 MHz block RAM with built-in FIFO functionality. These enable a reduction in design size, while leaving the CLB resources free for other functions.

Clock-to-Data Phase Alignment for Writes

Although the read operations are the most challenging part of memory interface design, the same level of precision is required in write interface implementation. During a write to the external memory device, the clock/strobe must be transmitted center-aligned with respect to data. In the Virtex-4 FPGA I/O, the clock/strobe is generated using the output DDR registers clocked by a DCM clock output (CLK0) on the global clock network. The write data is transmitted using the output DDR registers clocked by a

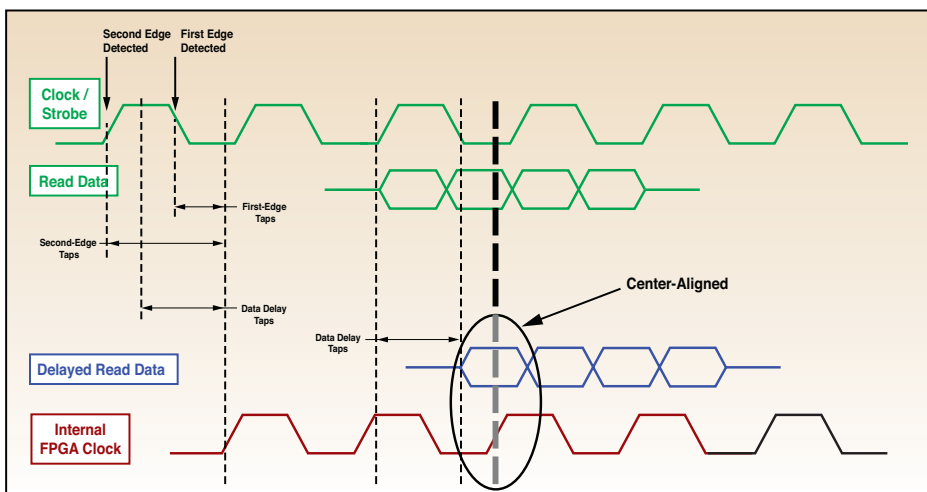


Figure 3 – Clock-to-data centering at “run time”

FPGA clock in the read data window for data capture. In the Virtex-4 FPGA architecture, the ChipSync I/O block includes a precision delay block known as IDELAY that can be used to generate the tap delays necessary to align the FPGA clock to the center of the read data (Figure 2).

Memory read strobe edge-detection logic uses this precision delay to detect the edges of the memory read strobe from which the pulse center can be calculated in terms of the number of delay taps counted between the first and second edges. Delaying the data by this number of taps aligns the center of the data window with the edge of the FPGA DCM output. The tap delays generated by this precision delay block allow alignment of the data and clock to within 75 ps resolution.

The first step in this technique is to

directly in input DDR flip-flops in the FPGA clock domain.

The phase detection is performed at run time by issuing dummy read commands after memory initialization. This is done to receive an uninterrupted strobe from the memory (Figure 3).

The goal is to detect two edges or transitions of the memory read strobe in the FPGA clock domain. To do this, you must input the strobe to the 64-tap IDELAY block that has a resolution of 75 ps. Then, starting at the 0-tap setting, IDELAY is incremented one tap at a time until it detects the first transition in the FPGA clock domain. After recording the number of taps it took to detect the first edge (first-edge taps), the state machine logic continues incrementing the taps one tap at a time until it detects the second

DCM clock output that is phase-offset 90 degrees (CLK270) with respect to the clock used to generate clock/strobe. This phase shift meets the memory vendor specification of centering the clock/strobe in the data window.

Another innovative feature of the output DDR registers is the SAME_EDGE mode of operation. In this mode, a third register clocked by a rising edge is placed on the input of the falling-edge register. Using this mode, both rising-edge and falling-edge data can be presented to the output DDR registers on the same clock edge (CLK270), thereby allowing higher DDR performance with minimal register-to-register delay.

Signal Integrity Challenge

One challenge that all chip-to-chip, high-speed interfaces need to overcome is signal integrity. Having control of cross-talk, ground bounce, ringing, noise margins, impedance matching, and decoupling is now critical to any successful design.

The Xilinx column-based ASMBL architecture enables I/O, clock, and power and ground pins to be located anywhere on the silicon chip, not just along the periphery. This architecture alleviates the problems associated with I/O and array dependency, power and ground distribution, and hard-IP scaling. Special FPGA packaging technology known as SparseChevron enables distribution of power and ground pins evenly across the package. The benefit to board designers is improved signal integrity.

The pin-out diagram in Figure 4 shows how Virtex-4 FPGAs compare with a competing Altera Stratix-II device that has many regions devoid of returns.

The SparseChevron layout is a major reason why Virtex-4 FPGAs exhibit unmatched simultaneous switching output (SSO) performance. As demonstrated by signal integrity expert Howard Johnson, Ph.D., these domain-optimized FPGA devices have seven times less SSO noise and crosstalk when compared to alternative FPGA devices (Figure 5).

Meeting I/O placement requirements and enabling better routing on a board requires unrestricted I/O placements for

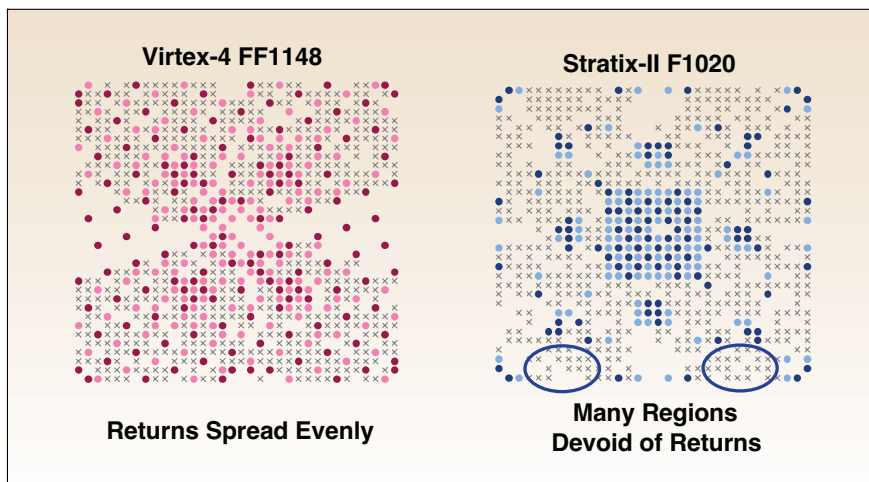


Figure 4 – Pin-out comparison between Virtex-4 and Stratix-II FPGAs

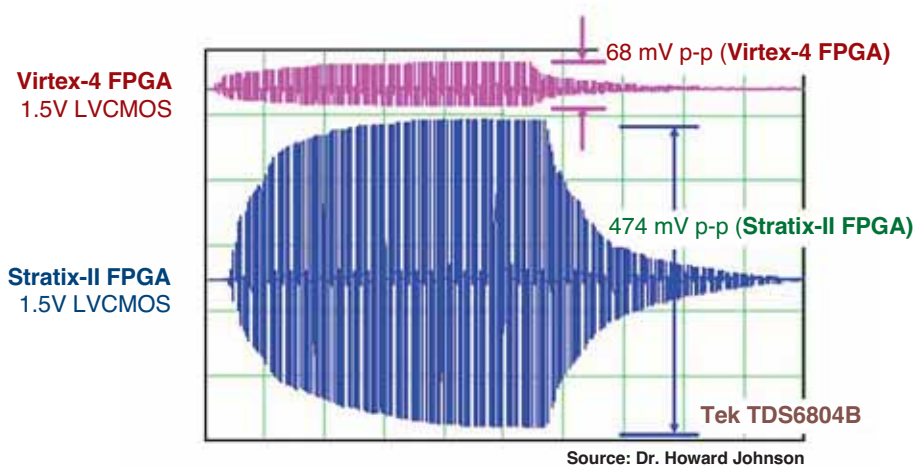


Figure 5 – Signal integrity comparison using the accumulated test pattern

an FPGA design. Unlike competing solutions that restrict I/O placements to the top and bottom banks of the FPGA and functionally designate I/Os with respect to address, data and clock, Virtex-4 FPGAs provide unrestricted I/O bank placements.

Finally, Virtex-4 devices offer a differential DCM clock output that delivers the extremely low jitter performance necessary for very small data-valid windows and diminishing timing margins, ensuring a robust memory interface design.

These built-in silicon features enable high-performance synchronous interfaces for both memory and data communications in single or differential mode. The ChipSync technology enables data rates

greater than 1 Gbps for differential I/O and more than 600 Mbps for single-ended I/O.

Conclusion

As with most FPGA designs, having the right silicon features solves only part of the challenge. Xilinx also provides complete memory interface reference designs that are hardware-verified and highly customizable. The Memory Interface Generator, a free tool offered by Xilinx, can generate all of the FPGA design files (.rtl, .ucf) required for a memory interface through an interactive GUI and a library of hardware-verified designs.

For more information, visit www.xilinx.com/memory.

Successful DDR2 Design

Mentor Graphics highlights design issues and solutions for DDR2, the latest trend in memory design.

by Steve McKinney
HyperLynx Technical Marketing Engineer
Mentor Graphics
steven_mckinney@mentor.com

The introduction of the first SDRAM interface, in 1997, marked the dawn of the high-speed memory interface age. Since then, designs have migrated through SDR (single data rate), DDR (double data rate), and now DDR2 memory interfaces to sustain increasing bandwidth needs in products such as graphics accelerators and high-speed routers. As a result of its high-bandwidth capabilities, DDR and DDR2 technology is used in nearly every sector of the electronics design industry – from computers and networking to consumer electronics and military applications.

DDR technology introduced the concept of “clocking” data in on both a rising and falling edge of a strobe signal in a memory interface. This provided a 2x bandwidth improvement over an SDR interface with the same clock speed. This, in addition to faster clock frequencies, allowed a single-channel DDR400 interface with a 200 MHz clock to support up to 3.2 GB/s, a 3x improvement over the fastest SDR interface. DDR2 also provided an additional 2x improvement in bandwidth over its DDR predecessor by doubling the maximum clock frequency to 400 MHz. Table 1 shows how the progression from SDR to DDR and DDR2 has allowed today’s systems to maintain their upward growth path.

SDR		DDR				DDR2			
PC100	PC133	DDR - 200	DDR - 266	DDR - 333	DDR - 400	DDR2 - 400	DDR2 - 533	DDR2 - 667	DDR2 - 800
0.8	1.1	1.6	2.1	2.7	3.2	3.2	4.266	5.33	6.4
Single Channel Bandwidth (GB/s)									

Table 1 – The progression from SDR to DDR and DDR2 has allowed today's systems to maintain their upward growth path. Speed grades and bit rates are shown for each memory interface.

With any high-speed interface, as supported operating frequencies increase it becomes progressively more difficult to meet signal integrity and timing requirements at the receivers. Clock periods become shorter, reducing timing budgets to a point where you are designing systems with only picoseconds of setup or hold margins. In addition to these tighter timing budgets, signals tend to deteriorate because faster edge rates are needed to meet these tight timing parameters. As edge rates get faster, effects like overshoot, reflections, and crosstalk become more significant problems on the interface, which results in a negative impact on your timing budget. DDR2 is no exception, though the JEDEC standards committee has created several new features to aid in dealing with the adverse effects that reduce system reliability.

Some of the most significant changes incorporated into DDR2 include on-die termination for data nets, differential strobe signals, and signal slew rate derating for both data and address/command signals. Taking full advantage of these new features will help enable you to design a robust memory interface that will meet both your signal integrity and timing goals.

On-Die Termination

The addition of on-die termination (ODT) has provided an extra knob with which to dial in and improve signal integrity on the DDR2 interface. ODT is a dynamic termination built into the SDRAM chip and memory controller. It can be enabled or disabled depending on addressing conditions and whether a read or write operation is being performed, as shown in Figure 1. In addition to being able to turn termination off or on, ODT also offers the flexibility of different termi-

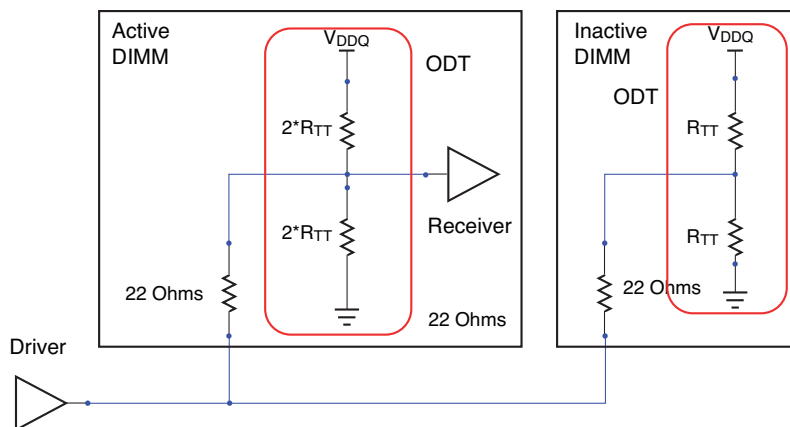


Figure 1 – An example of ODT settings for a write operation in a 2 DIMM module system where $R_{TT} = 150 \text{ Ohms}$.

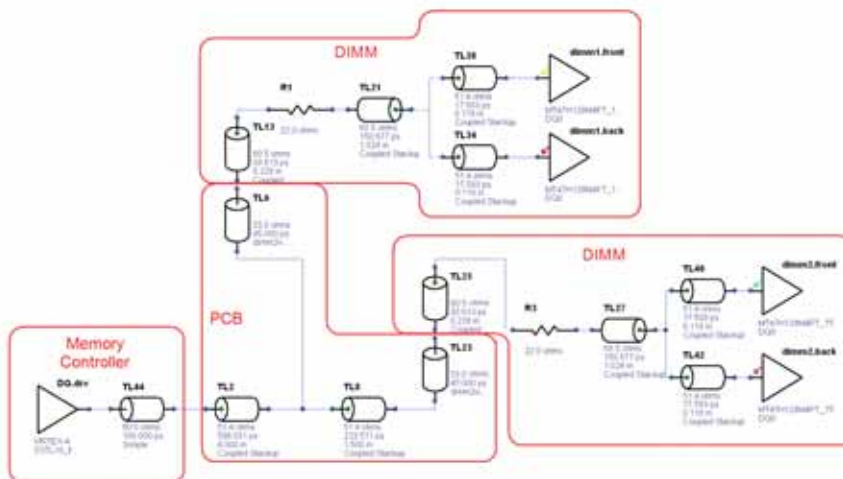


Figure 2 – The HyperLynx free-form schematic editor shows a pre-layout topology of an unbuffered 2 DIMM module system. Transmission line lengths on the DIMM are from the JEDEC DDR2 unbuffered DIMM specification.

nation values, allowing you to choose an optimal solution for your specific design.

It is important to investigate the effects of ODT on your received signals, and you can easily do this by using a signal integrity software tool like Mentor Graphics' HyperLynx product. Consider the example design shown in Figure 2, which shows a DDR2-533 interface (266 MHz) with two

unbuffered DIMM modules and ODT settings of 150 Ohms at each DIMM. You can simulate the effects of using different ODT settings and determine which settings would work best for this DDR2 design before committing to a specific board layout or creating a prototype.

With the 150 Ohm ODT settings, Figure 3 shows significant signal degrada-

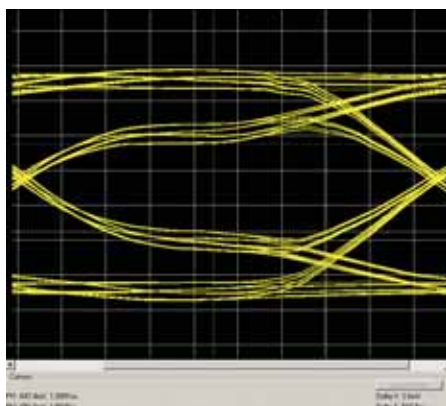


Figure 3 – The results of a received signal at the first DIMM in eye diagram form. Here, ODT settings of 150 Ohms are being used at both DIMM modules during a write operation. The results show there is an eye opening of approximately 450 ps outside of the VinAC switching thresholds.

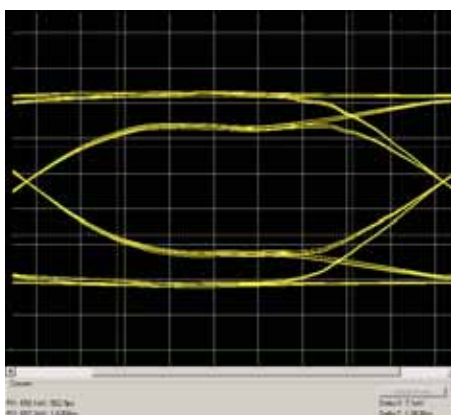


Figure 4 – This waveform shows a significant improvement in the eye aperture with a new ODT setting. Here, the ODT setting is 150 Ohms at the first DIMM and 75 Ohms at the second DIMM. The signal is valid for 1.064 ns with the new settings, which is an increase of 614 ps from the previous ODT settings.

tion at the receiver, resulting in eye closure. The eye shows what the signal looks like for all bit transitions of a pseudo-random (PRBS) bitstream, which resembles the data that you might see in a DDR2 write transaction. Making some simple measurements of the eye where it is valid outside the VinhAC and VinIAC thresholds, you can see that there is roughly a 450 ps window of valid signal at the first DIMM module.

It is appropriate to try to improve this eye aperture (opening) at the first DIMM if possible, and changing the ODT setting is one of the options available for this. To improve the signal quality at the first

DIMM, you must change the ODT value at the second DIMM. Setting the ODT at the second DIMM to 75 Ohms and re-running the simulation, Figure 4 shows more than a 100 percent increase in the eye aperture at the first DIMM, resulting in a 1.06 ns eye opening. As you can see, being able to dynamically change ODT is a powerful capability to improve signal quality on the DDR2 interface.

With respect to a DDR interface, ODT allows you to remove the source termination, normally placed at the memory controller, from the board. In addition, the pull-up termination to VTT at the end of the data bus is no longer necessary. This reduces component cost and significantly improves the layout of the board. By removing these terminations, you may be able to reduce layer count and remove unwanted vias on the signals used for layer transitions at the terminations.

Signal Slew Rate Derating

A challenging aspect of any DDR2 design is meeting the setup and hold time requirements of the receivers. This is especially true for the address bus, which tends to have significantly heavier loading conditions than the data bus, resulting in fairly slow edge rates. These slower edge rates can consume a fairly large portion of your timing budget, preventing you from meeting your setup and hold time requirements.

To enable you to meet the setup and hold requirements on address and data

buses, DDR2's developers implemented a fairly advanced and relatively new timing concept to improve timing on the interface: "signal slew rate derating." Slew rate derating provides you with a more accurate picture of system-level timing on the DDR2 interface by taking into account the basic physics of the transistors at the receiver.

For DDR2, when any memory vendor defines the setup and hold times for their component, they use an input signal that has a 1.0V/ns input slew rate. What if the signals in your design have faster or slower slew rates than 1.0V/ns? Does it make sense to still meet that same setup and hold requirement defined at 1.0V/ns? Not really. This disparity drove the need for slew rate derating on the signals specific to your design.

To clearly understand slew rate derating, let's consider how a transistor works. It takes a certain amount of charge to build up at the gate of the transistor before it switches high or low. Consider the 1.0V/ns slew rate input waveform between the switching region, Vref to Vin(h/l)AC, used to define the setup and hold times. You can define a charge area under this 1.0V/ns curve that would be equivalent to the charge it takes to cause the transistor to switch. If you have a signal that has a slew rate faster than 1.0V/ns, say 2.0V/ns, it transitions through the switching region much faster and effectively improves your timing margin. You've added some amount of timing margin into your system, but that was with the assumption of using the stan-

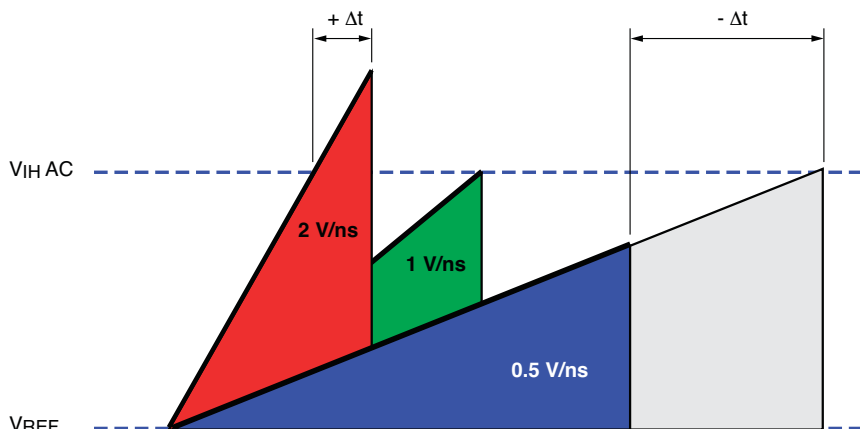


Figure 5 – A 1V/ns signal has a defined charge area under the signal between Vref and VinhAC. A 2V/ns signal would require a + Δt change in time to achieve the same charge area as the 1V/ns signal. A 0.5V/ns signal would require a - Δt change in time to achieve the same charge area as the 1V/ns signal. This change in time provides a clearer picture of the timing requirements needed for the receiver to switch.

Conversely, if you consider a much slower slew rate, such as 0.1V/ns, it would take a very long time to reach the switching threshold. You may never meet the setup and hold requirements in your timing budget with that slow of a slew rate through the transition region. This could cause you to overly constrain the design of your system, or potentially limit the con-

To assist you in meeting these timing goals, the memory vendors took this slew rate information into account and have constructed a derating table included in the DDR2 JEDEC specification (JESD79-2B on www.jedec.com). By using signal derating, you are now considering how the transistors at the receiver respond to charge building at their gates in your timing budgets. Although this adds a level of complexity to your analysis, it gives you more flexibility in meeting your timing goals, while also providing you with higher visibility into the actual timing of your system.

To properly use the derating tables, it is important to know how to measure the slew rate on a signal. Let's look at an example of a slew rate measurement for the rising edge of a signal under a setup condition.

The nominal slew rate is acceptable for use in the derating tables as long as the

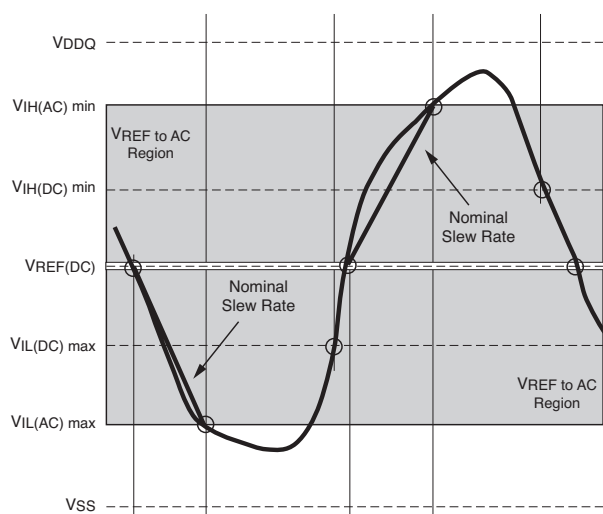


Figure 6 – The waveform illustrates how a nominal slew rate is defined for a signal when performing a derating in a setup condition. The waveform is taken from the DDR2 JEDEC specification (JESD79-2B).

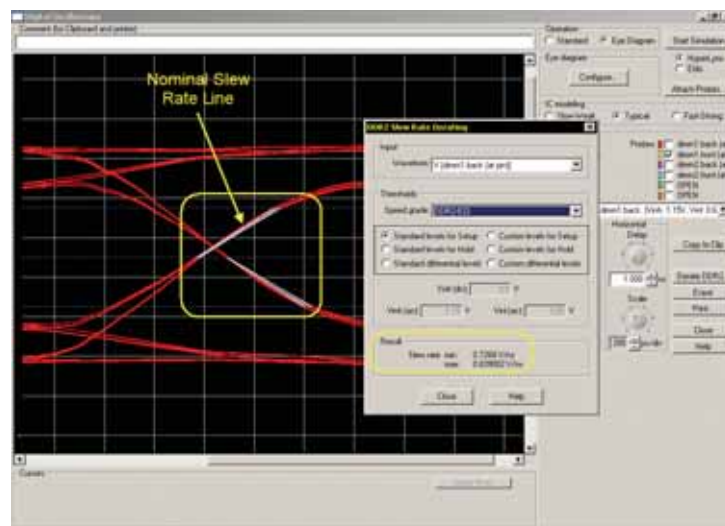


Figure 7 – The HyperLynx oscilloscope shows an automated measurement of the nominal slew rate for every edge in an eye diagram with the DDR2 slew rate derating feature. The measurement provides the minimum and maximum slew rates that can then be used in the DDR2 derating tables in the JEDEC specification.

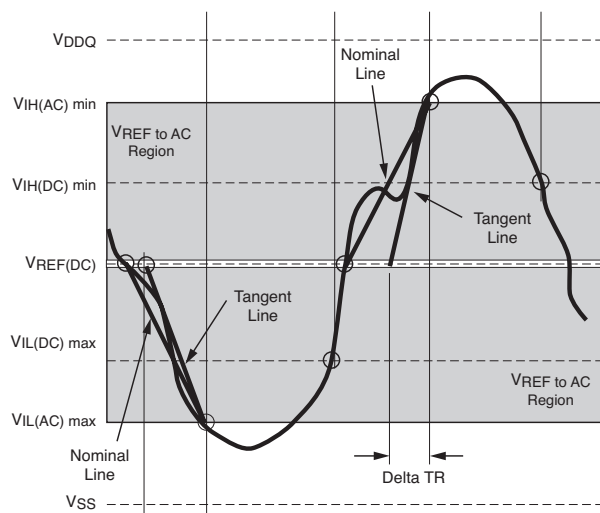


Figure 8 – This waveform, taken from the DDR2 JEDEC specification, shows how a tangent line must be found if any of the signal crosses the nominal slew rate line. The slew rate of this tangent line would then be used in the DDR2 derating tables.

received signal meets the condition of always being above (for the rising edge) or below (for the falling edge) the nominal slew rate line for a setup condition. If the signal does not have clean edges – possibly having some non-monotonicity or “shelf”-type effect that crosses the nominal slew rate line – you must define a new slew rate. This new slew rate is a tangent line on the received waveform that intersects with V_{inhAC} and the received waveform, as shown in Figure 8. The slew rate

of this new tangent line now becomes your slew rate for signal derating.

You can see in the example that if there is an aberration on the signal edge that would require you to find this new tangent line slew rate, HyperLynx automatically performs this check for you. If necessary, the oscilloscope creates the tangent line, which becomes part of the minimum and maximum slew rate results. As Figure 9 shows, the HyperLynx oscilloscope also displays all of the tangent lines,

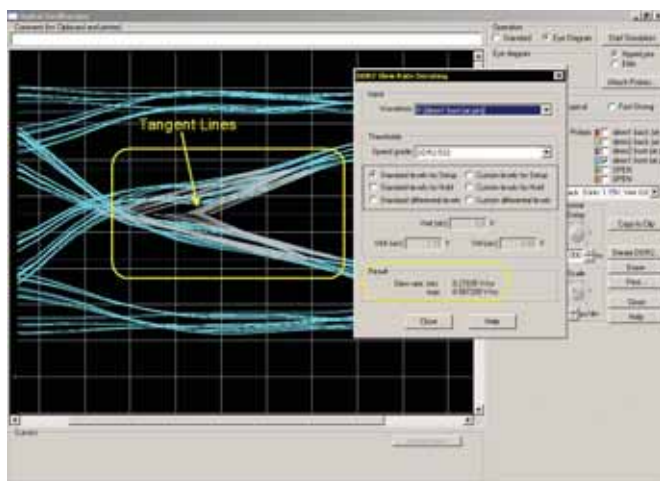


Figure 9 – The HyperLynx oscilloscope shows how the tangent line is automatically determined for you in the DDR2 slew rate derating feature. The slew rate lines in the display indicate that they are tangent lines because they no longer intersect with the received signal and V_{ref} intersection. The oscilloscope determines the slew rate of these new tangent lines for you and reports the minimum and maximum slew rates to be used in the derating tables.

making it easier to identify whether this condition is occurring.

For a hold condition, you perform a slightly different measurement for the slow rate. Instead of measuring from V_{ref} to the V_{inAC} threshold, you measure from V_{inDC} to V_{ref} to determine the nominal slew rate (shown in Figure 10). The same conditions regarding the nominal slew rate line and the inspection of the signal to determine the necessity for a tangent line for a new slew rate hold true here as well.

Conclusion

With the new addition of ODT, you've seen how dynamic on-chip termination can vastly improve signal quality. Performing signal derating per the DDR2 SDRAM specification has also shown that you can add as much as 1.42 ns back into your timing budget, giving you more flexibility in your PCB design and providing you with a better understanding of system timing.

Equipped with the right tools and an understanding of underlying technology, you will be able to move your designs from DDR to DDR2 in a reasonably pain-free process – realizing the added performance benefits and component-count reductions promised by DDR2. ●●

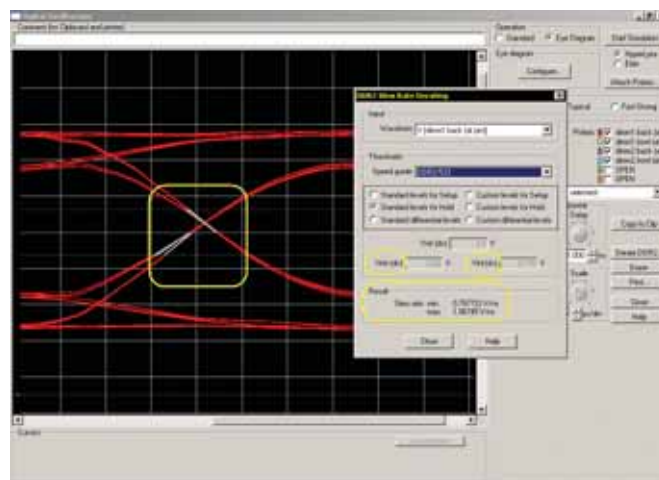


Figure 10 – The oscilloscope shows how a derating for a hold condition is being performed on the received signal. The DC thresholds are used in place of the AC switching thresholds, which are noted in the DDR2 derating dialog.

Designing a Spartan-3 FPGA DDR Memory Interface

Xilinx provides many tools to implement customized DDR memory interfaces.

by Rufino Olay
Marketing Manager, Spartan Solutions
Xilinx, Inc.
rufino.olay@xilinx.com

Karthikeyan Palanisamy
Staff Engineer, Memory Applications Group
Xilinx, Inc.
karthi.palanisamy@xilinx.com

Memory speed is a crucial component of system performance. Currently, the most common form of memory used is synchronous dynamic random access memory (SDRAM).

The late 1990s saw major jumps in SDRAM memory speeds and technology because systems required faster performance and larger data storage capabilities. By 2002, double-data-rate (DDR) SDRAM became the standard to meet this ever-growing demand, with DDR266 (initially), DDR333, and recently DDR400 speeds.

DDR SDRAM is an evolutionary extension of “single-data-rate” SDRAM and provides the benefits of higher speed, reduced power, and higher density components. Data is clocked into or out of the device on both the rising and falling edges of the clock. Control signals, however, still change only on the rising clock edge.

DDR memory is used in a wide range of systems and platforms and is the computing memory of choice. You can use Xilinx® Spartan™-3 devices to implement a custom DDR memory controller on your board.

Interfacing Spartan-3 Devices with DDR SDRAMs

Spartan-3 platform FPGAs offer an ideal connectivity solution for low-cost systems, providing the system-level building blocks necessary to successfully interface to the latest generation of DDR memories.

Included in all Spartan-3 FPGA input/output blocks (IOB) are three pairs

of storage elements. The storage-element pair on either the output path or the three-state path can be used together with a special multiplexer to produce DDR transmission. This is accomplished by taking data synchronized to the clock signal's rising edge and converting it to bits synchronized on both the rising and falling edge. The combination of two registers and a multiplexer is referred to as double-data-rate D-type flip-flop (FDDR).

Memory Controllers Made Fast and Easy

Xilinx has created many tools to get designers quickly through the process of building and testing memory controllers for Spartan devices. These tools include reference designs and application notes, the Memory Interface Generator (MIG), and more recently, a hardware test platform.

Xilinx application note XAPP454, “DDR2 SDRAM Memory Interface for Spartan-3 FPGAs,” describes the use of a Spartan-3 FPGA as a memory controller,

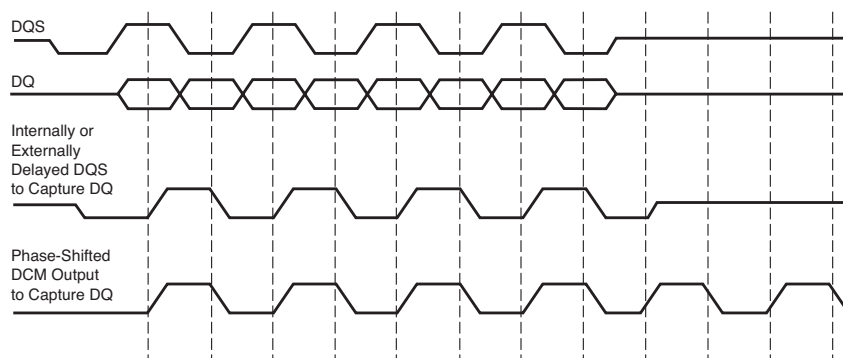


Figure 1 – Read operation timing diagram

with particular focus on interfacing to a Micron MT46v32M16TG-6T DDR SDRAM. This and other application notes illustrate the theory of operations, key challenges, and implementations of a Spartan-3 FPGA-based memory controller.

DDR memories use non-free-running strobes and edge-aligned read data (Figure 1). For 333 Mbps data speeds, the memory strobe must be used for higher margins. Using local clocking resources, a

delayed strobe can be centered in the data window for data capture.

To maximize resources within the FPGA, you can explore design techniques such as using the LUTs as RAMs for data capture – while at the same time minimizing the use of global clock buffers (BUFGs) and digital clock managers (DCMs) – as explained in the Xilinx application notes. Results are given with respect to the maximum data width per FPGA side for either right and left

or top and bottom implementations. Implementation challenges such as these are mitigated with the new Memory Interface Generator.

Xilinx created the Memory Interface Generator (MIG 007) to take the guesswork out of designing your own controller. To create the interface, the tool requires you to input data including FPGA device, frequency, data width, and banks to use. The interactive GUI (Figure 2) generates the RTL, EDIF, SDC, UCF, and related document files.

As an example, we created a DDR 64-bit interface for a Spartan XC3S1500-5FG676

using MIG. The results in Table 1 show that the implementation would use 17% of the slices, leaving more than 80% of the device free for data-processing functions.

Testing Out Your Designs

The last sequence in a design is the verification and debug in actual hardware. After using MIG 007 to create your customized memory controller, you can implement your design on the Spartan-3 Memory Development Kit, HW-S3-SL361, as shown in Figure 3. The \$995 kit is based on a Spartan-3 1.5M-gate FPGA (the XC3S1500) and includes additional features such as:

- 64 MB of DDR SDRAM Micron MT5VDDT1672HG-335, with an additional 128 MB DDR SDRAM DIMM for future expansion
- Two-line LCD
- 166 MHz oscillator
- Rotary switches
- Universal power supply 85V-240V, 50-60 MHz



Figure 3 – Spartan-3 memory development board (HW-S3-SL361)

Conclusion

With the popularity of DDR memory increasing in system designs, it is only natural that designers use Spartan-3 FPGAs as memory controllers. Implementing the controller need not be difficult.

For more information about the application notes, GUI, and development board, please visit www.xilinx.com/products/design_resources/mem_corner/index.htm.

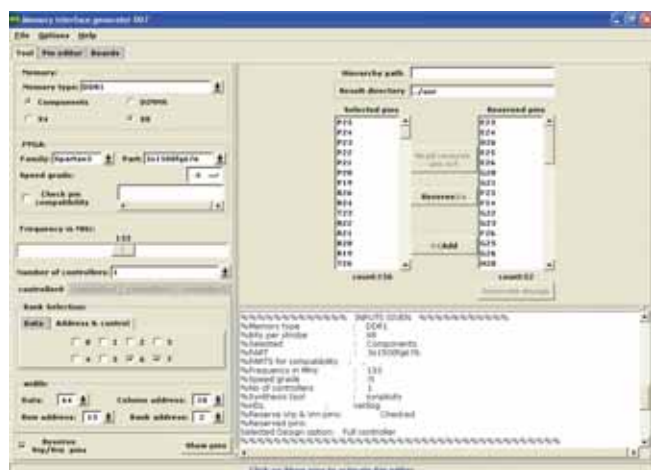


Figure 2 – Using the MIG 007 to automatically create a DDR memory controller

Feature	Utilization	Percent Used
Number of Slices	2,277 out of 13,312	17%
Number of DCMs	1 out of 4	25%
Number of External IOBs	147 out of 487	30%

Table 1 – Device utilization for a DDR 64-bit interface in an XC3S1500 FPGA

Xilinx/Micron Partner to Provide High-Speed Memory Interfaces

Micron's RLD RAM II and DDR/DDR2 memory combines performance-critical features to provide both flexibility and simplicity for Virtex-4-supported applications.

by Mike Black
Strategic Marketing Manager
Micron Technology, Inc.
mblack@micron.com

With network line rates steadily increasing, memory density and performance are becoming extremely important in enabling network system optimization. Micron Technology's RLD RAM™ and DDR2 memories, combined with Xilinx® Virtex-4™ FPGAs, provide a platform designed for performance.

This combination provides the critical features networking and storage applications need: high density and high bandwidth. The ML461 Advanced Memory Development System (Figure 1) demonstrates high-speed memory interfaces with Virtex-4 devices and helps reduce time to market for your design.

Micron Memory

With a DRAM portfolio that's among the most comprehensive, flexible, and reliable in the industry, Micron has the ideal solution to enable the latest memory platforms. Innovative new RLD RAM and DDR2 architectures are advancing system designs farther than ever, and Micron is at the forefront, enabling customers to take advantage of the new features and functionality of Virtex-4 devices.

RLD RAM II Memory

An advanced DRAM, RLD RAM II memory uses an eight-bank architecture optimized for high-speed operation and a double-data-rate I/O for increased bandwidth. The eight-bank architecture enables

RLD RAM II devices to achieve peak bandwidth by decreasing the probability of random access conflicts.

In addition, incorporating eight banks results in a reduced bank size compared to typical DRAM devices, which use four. The smaller bank size enables shorter address and data lines, effectively reducing the parasitics and access time.

Although bank management remains important with RLD RAM II architecture, even at its worst case (burst of two at 400 MHz operation), one bank is always available for use. Increasing the burst length of the device increases the number of banks available.

I/O Options

RLD RAM II architecture offers separate I/O (SIO) and common I/O (CIO) options. SIO devices have separate read and write ports to eliminate bus turnaround cycles and contention. Optimized for near-term read and write balance, RLD RAM II SIO devices are able to achieve full bus utilization.

In the alternative, CIO devices have a shared read/write port that requires one additional cycle to turn the bus around. RLD RAM II CIO architecture is optimized for data streaming, where the near-term bus operation is either 100 percent read or 100 percent write, independent of the long-term balance. You can choose an I/O version that provides an optimal compromise between performance and utilization.

The RLD RAM II I/O interface provides other features and options, including support for both 1.5V and 1.8V I/O lev-

els, as well as programmable output impedance that enables compatibility with both HSTL and SSTL I/O schemes. Micron's RLD RAM II devices are also equipped with on-die termination (ODT) to enable more stable operation at high speeds in multipoint systems. These features provide simplicity and flexibility for high-speed designs by bringing both end termination and source termination resistors into the memory device. You can take advantage of these features as needed to reach the RLD RAM II operating speed of 400 MHz DDR (800 MHz data transfer).

At high-frequency operation, however, it is important that you analyze the signal driver, receiver, printed circuit board network, and terminations to obtain good signal integrity and the best possible voltage and timing margins. Without proper terminations, the system may suffer from excessive reflections and ringing, leading to reduced voltage and timing margins. This, in turn, can lead to marginal designs and cause random soft errors that are very difficult to debug. Micron's RLD RAM II devices provide simple, effective, and flexible termination options for high-speed memory designs.

On-Die Source Termination Resistor

The RLD RAM II DQ pins also have on-die source termination. The DQ output driver impedance can be set in the range of 25 to 60 ohms. The driver impedance is selected by means of a single external resistor to ground that establishes the driver impedance for all of the device DQ drivers.

As was the case with the on-die end termination resistor, using the RLD RAM II

on-die source termination resistor eliminates the need to place termination resistors on the board – saving design time, board space, material costs, and assembly costs, while increasing product reliability. It also eliminates the cost and complexity of end termination for the controller at that end of the bus. With flexible source termination, you can build a single printed circuit board with various configurations that differ only by load options, and adjust the Micron RLD RAM II memory driver impedance with a single resistor change.

DDR/DDR2 SDRAM

DRAM architecture changes enable twice the bandwidth without increasing the demand on the DRAM core, and keep the power low. These evolutionary changes enable DDR2 to operate between 400 MHz and 533 MHz, with the potential of extending to 667 MHz and 800 MHz. A summary of the functionality changes is shown in Table 1.

Modifications to the DRAM architecture include shortened row lengths for reduced activation power, burst lengths of four and eight for improved data bandwidth capability, and the addition of eight banks in 1 Gb densities and above.

New signaling features include on-die termination (ODT) and on-chip driver (OCD). ODT provides improved signal quality, with better system termination on the data signals. OCD calibration provides the option of tightening the variance of the pull-up and pull-down output driver at 18 ohms nominal.

Modifications were also made to the mode register and extended mode register, including column address strobe CAS latency, additive latency, and programmable data strobes.

Conclusion

The built-in silicon features of Virtex-4 devices – including ChipSync™ I/O technology, SmartRAM, and Xesium differential clocking – have helped simplify interfacing FPGAs to very-high-speed memory devices. A 64-tap 80 ps absolute delay element as well as input and output DDR registers are available in each I/O element, providing for the first time a run-time center alignment of data and clock that guarantees reliable data capture at high speeds.



Figure 1 – ML461 Advanced Memory Development System

Xilinx engineered the ML461 Advanced Memory Development System to demonstrate high-speed memory interfaces with Virtex-4 FPGAs. These include interfaces with Micron's PC3200 and PC2-5300 DIMM modules, DDR400 and DDR2533 components, and RLD RAM II devices.

In addition to these interfaces, the ML461 also demonstrates high speed QDR-II and FCRAM-II interfaces to

Virtex-4 devices. The ML461 system, which also includes the whole suite of reference designs to the various memory devices and the memory interface generator, will help you implement flexible, high-bandwidth memory solutions with Virtex-4 devices.

Please refer to the RLD RAM information pages at www.micron.com/products/dram/rlram/ for more information and technical details. 🌈

FEATURE/OPTION	DDR	DDR2
Data Transfer Rate	266, 333, 400 MHz	400, 533, 667, 800 MHz
Package	TSOP and FBGA	FBGA only
Operating Voltage	2.5V	1.8V
I/O Voltage	2.5V	1.8V
I/O Type	SSTL_2	SSTL_18
Densities	64 Mb-1 Gb	256 Mb-4 Gb
Internal Banks	4	4 and 8
Prefetch (MIN Write Burst)	2	4
CAS Latency (CL)	2, 2.5, 3 Clocks	3, 4, 5 Clocks
Additive Latency (AL)	No	0, 1, 2, 3, 4 Clocks
READ Latency	CL	AL + CL
WRITE Latency	Fixed	READ Latency - 1 Clock
I/O Width	x4/ x8/ x16	x4/ x8/ x16
Output Calibration	None	OCD
Data Strobes	Bidirectional Strobe (Single-Ended)	Bidirectional Strobe (Single-Ended or Differential) with RDQS
On-Die Termination	None	Selectable
Burst Lengths	2, 4, 8	4, 8

Table 1 – DDR/DDR2 feature overview

Meeting Signal Integrity Requirements in FPGAs with High-End Memory Interfaces

As valid signal windows shrink, signal integrity becomes a dominant factor in ensuring that high-end memory interfaces perform flawlessly.

by Olivier Despaux
Senior Applications Engineer
Xilinx Inc.
olivier.despaux@xilinx.com

Wider parallel data buses, increasing data rates, and multiple loads are challenges for high-end memory interface designers. The demand for higher bandwidth and throughput is driving the requirement for even faster clock frequencies. As valid signal windows shrink, signal integrity (SI) becomes a dominant factor in ensuring that memory interfaces perform flawlessly.

Chip and PCB-level design techniques can improve simultaneous switching output (SSO) characteristics, making it easier to achieve the signal integrity required in wider memory interfaces. EDA vendors are making a wide range of tools available to designers for optimizing the signal integrity quality of memory interfaces.

Features that are integrated on the FPGA silicon die, such as digitally controlled impedance (DCI), simplify the PCB layout design and enhance performance. This article discusses these design techniques and hardware experiment results, illustrating the effect of design parameters on signal integrity.

Optimizing Timing in DDR SDRAM Interfaces

Shrinking data periods and significant memory timing uncertainties are making timing closure a real challenge in today's higher performance electronic systems. Several design practices help in preserving the opening of the valid data window. For example, in the case of interfaces with DDR2 SDRAM devices, the JEDEC standard allows the memory device suppliers to have a substantial amount of skew on the data transmitted to the memory controller. There are several components to this skew

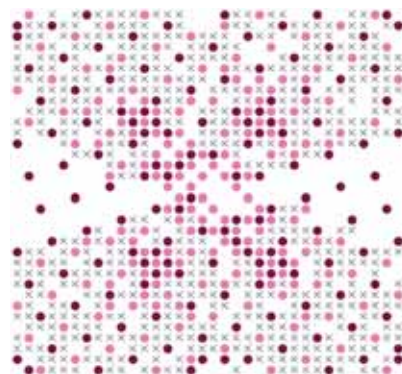
factor, including output access time, package and routing skew, and data-to-strobe skew. In the case where the memory controller is using a fixed phase-shift to register data across the entire interface, the sum of the skew uncertainties must be accounted for in the timing budget. If the worst-case sum of skew uncertainties is high, it reduces the data valid window and thereby limits the guaranteed performance for the interface. Table 1 shows an example of timing parameters for a 267 MHz memory interface.

Assuming that data capture is based on the timing of the DQS signals, leveraging the source-synchronous nature of the interface, it is possible to compute the memory valid data window across the entire data bus as follows:

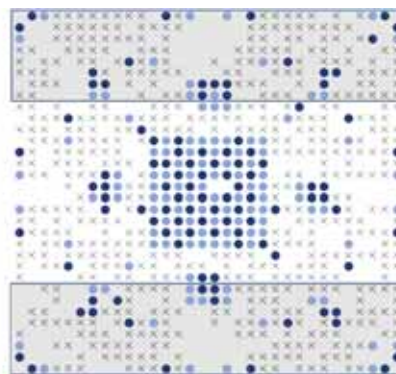
$$\begin{aligned} T_{\text{MEMORY_VDW}} &= T_{\text{DATA_PERIOD}} \\ &\quad - T_{\text{DQSCK}} - T_{\text{DQSQ}} - T_{\text{QHS}} \\ 1687 \text{ ps} - 900 \text{ ps} - 300 \text{ ps} - 400 \text{ ps} &= 87 \text{ ps} \end{aligned}$$

Uncertainty Parameters	Value (ps)	Uncertainty Before Clock	Uncertainty After Clock	Description
T_{CLOCK}	3750			Clock period
$T_{\text{MEM_DCD}}$	188	0	0	Duty cycle distortion tolerance is subtracted from clock phase to determine $T_{\text{DATA_PERIOD}}$
$T_{\text{DATA_PERIOD}}$	1687			Data period is half the clock period minus 10% of duty cycle distortion
T_{DQSQ}	300	300	0	Strobe-to-data distortion specified by memory vendor
T_{QHS}	400	0	400	Hold skew factor specified by memory vendor
T_{DQSK}	900	450	450	DQS output access time from CK/CK#

Table 1 – Memory parameters valid data window uncertainty summary



Pinout of Package with Distributed Power and Ground Pins
HSTL/SSTL Supported on All I/Os



Pinout of Package with Centered Power and Ground Pins
HSTL/SSTL Restricted to Top and Bottom

Figure 1 – Examples of advanced FPGA package pinouts

This equation sets the first condition for the data capture timing budget; the memory valid data window must be larger than the sampling window of the memory controller receiver, including setup and hold times and all the receiver timing uncertainties. Capturing data using a fixed delay or a fixed phase shift across the entire data bus is no longer sufficient to register data reliably. However, there are several other methods available. Among all the different options, the “direct clocking” data capture method is a very efficient way to register data bits and transfer them into

the memory controller clock domain. This method consists of detecting transitions on DQS signals and implementing the appropriate delay on data bits to center-align DQ signals with the memory controller clock. This technique also has the advantage of making the clock domain transfers from the memory to the controller efficient and reliable. When the calibration is used for:

- The entire interface, the delay on data bits is automatically adjusted independently of the system parameters.

- On one byte of data, the data hold skew factor, T_{DQSQ} , and the strobe-to-data distortion parameter, T_{QHS} , are removed from the timing budget equation.
- On one bit of data, the DQS-to-DQ skew uncertainty, T_{DQSK} , is accounted for in addition to the data hold skew factor, T_{DQSQ} , and the strobe-to-data distortion parameter, T_{QHS} . In this case, the valid data window is equal to the data period as provided by the memory device.

In high data rate systems that are subject to variations in voltage and temperature, dynamic calibration is required. In leading edge interfaces, performing the calibration sequence periodically makes this scheme independent of voltage and temperature variations at all times.

Increasing Bandwidth with Wider Data Buses: Distributed Power and Ground Pins

Meeting higher bandwidth requirements can be achieved by widening data buses. Interfaces of 144 or 288 bits are not uncommon nowadays. Memory controller device packages with many I/Os are required to achieve those wide buses. Numerous bits switching simultaneously can create signal integrity problems. The SSO limit is specified by the device vendor and represents the number of pins that the user can use simultaneously per bank in the device. This limit increases for devices that are architected to support a large number of I/Os and contain distributed power and ground pins. These packages offer better immunity against crosstalk.

Examples of package pinouts from two FPGA vendors are shown in Figure 1. The dots represent power and ground pins, the crosses represent pins available for the user.

These two devices have been used in an experiment emulating a 72-bit memory interface. The worst-case noise level on a user pin is six times smaller in the package with the distributed power and ground pins using SSTL 1.8V – the standard for DDR2 interfaces – with external terminations. For wide interfaces, crosstalk and data dependent jitter can be major contributors to the timing budget. For wide interfaces,

crosstalk and data dependent jitter can be major contributors to the timing budget and cause setup and hold time violations. The data dependent jitter depends on the transitions on the data bus (examples of possible data transitions: “Z-0-1-Z”, “1-1-1-1” or “0-1-0-1” transitions). For design using I/Os extensively, distributed power and ground packages and careful PCB design are elements that add to the stability and robustness of the electronic system.

The Challenge of Capacitive Loading on the Bus

When designing a large memory interface system, cost, density, throughput, and latency are key factors in determining the choice of interface architecture. In order to achieve the desired results, one solution is to use multiple 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. It causes these signals to have edges that take more than one clock period to rise and fall resulting in setup and hold violations at the memory device. The eye diagrams obtained by IBIS simulations using different configurations: one registered DIMM, one unbuffered DIMM, and two single rank unbuffered DIMMs, are shown in Figure 2. The capacitive loads range from 2 for the registered DIMM to 36 for the unbuffered DIMM.

These eye diagrams clearly show the effect of loading on the address bus; the registered DIMMs offer a wide open valid window on the ADDR CMD bus. The eye opening for one DIMM appears to be still good at 267 MHz; however, with 32 loads, the ADDR CMD valid window is collapsed, and the conventional implementation is no longer sufficient to interface reliably to the 2 unbuffered DIMMs.

The timing characteristics of falling-

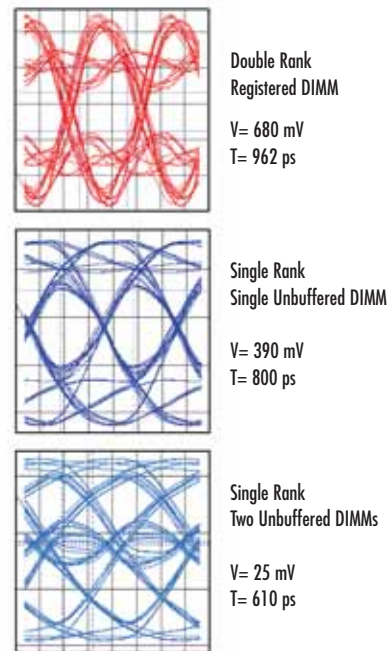


Figure 2 – ADDR CMD signals eye openings obtained with IBIS simulations of DIMM interfaces with different loads on the address bus

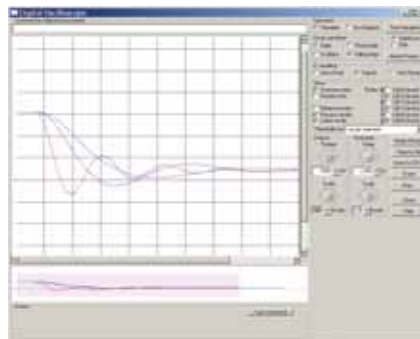


Figure 3 – Falling edge timing of ADDR CMD signals obtained with IBIS simulation of DIMM interfaces with different loads on the address bus

edges on the same signal under the same loading conditions using IBIS simulations are shown in Figure 3.

This simple test case illustrates that the loading causes the edges to slow down significantly and the eye to close itself past a certain frequency. In systems where the load on the bus cannot be reduced, lowering the frequency of operation is one way to keep the integrity of the signals acceptable.

Each load has a small capacitance that adds up on the bus. However, the driver has a fixed or limited current drive strength. Because the voltage is inversely

proportional to the capacitive load (as shown in the following equation); once the driver is saturated, rising and falling edges become slower:

$$v(t) = \frac{1}{C} \cdot \int i(t) \cdot dt$$

The result is a limitation of the maximum clock frequency that can be achieved with a fixed configuration: there is an instance when the edges are slow to the point of limiting the performance of the interface. This limitation is presented in Figure 4, which shows the experimental analysis of performance limitation versus capacitive load on the bus.

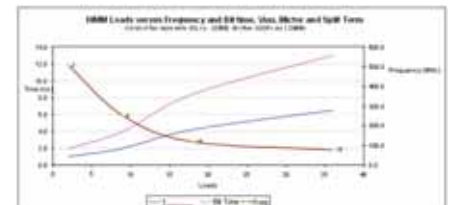


Figure 4 – Maximum possible clock rate based on address bus loading in Xilinx FPGAs with DDR2 SDRAM devices

There are several ways to resolve the capacitive loading issue:

- Duplicate the signals that have an excessive load across the interface. For example, replicating address and command signals every 4 or 8 loads can be very efficient in ensuring high quality signal integrity characteristics on these signals.
- 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 heavily loaded signals like address and command signals. In exchange for one additional latency cycle in the address and command signals, these modules drastically reduce the load on control and address signals by a factor of 4 to 18, thereby helping with the capacitive loading problem.

- Use the design technique based on two clock periods on address and command signals. More details on this method are presented in the following section.

Using Two Clock Periods

The use of unbuffered DIMMs can be required:

- When latency is the preponderant performance factor. If the memory accesses are short and at random locations in the memory array, adding one clock cycle of latency will degrade the data bus utilization and the overall performance of the interface decreases. In this case, a slower interface with minimal cycles of latency can be more efficient than an interface running faster with one more clock cycle of latency. However, when the memory is accessed in several bursts of adjacent locations, the faster clock rate compensates for the initial latency, and the overall performance increases.
- When cost is sensitive and the additional premium for using a registered DIMM is not a possibility.
- When hardware is already available but has to support a deeper memory interface or a faster clock rate.
- When the number of pins for the memory controller is fixed by an existing PCB or a feature set, and the additional pinout for registered DIMMs is not available.

In these cases, the design technique using two clock periods to transmit signals on heavily loaded buses can be utilized to resolve the capacitive loading on the address and command bus effectively. However, the controller will be able to present data only every two clock cycles to the memory, reducing the efficiency of the interface in certain applications.

The principle for the two period clocking on address and command buses consists of pre-launching command and address signals (ADDRCMD) one clock period in advance of loading data bits and keeps these signals valid for two clock periods. This leaves more time for the address

and command signals to rise and meet the setup and hold time memory requirements. The control signals, such as Chip Select signals, that have a load limited to components of one rank of a DIMM, are used to indicate the correct time for the memory to load address and command signals. The design technique has been successfully tested and characterized in multiple systems.

Reduce the BOM and Simplify PCB Layout: Use On-Chip Terminations

One way to reduce the design complexity and bill-of-materials (BOM) for a board is to use on-chip terminations. The JEDEC industry standard for DDR2 SDRAM has defined the on-die termination (ODT feature). This feature provides an embedded termination apparatus on the die of the device that eliminates the need for external PCB terminations on data, strobe, and data mask (DQ, DQS and DM) signals. However, the other signals still require external termination resistors on the PCB.

For DDR2, memory vendors have also increased the quality of signal integrity of the signals on DIMM modules compared to the original DDR1 devices. For example, they match all flight times and loading on a given signal, reducing the effect of stubs and reflection on the transmission lines. But using ODT also requires running additional signal integrity simulations, because the configuration of terminations and loading on the bus changes based on the number of populated sockets. Based on the JEDEC matrix for write operations or read operations, the memory controller should be able to turn the ODT terminations on or off depending on how the memory bus is loaded. JEDEC recommends in the termination reference matrix that in the case where both sockets are loaded with DIMMs with 2 ranks each, only the front side of the DIMM on the second slot be ODT enabled. IBIS simulations are the safest way to determine which ODT terminations need to be turned on.

Running the Interface Twice as Fast as the Internal Memory Controller

Feature-rich IC devices can facilitate meeting timing for memory interfaces.

For example, in cases of high-speed interfaces, reducing the operating frequency of the interface by 50 percent can save design time or enable meeting timing on a complex controller state machine. This can be done using dedicated SERDES features in FPGAs, for example. This design technique is very advantageous when a large burst of contiguous locations is accessed. Depending on the SERDES configuration, there is a possibility that one clock cycle of latency is inserted in the interface. Although the size of the bus is multiplied by two, the logic design can leverage the inherent structure of parallelism in the FPGA device and take advantage of a slower switching rate to meet timing more easily and to consume less dynamic power. The state machine of the controller runs slower, allowing the use of a more complex controller logic that can increase overall efficiency and optimize bus utilization. This makes the logic design easier to place and route, and the end result is a more flexible and robust system that is less susceptible to change.

Conclusion

With rising clocking rates and shrinking valid windows, parallel buses for memory interfaces are becoming more challenging for designers. All the stages of the design and implementation should be considered carefully to tune the interface parameters so as to determine the optimal settings. Signal integrity simulations and simultaneous switching output checks are key factors in tailoring the interface. The feature-rich silicon resources in devices on which memory controllers are implemented, such as process, voltage, and temperature compensated delays; dedicated routing and registers; and specific clocking resources can also help in maintaining or improving the memory interface performance targets. 🌈

This article is reprinted with permission from CMP. It originally ran on the Programmable Logic DesignLine (www.pldesignline.com) on January 18, 2006.

How to Detect Potential Memory Problems Early in FPGA Designs

System compatibility testing for FPGA memory requires methods other than traditional signal integrity analysis.

by Larry French
FAE Manager
Micron Semiconductor Products, Inc.
lfrench@micron.com

As a designer, you probably spend a significant amount of time simulating boards and building and testing prototypes. It is critical that the kinds of tests performed on these prototypes are effective in detecting problems that can occur in production or in the field.

DRAM or other memory combined in an FPGA system may require different test methodologies than an FPGA alone. Proper selection of memory design, test, and verification tools reduces engineering time and increases the probability of detecting potential problems. In this article, we'll discuss the best practices for thoroughly debugging a Xilinx® FPGA design that uses memory.

Memory Design, Testing, and Verification Tools

You can use many tools to simulate or debug a design. Table 1 lists the five essential tools for memory design. Note that this is not a complete list as it does not include thermal simulation tools; instead, it focuses only on those tools that you can use to validate the functionality and robustness of a design. Table 2 shows when these tools can be used most effectively.

This article focuses on the five phases of product development, as shown in Table 2:

- **Phase 1** – Design (no hardware, only simulation)
- **Phase 2** – Alpha (or Early) Prototype (design and hardware changes likely to occur before production)
- **Phase 3** – Beta Prototype (nearly “production-ready” system)

- **Phase 4** – Production

- **Phase 5** – Post-Production (in the form of memory upgrades or field replacements)

The Value of SI Testing

SI is not a panacea and should be used judiciously. SI should not be overused, although it frequently is. For very early or alpha prototypes, SI is a key tool for ensuring that your system is free of a number of memory problems, including:

- Ringing and overshoot/undershoot
- Timing violations, such as:
 - Setup and hold time
 - Slew rate (weakly driven or strongly driven signals)
 - Setup/hold time (data, clock, and controls)

Tool	Example
Electrical Simulations	SPICE or IBIS
Behavioral Simulations	Verilog or VHDL
Signal Integrity	Oscilloscope and probes; possibly mixed-mode to allow for more accurate signal capture
Margin Testing	Guardband testing and four-corner testing by variation of voltage and temperature
Compatibility Testing	Functional software testing or system reboot test

Table 1 – Memory design, test, and verification tools

- Clock duty cycle and differential clock crossing (CK/CK#)
- Bus contention

By contrast, SI is not useful in the beta prototype phase unless there are changes to the board signals. (After all, each signal net is validated in the alpha prototype.) However, if a signal does change, you can use SI to ensure that no SI problems exist with the changed net(s). Rarely – if ever – is there a need for SI testing in production.

SI is commonly overused for testing because electrical engineers are comfortable looking at an oscilloscope and using the captures or photographs as documentation to show that a system was tested (Figure 1). Yet extensive experience at Micron Technology shows that much more effective tools exist for catching failures. In fact, our experience shows that SI cannot detect all types of system failures.

Limitations of SI Testing

SI testing has a number of fundamental limitations. First and foremost is the memory industry migration to fine-pitch ball-grid array (FBGA) packages. Without taking up valuable board real estate for probe pins, SI is difficult or impossible because there is no way to probe under the package.

Micron has taken several hundred

Tool	Design	Alpha Proto	Beta Proto	Production	Post-Prod
Simulation – Electrical	Essential	Very Valuable	Limited Value	Rarely Used	No Value
Simulation – Behavioral	Essential	Very Valuable	Limited Value	Rarely Used	No Value
Signal Integrity	Unavailable	Critical	Limited Value	Rarely Used	No Value
Margin Testing	Unavailable	Essential	Essential	Essential	Essential
Compatibility	Unavailable	Valuable	Essential	Essential	Essential

Table 2 – Tools for verifying memory functionality versus design phase

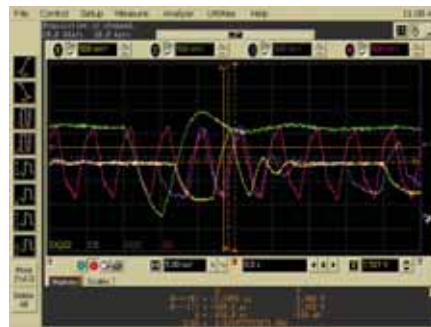


Figure 1 – Typical signal integrity shot from an oscilloscope

thousand scope shots in our SI lab during memory qualification testing. Based on this extensive data, we concluded that system problems are most easily found with margin and compatibility testing. Although SI is useful in the alpha prototype phase, it should be replaced by these other tests during beta prototype and production.

Here are some other results of our SI testing:

- SI did not find a single issue that was not identified by memory or system-level diagnostics. In other words, SI found the same failures as the other tests, thus duplicating the capabilities of margin testing and software testing.

- SI is time-consuming. Probing 64-bit or 72-bit data buses and taking scope shots requires a great deal of time.
- SI uses costly equipment. To gather accurate scope shots, you need high-cost oscilloscopes and probes.
- SI takes up valuable engineering resources. High-level engineering analysis is required to evaluate scope shots.
- SI does not find all errors. Margin and compatibility testing find errors that are not detectable by SI.

The best tests for finding FPGA/memory issues are margin and compatibility testing.

Margin Testing

Margin testing is used to evaluate how systems work under extreme temperatures and voltages. Many system parameters change with temperature/voltage, including slew rate, drive strength, and access time. Validation of a system at room temperature is not enough. Micron found that another benefit of margin testing is that it detects system problems that SI will not.

Four-corner testing is a best industry practice for margin testing. If a failure is

How Does the Logic Analyzer (or Mixed-Mode Analysis) Fit In?

You may have noticed that Table 1 does not include logic analyzers. Although it is rare to find a debug lab that does not include this tool as an integral part of its design and debug process, we will not discuss logic analyzers in this article. Because of the cost and time involved, they are rarely the first tool used to detect a failure or problem in a system. Logic analyzers are, however, invaluable in linking a problem, after it has been identified, to its root cause. Like signal integrity (SI), logic analyzers should be used after a problem has been detected.

...margin and compatibility testing will identify more marginalities or problems within a system than traditional methods such as SI.

going to occur during margin testing, it will likely occur at one of these points:

- Corner #1: high voltage, high temperature
- Corner #2: high voltage, low temperature
- Corner #3: low voltage, high temperature
- Corner #4: low voltage, low temperature

There is one caveat to this rule. During the alpha prototype, margin testing may not be of value because the design is still changing and the margin will be improved in the beta prototype. Once the system is nearly production-ready, you should perform extensive margin testing.

Compatibility Testing

Compatibility testing refers simply to the software tests that are run on a system. These can include BIOS, system operating software, end-user software, embedded software, and test programs. PCs are extremely programmable; therefore, you should run many different types of software tests.

In embedded systems where the FPGA acts like a processor, compatibility testing can also comprise a large number of tests. In other embedded applications where the DRAM has a dedicated purpose such as a FIFO or buffer, software testing by definition is limited to the final application. Thorough compatibility testing (along with margin testing) is one of the best ways to detect system-level issues or failures in all of these types of systems.

Given the programmable nature of Xilinx FPGAs, you might even consider a special FPGA memory test program. This program would only be used to run numerous test vectors (checkerboard, inversions) to and from the memory to validate the DRAM interface. It could eas-

ily be written to identify a bit error, address, or row – in contrast to the standard embedded program that might not identify any memory failures. This program could be run during margin testing. It would be especially interesting for embedded applications where the memory interface runs a very limited set of operations. Likely, this type of test would have more value than extensive SI testing of the final product.

Tests Not To Ignore

The following tests, if ignored, can lead to production and field problems that are subtle, hard to detect, and intermittent.

Power-Up Cycling

A good memory test plan should include several tests that are sometimes skipped and can lead to production or field problems. The first of these is power-up cycling. During power-up, a number of unique events occur, including the ramp-up of voltages and the JEDEC-standard DRAM initialization sequence. Best industry practices for testing PCs include power-up cycling tests to ensure that you catch intermittent power-up issues.

Two types of power-up cycling exist: cold- and warm-boot cycling. A cold boot occurs when a system has not been running and is at room temperature. A warm boot occurs after a system has been running for awhile and the internal temperature is stabilized. You should consider both tests to identify temperature-dependent problems.

Self-Refresh Testing

DRAM cells leak charge and must be refreshed often to ensure proper operation. Self-refresh is a key way to save system power when the memory is not used for long periods of time. It is critical that the memory controller provide the prop-

er in-spec commands when entering and exiting self-refresh; otherwise, you could lose data.

Like power-up cycling, self-refresh cycling is a useful compatibility test. If an intermittent self-refresh enter or exit problem is present, repeated cycling can help detect it. Applications that do not use self-refresh should completely skip this test.


Sustaining Qualifications

One last area to consider is the test methodology for sustaining qualifications. That is, what tests should you perform to qualify a memory device once a system is in production? This type of testing is frequently performed to ensure that an adequate supply of components will be available for uninterrupted production.

During production a system is stable and unchanging. Our experience has shown that margin and compatibility testing are the key tests for sustaining qualifications. Because a system is stable, SI has little or no value.

Conclusion

In this article, our intent has been to encourage designers to rethink the way they test and validate FPGA and memory interfaces. Using smart test practices can result in an immediate reduction in engineering hours during memory qualifications. In addition, proper use of margin and compatibility testing will identify more marginalities or problems within a system than traditional methods such as SI. No “one-size-fits-all” test methodology exists, so you should identify the test methodology that is most effective for your designs.

For more detailed information on testing memory, see Micron’s latest DesignLine article, “Understanding the Value of Signal Integrity,” on our website, www.micron.com. 

Interfacing QDR II SRAM with Virtex-4 FPGAs

QDR II SRAM devices provide a suitable solution for memory requirements when partnered with Virtex-4 FPGAs.

by Veena Kondapalli
Applications Engineer Staff
Cypress Semiconductor Corp.
vee@cypress.com

The growing demand for higher performance communications, networking, and DSP necessitates higher performance memory devices to support such applications. Memory manufacturers like Cypress have developed specialized memory products such as quad data rate II (QDR II) SRAM devices to optimize memory bandwidth for a specific system architecture. In this article, I'll provide a general outline of a QDR II SRAM interface implemented in a Xilinx® Virtex™-4 XC4VP25 FF6688-11 device.

Figure 1 shows a block diagram of the QDR II SRAM design interface, with the physical interface to the actual memory device on the controller.

QDR II SRAM

QDR II can perform two data write and two data reads per clock cycle. It uses one port for writing data and one port for reading data. These unidirectional ports support simultaneous reads and writes and allow back-to-back transactions without the bus contention that may occur with a single bidirectional data bus.

Clocking Scheme

The FPGA generates all of the clock and control signals for reads and writes to memory. The memory clocks are typically generated using a double-data-rate (DDR) register. A digital clock manager (DCM) generates the clock and its inverted version. This has two advantages. First, the data, control, and clock signals all go through similar delay elements while exiting the FPGA. Second, the clock-duty cycle distortion is minimal when global clock nets are used for the clock and the 180° phase-shifted clock.

The reference design uses the phase-shifted outputs of the DCM to clock the interface on the transmit side. This configuration gives the best jitter and skew characteristics.

QDR II devices include the following features:

- Maximum frequency of operations - 250 MHz - tested up to 278 MHz
- Available in QDR II architecture with burst of 2 or 4
- Supports simultaneous reads/writes and back-to-back transactions without bus contention issues
- Supports multiple QDR II SRAM devices on the same bus to:
 - Increase the density of the memory resource

– Divide the speed of the interface by using multiple devices to achieve a given bandwidth

- Read: valid window worst-case 440 ps
- Write: valid window worst-case 460 ps
- Address and control signal timing analysis: command window worst-case 2360 ps

Conclusion

For more information about QDR II and Virtex-4 devices, see Xilinx application note XAPP703, "QDR II SRAM Interface for Virtex-4 Devices," at www.xilinx.com/bvdocs/appnotes/xapp703.pdf, as well as Cypress application note "Interfacing QDR-II SRAM with Virtex-4 Devices" at www.cypress.com.

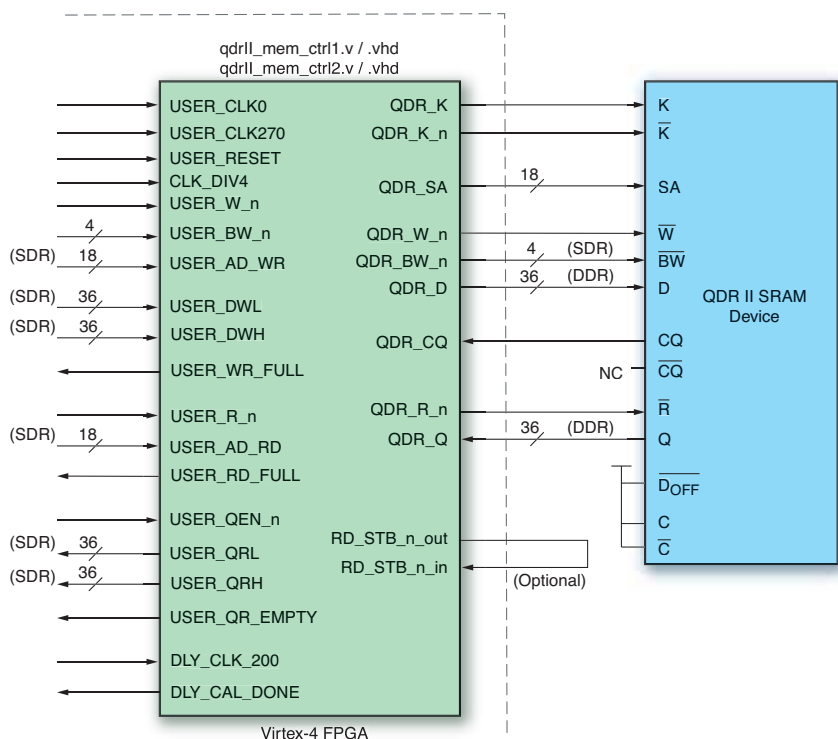


Figure 1 - Top-level architecture block diagram

SIGNAL INTEGRITY

Xilinx Low-Noise FPGAs Meet SI Challenges

Unique chip package and I/Os accelerate system development

The good news is plentiful. Advances in silicon technology are enabling higher system performance to satisfy the requirements of networking, wireless, video, and other demanding applications. At the same time, an abundance of I/O pins with faster data edge-rates enables higher interface speeds.

However, every advance creates new design challenges. Wide buses with hundreds of simultaneously switching outputs (SSOs) create crosstalk. The undesirable effects of this electrical noise include jitter that threatens the stability of high-bandwidth interfaces. Sharp edge-rates further exacerbate noise problems.

“High signal integrity demands a low-noise chip.”



Howard Johnson
The world's foremost authority on signal integrity

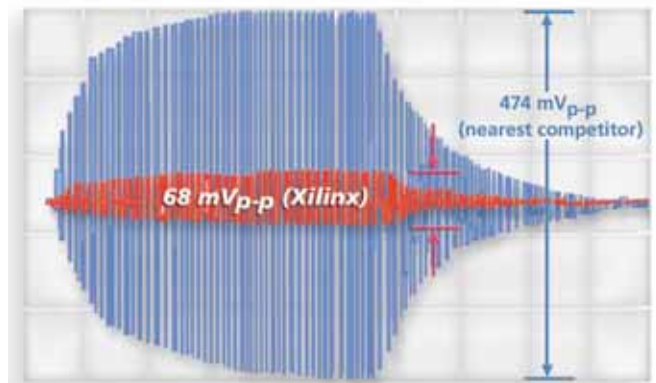
Essential noise control

To address these issues, careful printed circuit-board (PCB) design and layout are critical for controlling system-level noise and crosstalk. Another important consideration is the electrical characteristics of the components mounted on the PCB. With its Virtex™-4 FPGAs, Xilinx, Inc., uses special chip I/O and packaging technologies that significantly improve signal-integrity not only at the chip level, but also at the system-level.

"Xilinx preempts signal-integrity issues at the chip level," says Xilinx Senior Director of Systems and Applications Engineering Andy DeBaets. "This reduces board development and debug effort, and may even make the difference between a successful and scrapped board design."

In high-speed systems, a significant source of crosstalk is inductive coupling within the PCB via field under the BGA package. In systems with sub-optimal design, noise from simultaneously switching outputs can reach levels that severely degrade performance. In extreme cases, this noise can even lead to system failure. A properly designed BGA package minimizes inductive coupling between I/Os by placing a power/ground pin pair next to every signal pin.

"Xilinx's innovative SparseChevron™ package design minimizes crosstalk problems that can degrade system performance. This is particularly important for wide, high-speed DDR2 SDRAM or QDR II SRAM memory designs," DeBaets says.



Design Example: 1.5 volt LVCMOS 4mA, I/O, 100 aggressors shown

Seven times less crosstalk

Analysis by independent signal-integrity expert Dr. Howard Johnson verifies the ability of SparseChevron technology to control SSO noise/crosstalk at the chip level. "High signal integrity demands a low-noise chip" states Johnson. "Compared to competing 90nm FPGAs, the Virtex-4 FPGAs in SparseChevron packaging demonstrate seven times less SSO noise," Johnson stresses.

Xilinx Virtex-4 devices include several other features to help designers improve system-level signal integrity prior to board layout. Programmable edge rates and drive strength minimize noise while meeting other design objectives. Xilinx Digitally Controlled Impedance (DCI) technology enables designers to implement on-chip line termination for single-ended and differential I/Os. By eliminating the need for external termination resistors, DCI technology enables designers to minimize system component count and significantly simplify board layout and manufacturing.

To learn more about how Virtex-4 FPGAs can help control noise in your system, visit www.xilinx.com/virtex4.

For more information on signal integrity, go to
www.xilinx.com/platformsi



Memory Interfaces Reference Designs

Give your designs the Virtex-4 FPGA advantage.

by Adrian Cosoroaba
Marketing Manager
Xilinx, Inc.
adrian.cosoroaba@xilinx.com

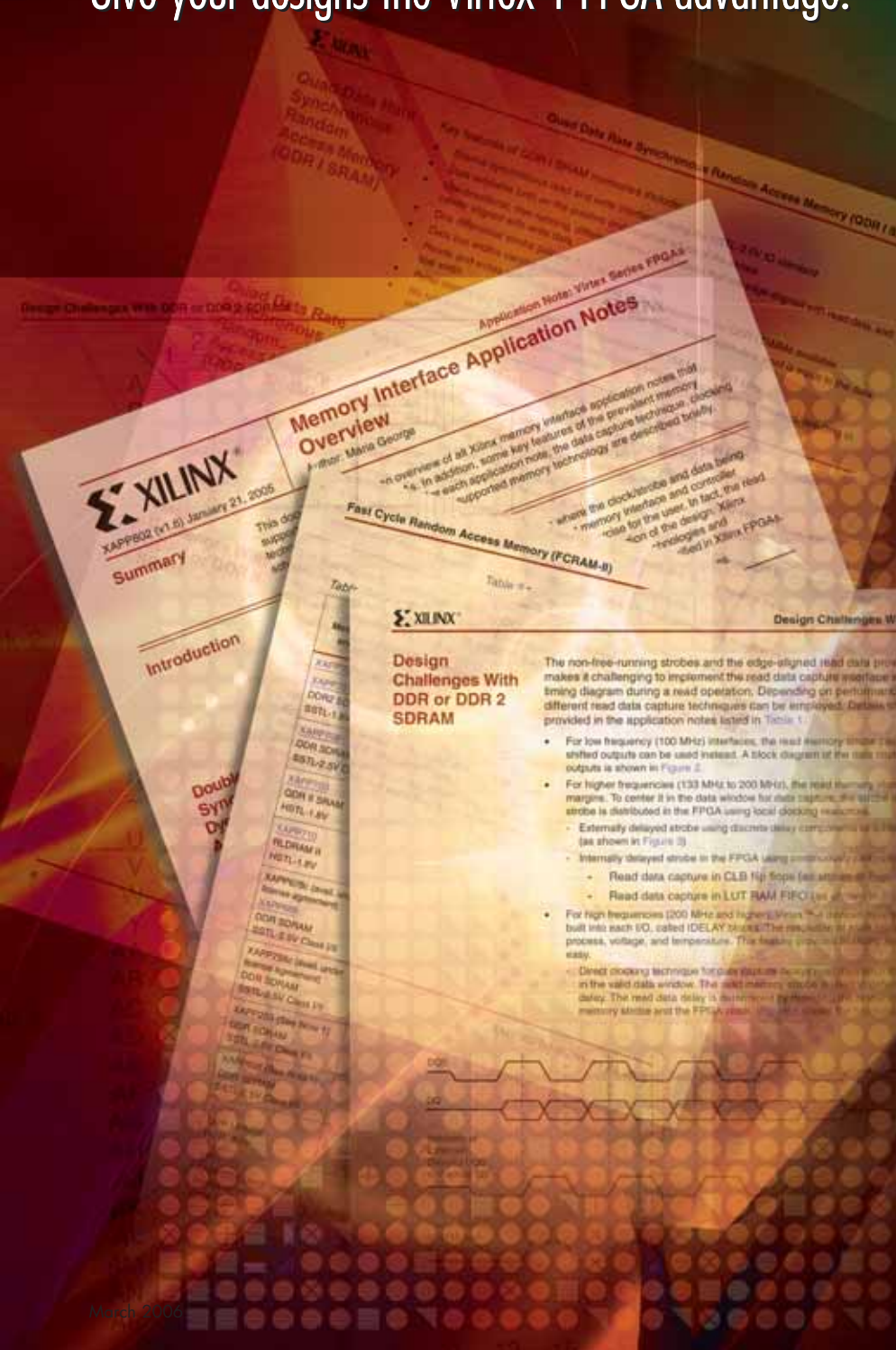
Memory interfaces are source-synchronous interfaces in which the clock/strobe and data being transmitted from a memory device are edge-aligned. Most memory interface and controller vendors leave the read data capture implementation as an exercise for the user. In fact, the read data capture implementation in FPGAs is the most challenging portion of the design. Xilinx provides multiple read data capture techniques for different memory technologies and performance requirements. All of these techniques are implemented and verified in Xilinx® FPGAs.

The following sections provide a brief overview of prevalent memory technologies.

Double Data Rate Synchronous Dynamic Random Access Memory (DDR SDRAM)

Key features of DDR SDRAM memories include:

- Source-synchronous read and write interfaces using the SSTL-2.5V Class I/II I/O standard
- Data available both on the positive and negative edges of the strobe
- Bi-directional, non-free-running, single-ended strobes that are output edge-aligned with read data and must be input center-aligned with write data
- One strobe per 4 or 8 data bits
- Data bus widths varying between 8, 16, and 32 for components and 32, 64, and 72 for DIMMs
- Supports reads and writes with burst lengths of two, four, or eight data words, where each data word is equal to the data bus width
- Read latency of 2, 2.5, or 3 clock cycles, with frequencies of 100 MHz, 133 MHz, 166 MHz, and 200 MHz
- Row activation required before accessing column addresses in an inactive row
- Refresh cycles required every 15.6 μ s
- Initialization sequence required after power on and before normal operation



Double Data Rate Synchronous Dynamic Random Access Memory (DDR 2 SDRAM)

Key features of DDR 2 SDRAM memories, the second-generation DDR SDRAMs, include:

- Source-synchronous read and write interfaces using the SSTL-1.8V Class I/II I/O standard
- Data available both on the positive and negative edges of the strobe
- Bi-directional, non-free-running, differential strobes that are output edge-aligned with read data and must be input center-aligned with write data
- One differential strobe pair per 4 or 8 data bits
- Data bus widths varying between 4, 8, and 16 for components and 64 and 72 for DIMMs
- Supports reads and writes with burst lengths of four or eight data words, where each data word is equal to the data bus width
- Read latency is a minimum of three clock cycles, with frequencies ranging from 200 MHz to 400 MHz
- Row activation required before accessing column addresses in an inactive row
- Refresh cycles required every 7.8 μ s
- Initialization sequence required after power on and before normal operation

Quad Data Rate Synchronous Random Access Memory (QDR II SRAM)

Key features of QDR II SRAM memories, the second-generation QDR I SRAMs, include:

- Source-synchronous read and write interfaces using the HSTL-1.8V I/O standard
- Data available both on the positive and negative edges of the strobe
- Uni-directional, free-running, differential data/echo clocks that are edge-aligned with read data and center-aligned with write data
- One differential strobe pair per 8, 9, 18, 36, or 72 data bits
- Data bus widths varying between 8, 9, 18, 36, and 72 for components (no QDR II SDRAM DIMMs available)

Memory Technology and I/O Standard	Supported FPGAs	Maximum Performance	Maximum Data Width	XAPP Number	XAPP Title	Data Capture Scheme
DDR 2 SDRAM SSTL-1.8V Class II	Virtex-4	333 MHz	8 bits (Components)	XAPP721 XAPP723	High Performance DDR 2 SDRAM Interface Data Capture Using ISERDES and OSERDES DDR2 Controller (267 MHz and Above) Using Virtex-4 Devices	Read data is captured in the delayed DQS domain and transferred to the FPGA clock domain within the ISERDES.
DDR 2 SDRAM SSTL-1.8V Class II	Virtex-4	267 MHz	16 bits (Components) 144-bit Registered DIMM	XAPP702 XAPP701	DDR 2 SDRAM Controller Using Virtex-4 Devices Memory Interfaces Data Capture Using Direct Clocking Technique	Read data delayed such that FPGA clock is centered in data window. Memory read strobe used to determine amount of read data delay.
DDR SDRAM SSTL-2.5V Class I/II	Virtex-4	200 MHz	16 bits (Components) 144-bit Registered DIMM	XAPP709	DDR SDRAM Controller Using Virtex-4 Devices	Read data delayed such that FPGA clock is centered in data window. Memory read strobe used to determine amount of read data delay.
QDR II SRAM HSTL-1.8V	Virtex-4	300 MHz	72 bits (Components)	XAPP703	QDR II SRAM Interface	Read data delayed such that FPGA clock is centered in data window. Memory read strobe used to determine amount of read data delay.
RLDRAM II HSTL-1.8V	Virtex-4	300 MHz	36 bits (Components)	XAPP710	Synthesizable CIO DDR RLDRAM II Controller for Virtex-4 FPGAs	Read data delayed such that FPGA clock is centered in data window. Memory read strobe used to determine amount of read data delay.

Table 1 – Virtex-4 memory interface application notes (XAPPs) currently available, with a brief description of the read data capture technique

XAPP Number Memory Technology and I/O Standard	Performance	Number of DCMs/DLLs	Number of BUFs	Number of Interfaces with Listed DCMs and BUFs	Device(s) Used for Hardware Verification	Requirements
XAPP721 XAPP723 DDR2 SDRAM SSTL-1.8V Class II	333 MHz	1 DCM 2 PMCDs	6	Multiple at Same Frequency	XC4VLX25 -11 FF668	All Banks Supported
XAPP702 XAPP701 DDR2 SDRAM SSTL-1.8V Class II	267 MHz	1	6	Multiple at Same Frequency	XC4VLX25 -11 FF668	All Banks Supported
XAPP709 DDR SDRAM SSTL-2.5V Class I/II	200 MHz	1	6	Multiple at Same Frequency	XC4VLX25 -11 FF668	All Banks Supported
XAPP703 QDR II SRAM HSTL-1.8V	300 MHz	1	3	Multiple at Same Frequency	XC4VLX25 -11 FF668	All Banks Supported
XAPP710 RLDRAM II HSTL-1.8V	300 MHz	1	5	Multiple at Same Frequency	XC4VLX25 -11 FF668	All Banks Supported

Table 2 – Resource utilization for all Virtex-4 memory interface application notes currently available


- Reads and writes with burst lengths of two or four data words, where each data word is equal to the data bus width
- Read latency is 1.5 clock cycles, with frequencies from 154 MHz to 300 MHz
- No row activation, refresh cycles, or initialization sequence after power on required, resulting in more efficient memory bandwidth utilization

Reduced Latency Dynamic Random Access Memory (RLDRAM II)

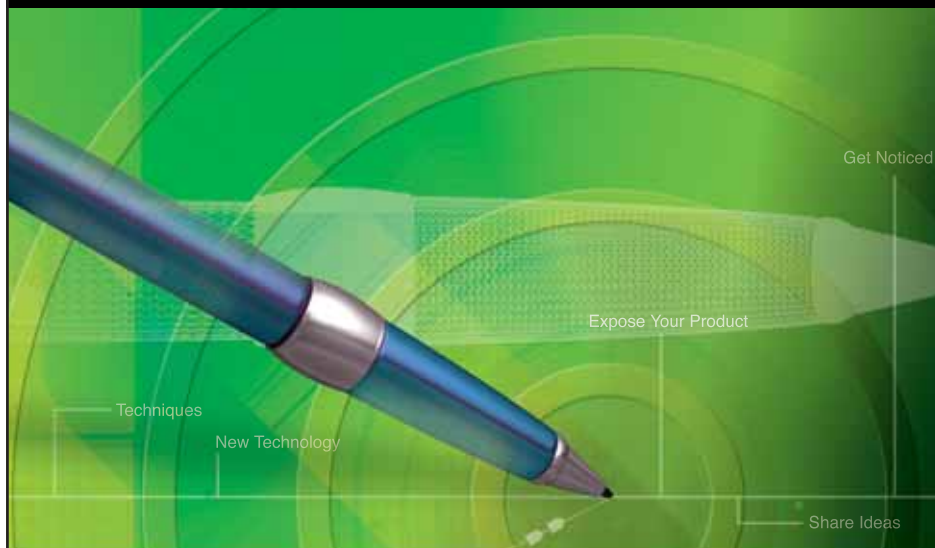
Key features of RLDRAM II memories include:

- Source-synchronous read and write interfaces using the HSTL-1.8V I/O standard
- Data available both on the positive and negative edges of the strobe
- Uni-directional, free-running, differential memory clocks that are edge-aligned with read data and center-aligned with write data
- One strobe per 9 or 18 data bits
- Data bus widths varying between 9, 18, and 36 for components and no DIMMs
- Supports reads and writes with burst lengths of two, four, or eight data words, where each data word is equal to the data bus width
- Read latency of five or six clock cycles, with frequencies of 200 MHz, 300 MHz, and 400 MHz
- Data-valid signal provided by memory device
- No row activation required; row and column can be addressed together
- Refresh cycles required every 3.9 μ s
- Initialization sequence required after power on and before normal operation

Conclusion

For application notes on various memory technologies and performance requirements, visit www.xilinx.com/memory. The summaries in Table 1 and Table 2 can help you determine which application note is relevant for a particular design. 

GET PUBLISHED



WOULD YOU LIKE TO WRITE FOR XCELL PUBLICATIONS?

It's easier than you think!

Submit an article draft for our Web-based or printed publications and we will assign an editor and a graphic artist to work with you to make your work look as good as possible.

For more information on this exciting and highly rewarding program, please contact:

Forrest Couch
Publisher, Xcell Publications
xcell@xilinx.com



See all the new publications on our website.

www.xilinx.com/xcell



XAPP454 (v1.0) December 6, 2004

DDR2 SDRAM Memory Interface for Spartan-3 FPGAs

Author: Karthikeyan Palanisamy

Summary

This application note describes a DDR2 SDRAM memory interface implementation in a Spartan-3 device, interfacing with a Micron DDR2 SDRAM device. This document provides a brief overview of the DDR2 SDRAM device features, followed by a detailed explanation of the DDR2 SDRAM memory interface implementation.

DDR2 SDRAM Device Overview

DDR2 SDRAM devices are the next generation DDR SDRAM devices. The DDR2 SDRAM memory interface is source-synchronous and supports double-data rate like DDR SDRAM memory. DDR2 SDRAM devices use the SSTL 1.8V I/O standard.

DDR2 SDRAM devices use a DDR SDRAM architecture to achieve high-speed operation. The memory operates using a differential clock provided by the controller. (The reference design on the web does not support differential strobes. Support for this is planned to be added later.) Commands are registered at every positive edge of the clock. A bi-directional data strobe (DQS) is transmitted along with the data for use in data capture at the receiver. DQS is a strobe transmitted by the DDR2 SDRAM device during reads, and by the controller during writes. DQS is edge-aligned with data for reads, and center-aligned with data for writes.

Read and write accesses to the DDR2 SDRAM device are burst oriented. Accesses begin with the registration of an active command and are then followed by a read or a 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 with the read or write command are used to select the bank and starting column location for the burst access.

Interface Model

The DDR2 SDRAM memory interface is layered to simplify the design and make the design modular. [Figure 1](#) shows the layered memory interface. The three layers consist of an application layer, an implementation layer, and a physical layer.

© 2004 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.

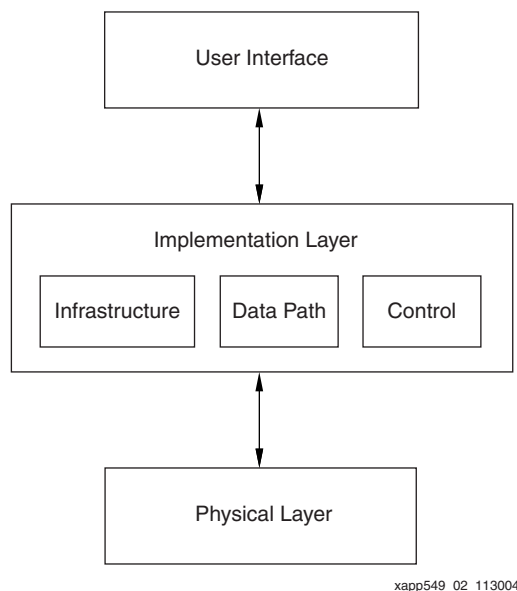


Figure 1: Interface Layering Model

DDR2 SDRAM Controller Modules

Figure 2 is a block diagram of the Spartan-3 DDR2 SDRAM memory interface. All four blocks shown in this figure are sub-blocks of the ddr2_top module. The function of each block is explained in the following sections.

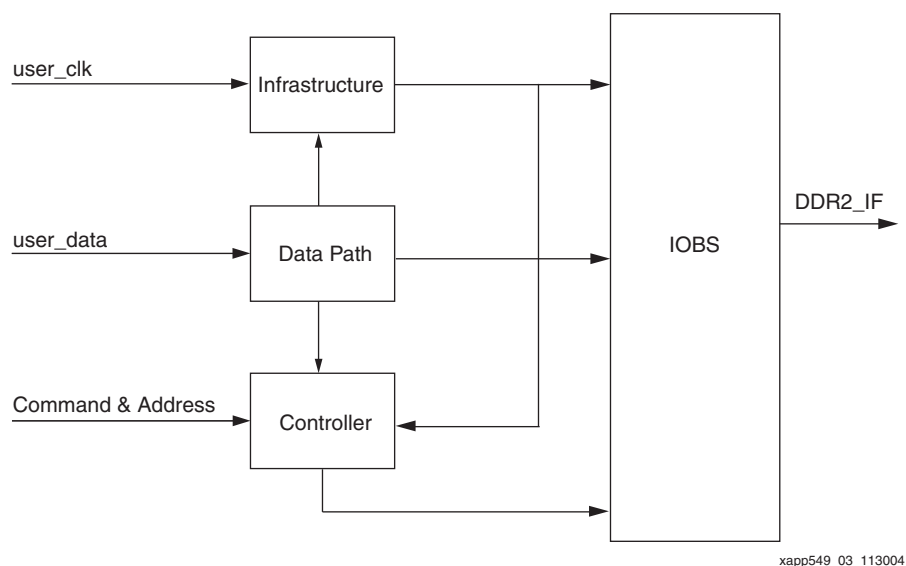


Figure 2: DDR2 SDRAM Memory Interface Modules

Controller

The controller's design is based on the design shown in [XAPP253, Synthesizable 400 Mb/s DDR SDRAM Controller](#), but is modified to incorporate changes for the DDR2 SDRAM memory. It supports a burst length of four, and CAS latencies of three and four. The design is modified to implement the write latency feature of the DDR2 SDRAM memory. The controller initializes the EMR(2) and EMR(3) registers during the Load Mode command and also generates differential data strobes.

The controller accepts user commands, decodes these user commands, and generates read, write, and refresh commands to the DDR2 SDRAM memory. The controller also generates signals for other modules. Refer to XAPP253 for detailed design and timing analyses of the controller module.

Data Path

The data path module is responsible for transmitting data to and receiving data from the memories. Major functions include:

- Writing data to the memory
- Reading data from the memory
- Transferring the read data from the memory clock domain to the FPGA clock domain

For a description of data write and data read capture techniques, see XAPP768c, *Interfacing Spartan-3 Devices With 166 MHz or 333 Mb/s DDR SDRAM Memories*. The write data and strobe are clocked out of the FPGA. The strobe is center-aligned with respect to the data during writes. For DDR2 SDRAM memories, the strobe is non-free running. To meet the requirements specified above, the write data is clocked out using a clock that is shifted 90° and 270° from the primary clock going to the memory. The data strobes are generated out of primary clocks going to the memory.

Memory read data is edge-aligned with a source-synchronous clock. The DDR2 SDRAM clock is a non-free running strobe. The data is received using the non-free running strobe and transferred to the FPGA clock domain. The input side of the data uses resources similar to the input side of the strobe. This ensures matched delays on data and strobe signals until the strobe is delayed in the strobe delay circuit.

Infrastructure

The Infrastructure module generates the FPGA clocks and reset signals. A Digital Clock Manager (DCM) is used to generate the clock and its inverted version. A delay calibration circuit is also implemented in this module.

The delay calibration circuit is used to select the number of delay elements used to delay the strobe lines 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. The calibration circuit selects the number of delay elements for any given time. After the calibration is done, it asserts the select lines for the delay circuit. Refer to XAPP768c for details about delay calibration.

IOBS

All FPGA input and output signals are implemented in the IOBS module. All address and control signals are registered going into and coming out from the IOBS module.

User Interface Signals

Table 1 shows user interface signal descriptions; all signal directions are with respect to the DDR2 SDRAM controller.

Table 1: User Interface Signals

Signal Name	Direction	Description
dip1	Input	Clock enable signal for DDR2 SDRAM (active low)
rst_dqs_div_in	Input	This signal enables the dqs_div flop during DDR2 SDRAM memory read.
reset_in	Input	System reset
user_input_data[(2n-1):0]	Input	Write Data for DDR2 SDRAM, where 'n' is the width of the memory interface
user_input_address[addwidth:0]	Input	DDR2 SDRAM row and column address
user_bank_address[bankaddwidth:0]	Input	DDR2 SDRAM bank address
user_config_reg1[14:0]	Input	DDR2 SDRAM configuration data register1
user_config_reg2[12:0]	Input	DDR2 SDRAM configuration data register2
user_command_reg[3:0]	Input	User command register for DDR2 SDRAM controller
burst_done	Input	Burst data transfer done signal
rst_dqs_div_out	Output	This signal is externally connected to rst_dqs_div_in. This signal enables the dqs_div flop.
user_output_data[(2n-1):0]	Output	Read data from DDR2 SDRAM
user_data_valid	Output	This active Low signal indicates that read data from DDR2 SDRAM memory is valid.
user_cmd_ack	Output	Acknowledge signal for user_command
user_ODT_ack	Output	Acknowledge signal for ODT command
init_val	Output	Indicates DDR2 SDRAM is initialized
ar_done	Output	Indicates auto-refresh command is given to DDR2 SDRAM
clk_int	Input	Clock generated by DDR2 SDRAM controller
clk90_int	Input	90 degrees phase-shifted clock generated by DDR2 SDRAM controller
sys_rst	Input	This is generated with system reset input
sys_rst90	Input	90 degrees phase-shifted Reset generated with system reset input
sys_rst180	Input	180 degrees phase-shifted Reset generated with system reset input.
sys_rst270	Input	270 degrees phase-shifted Reset generated with system reset input.

Notes:

1. All signal directions are with respect to DDR2 SDRAM controller.

Signal Descriptions

user_input_data[(2n-1):0]

This is the write data to DDR2 SDRAM from the user interface. The data is valid on a DDR2 SDRAM write command, where n is the width of the DDR2 SDRAM memory. The DDR2 SDRAM controller converts single data rate to double data rate on the physical layer side.

user_input_address[addwidth:0]

This is the sum of row and column address for DDR2 SDRAM writes and reads. Depending on address width variable selection, user_input_address is divided into row and column address bits.

user_bank_address[bankaddwidth:0]

Bank address for DDR2 SDRAM. There is a variable through which the bank address is selectable.

user_config_reg1[14:0]

Configuration data for DDR2 SDRAM memory initialization. The contents of this register are loaded into the mode register during a Load Mode command. The format for user_config_reg1 is as follows:

14	13	11	10 9	7 6	4 3	2	0
PD	WR	TM	Res	Cas_latency	BT	Burst_length	

Burst_length[2:0]

The controller supports only a burst length of four.

BT

This bit selects the burst type. The controller supports only sequential bursts. This bit is always set to zero in the controller.

Cas_latency [6:4]

Bits 6:4 select the cas latency. The DDR2 SDRAM controller supports a cas latency of 3 and 4.

Res [9:7]

Bits 9:7 are reserved for future implementation.

TM

This bit is loaded into the TM bit of the Load Mode Register.

WR [13:11]

These three bits are written to WR (write recovery) bits of the Load Mode register.

PD

This bit is written to PD (Power Down Mode) bit of the Load Mode register.

Refer to the Micron DDR2 SDRAM data sheets for details on the Load Mode register.

user_config_reg2[12:0]

DDR2 SDRAM configuration data for the Extended Mode Register. The format of user_config_reg2 is as follows.

12	11	10	9	7 6	4 3	2	1	0
OUT	RDQS	DQS	OCD	Posted CAS	RTT	ODS	Res	

Refer to the Micron DDR2 SDRAM data sheets for details on the Extended Mode register.

user_command_reg[3:0]

This is the user command register. Various commands are passed to the DDR2 SDRAM module through this register. [Table 2](#) illustrates the various supported commands.

Table 2: User Commands

user_command_reg[3:0]	User Command Description
0000	NOP
0010	Memory (DDR2 SDRAM) initialization
0011	Auto-refresh
0100	Write
0101	Load Mode (Only Load mode)
0110	Read
Others	Reserved

burst_done

Users should enable this signal, for two clock periods, at the end of the data transfer. The DDR2 SDRAM controller supports write burst or read burst for a single row. Users must terminate on a column boundary and reinitialize on a column boundary for the next row of transactions. The controller terminates a write burst or read burst by issuing a pre-charge command to DDR2 SDRAM memory.

user_output_data[(2n-1):0]

This is the read data from DDR2 SDRAM memory. The DDR2 SDRAM controller converts DDR SDRAM data from DDR2 SDRAM memory to SDR data. As the DDR SDRAM data is converted to SDR data, the width of this bus is 2n, where n is data width of DDR2 SDRAM memory.

user_data_valid

The user_output_data[(2n-1):0] signal is valid on assertion of this signal.

user_cmd_ack

This is the acknowledgement signal for a user read or write command. It is asserted by the DDR2 SDRAM controller during a read or write to DDR2 SDRAM. No new command should be given to the controller until this signal is deasserted.

init_val

The DDR2 SDRAM controller asserts this signal after completing DDR2 SDRAM initialization.

ar_done

The DDR2 SDRAM controller asserts this signal for one clock cycle after the auto-refresh command is given to DDR2 SDRAM.

Note: The output clock and reset signals can be used for data synchronization.

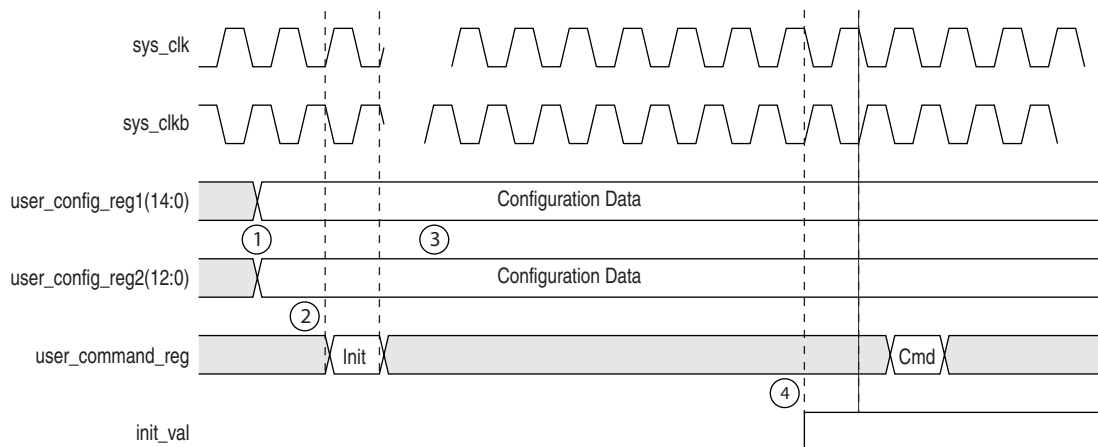
[Table 3](#) shows memory interface signals.

Table 3: Memory Interface Signals

Signal Name	Direction	Description
ddr_dq[(datawidth-1):0]	Inout	Bidirectional DDR2 SDRAM memory data
ddr_dqs[(dqswidth-1):0]	Inout	Bidirectional DDR2 SDRAM memory data strobe signals. The number of strobe signals depends on the data width and strobe to data ratio.
ddr_cke	Output	Clock enable signal for DDR2 SDRAM memory
ddr_csb	Output	Active low memory chip select signal
ddr_rasb	Output	Active low memory row address strobe
ddr_casb	Output	Active low memory column address strobe
ddr_web	Output	Active low memory write enable signal
ddr_dm	Output	Memory data mask signal
ddr_ba	Output	Memory bank address
ddr_address	Output	Memory address (both row and column address)
ddr2_clk*	Output	Memory differential clock signals
ddr_odt[4:0]	Output	Memory on-die termination signals.

Initializing DDR2 SDRAM Memory

Before issuing the memory read and write commands, the DDR2 SDRAM memory must be initialized using the memory initialization command. The data written in the Mode Register and in the Extended Mode Register should be placed on user_config_reg1 [14:0] and user_config_reg2 [12:0] until DDR2 SDRAM initialization is completed. Once the DDR2 SDRAM is initialized, the init_val signal is asserted by the DDR2 SDRAM controller. [Figure 3](#) shows a timing diagram of the memory initialization command.



xapp549_09_120804

Figure 3: DDR2 SDRAM Memory Initialization

1. Two clocks prior to placing the initialization command on command_reg [2:0], the user places valid configuration data on user_config_reg1[14:0] and user_config_reg2[12:0].
2. The user places the initialization command on command_reg [2:0]. This starts the initialization sequence.
3. Data on user_config_reg1[14:0] and user_config_reg2[12:0] should not be changed for any subsequent memory operations.
4. The controller indicates that the configuration is complete by asserting the init_val signal.

DDR2 SDRAM Memory Write

Figure 4 shows a DDR2 SDRAM memory write timing diagram for a burst length of four. The waveform shows two successive bursts. Memory write is preceded by a write command to the DDR2 SDRAM controller. In response to the write command, the DDR2 SDRAM controller acknowledges with a `user_cmd_ack` signal on the rising edge of `SYS_CLKb`. Users should wait for a user command acknowledged signal before proceeding to the next step.

Two and a half clock cycles after the `user_cmd_ack` signal assertion, the memory burst address is placed on `user_input_address[addwidth:0]` lines. The `user_input_address` should be asserted on the rising edge of `SYS_CLK`. The data to be written into memory should be asserted with `clk90_int_val` and should be given to the controller before placing the memory address on `user_input_address`. The user data width is twice that of the memory data width. The controller converts it into double data rate before it is passed to memory.

For a burst length of four, two `user_input_data[(2n-1):0]` pieces of data are given to the DDR2 SDRAM controller with each user address. To terminate the write burst, `burst_done` is asserted on the rising edge of `SYS_CLK` for two clocks. The `burst_done` signal should be asserted for two clocks with the last memory address. Any further commands to the DDR2 SDRAM controller should be given only after the `user_cmd_ack` signal is deasserted.

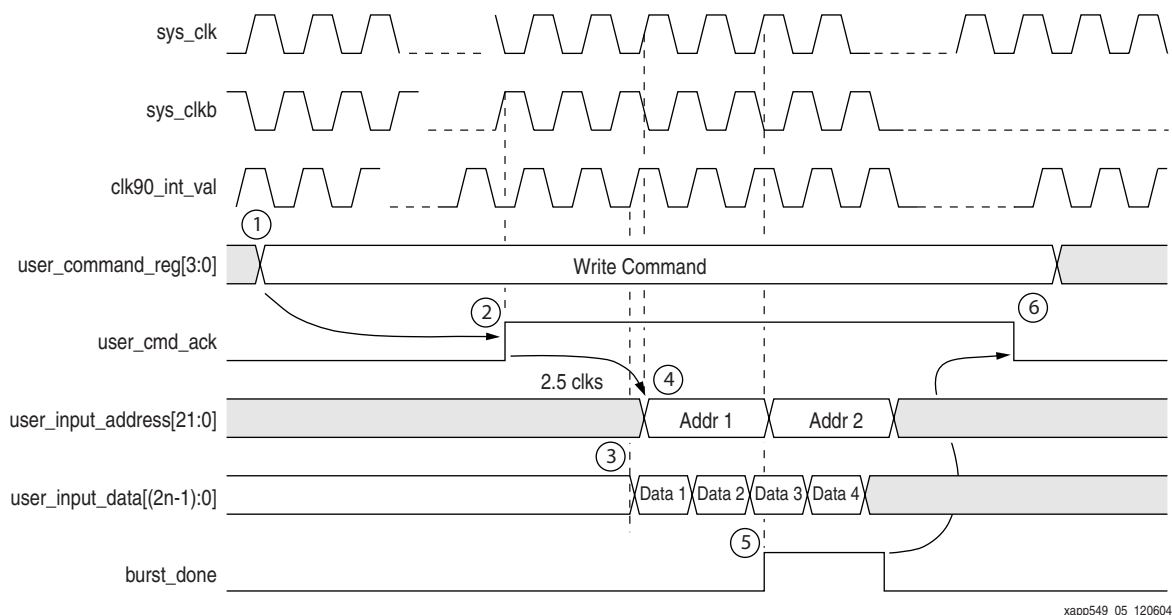


Figure 4: DDR2 SDRAM Memory Write Burst Length of Four

1. The user initiates a memory write by issuing a write command to the DDR2 SDRAM controller. The write command must be asserted on the rising edge of the `SYS_CLK`.
2. The DDR2 SDRAM controller acknowledges the write command by asserting the `user_cmd_ack` signal on the rising edge of the `SYS_CLKb`.
3. The user should place the data to be written into the memory onto the `user_input_data` pins before placing the memory address on the `user_input_address`. The input data is asserted with the `clk90_int_val` signal.
4. Two and half clocks after the `user_cmd_ack` signal assertion, the user should place the memory address on `user_input` address [21:0]. The `user_input_address` signal should be asserted on the rising edge of the `SYS_CLK`.
5. To terminate write burst, the user should assert the `burst_done` signal for two clocks with the last `user_input_address`.
6. Any further commands to the DDR2 SDRAM controller should be given only after the `user_cmd_ack` signal is de-asserted.

DDR2 SDRAM Memory Read

Figure 5 shows a memory read timing diagram for two successive bursts with a burst length of four. The user initiates a memory read by sending a read command to the DDR2 SDRAM controller.

The read command flow is similar to the write command. A read command is asserted on the rising edge of SYS_CLK. The DDR2 SDRAM controller asserts the user_cmd_ack signal in response to the read command on the rising edge of SYS_CLKb. After two and half clock cycles of user_cmd_ack, the memory burst read address is placed on user_input_address[addwidth:0]. The user_input_address signal is asserted on the rising edge of SYS_CLK.

The data read from the DDR2 SDRAM memory is available on user_output_data, which is asserted with clk90_int_val. The data on user_output_data is valid only when user_data_valid signal is asserted. As the DDR SDRAM data is converted to SDR data, the width of this bus is 2n, where n is the data width of the DDR2 SDRAM memory. For a read burst length of four, the DDR2 SDRAM controller outputs only two data with each user address, each of 2n width of DDR2 SDRAM memory. To terminate the read burst, a burst_done signal is asserted for two clock cycles on the rising edge of SYS_CLK. The burst_done signal should be asserted after the last memory address. Any further commands to the DDR2 SDRAM controller should be given after user_cmd_ack signal deassertion.

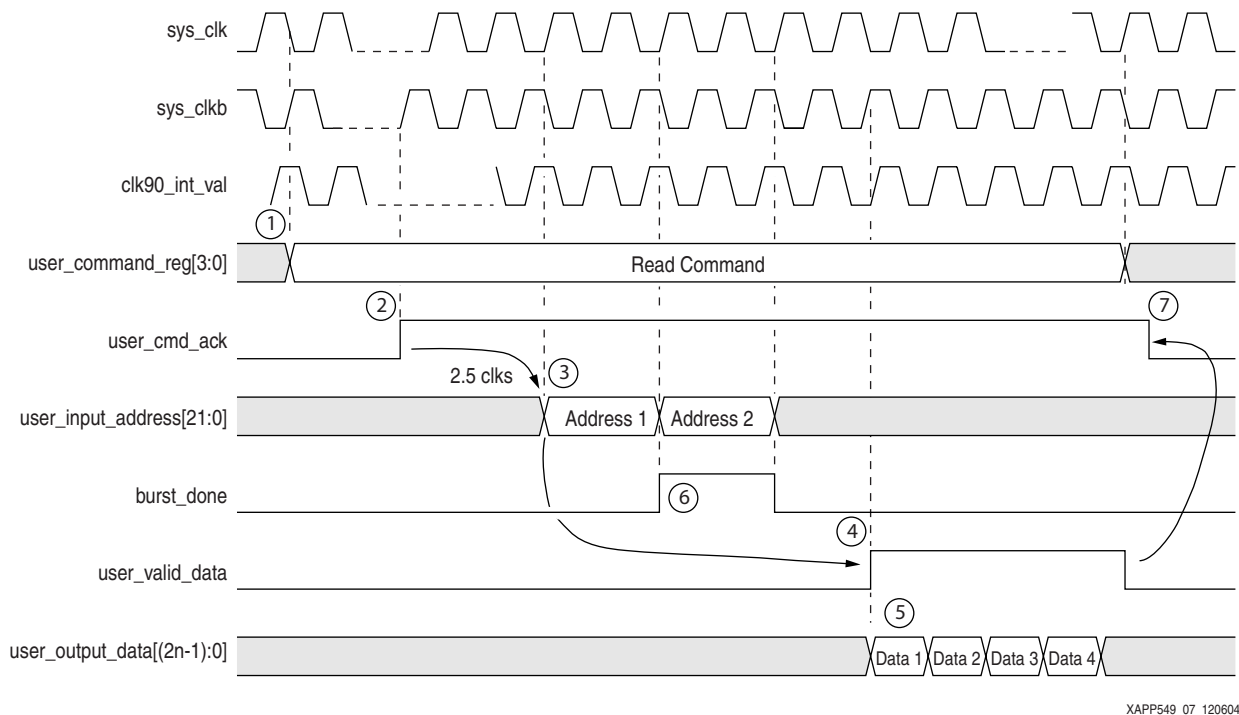


Figure 5: DDR2 SDRAM Memory Read Burst Length of Four

The read command flow is similar to the write command flow:

1. The user inputs the read command. It is accepted on the rising edge of the SYS_CLK.
2. The DDR2 SDRAM controller asserts the user_cmd_ack signal on the rising edge of the SYS_CLKb in response to the read command.
3. Two and half clocks after user_cmd_ack, the user places the memory read address on user_input_address [21:0]. The user_input_address signal is then accepted on the rising edge of SYS_CLK.
4. The data on user_output_data is valid only when the user_data_valid signal is asserted.

5. The data read from the DDR2 SDRAM memory is available on user_output_data. The user_output_data is asserted with clk90_int_val. Since the DDR SDRAM data is converted to SDR data, the width of this bus is 2n, where n is the data width of the DDR2 SDRAM memories. For a read burst length of four, with each user address the DDR2 SDRAM controller outputs only two data words.
6. To terminate the read burst, the burst_done signal is asserted for two clocks on the rising edge of SYS_CLK. The burst_done signal should be asserted with the last memory address.
7. Any further commands to the DDR2 SDRAM controller should be given after the user_cmd_ack signal is de-asserted.

DDR2 SDRAM Memory Auto_Refresh

The DDR2 SDRAM controller does not support memory refresh on its own and must periodically be provided with an auto_refresh command. The auto_refresh command is asserted with SYS_CLK. The ar_done signal is asserted by the DDR2 SDRAM controller upon completion of the auto_refresh command. The ar_done signal is asserted with SYS_CLKb.

Physical Layer and Delay Calibration

The physical layer for DDR2 SDRAM is similar to the DDR SDRAM physical layer described in application note XAPP768c. The delay calibration technique described in XAPP768c is also used in the DDR2 SDRAM interface.

Timing Calculations

Write Timing

Table 4: Write Data

Parameter	Value (ps)	Leading Edge Uncertainties	Trailing Edge Uncertainties	Meaning
Tclock	6000			Clock period
Tclock_phase	3000			Clock phase
Tdcd	250			Duty cycle distortion of clock to memory
Tdata_period	2750			Total data period, Tclock_phase-Tdcd
Tclock_skew	50	50	50	Minimal skew, since the right/left sides are used and the bits are close together
Tpackage_skew	90	90	90	Skew due to package pins and board layout (This can be reduced further with tighter layout.)
Tsetup	350	350	0	Setup time from memory data sheet
Thold	350	0	350	Hold time from memory data sheet
Tphase_offset_error	140	140	140	Offset error between different clocks from the same DCM
Tjitter	0	0	0	The same DCM is used to generate the clock and data; hence, they jitter together.
Total uncertainties	980	630	630	Worst case for leading and trailing can never happen simultaneously.
Window	1490	630	2120	Total worst-case window is 1490ps.

Read Timing

Table 5: Read Data

Parameter	Value (ps)	Leading Edge Uncertainties	Trailing Edge Uncertainties	Meaning
Tclock	6000			Clock period
Tclock_phase	3000			Clock phase
Tclock_duty_cycle_dist	300	0	0	Duty cycle distortion of clock to memory
Tdata_period	2700			Total data period, Tclock_phase-Tdcd
Tdqsq	350	350	0	Strobe to data distortion from memory data sheet
Tpackage_skew	90	90	90	Worst-case package skew
Tds	452	452	0	Setup time from Spartan-3 –5 data sheet
Tdh	-35	0	-35	Hold time from Spartan-3 –5 data sheet
Tjitter	100	0	0	Data and Strobe jitter together, since they are generated off of the same clock.
Tlocal_clock_line	20	20	20	Worst-case local clock line skew
Tpcb_layout_skew	50	50	50	Skew between data lines and strobes on the board
Tqhs	450	0	450	Hold skew factor for DQ from memory data sheet
Total uncertainties		962	575	Worst-case for leading and trailing can never happen simultaneously.
Window for DQS position for normal case	1163	962	2125	Worst-case window of 1163 ps.

Notes:

1. Reference for Tdqsq and Tqhs are from Micron data sheet for MT47H64M4FT-37E, Rev C, 05/04 EN.
2. Reference for Spartan-3 timing is –5 devices, Speeds file version 1.33.

Address and Command Timing

Table 6: Address and Command Data

Parameter	Value (ps)	Leading Edge Uncertainties	Trailing Edge Uncertainties	Meaning
Tclock	6000			Clock period
Tclock_skew	50	50	50	Minimal skew, since right/left sides are used and the bits are close together
Tpackage_skew	90	90	65	Using same bank reduces the package skew
Tsetup	500	500	0	Setup time from memory data sheet
Thold	500	0	500	Hold time from memory data sheet
Tphase_offset_error	140	140	140	Offset between different phases of the clock
Tduty_cycle_distortion	0	0	0	Duty cycle distortion does not apply
Tjitter	0	0	0	Since the clock and address are generated using the same clock, the same jitter exists in both; hence, it does not need to be included.
Total uncertainties		780	755	
Command window	3025	2220	5245	Worst-case window of 3025 ps

References

Xilinx Application Notes:

- [XAPP253](#), “Synthesizable 400 Mb/s DDR SDRAM Controller”
- XAPP768c, “Interfacing Spartan-3 Devices With 166 MHz or 333 Mb/s DDR SDRAM Memories” (available under click license)

Xilinx Reference Designs:

- <http://www.xilinx.com/bvdocs/apnotes/xapp253.zip>
- <http://www.xilinx.com/memory>

Micron Data Sheet MT47H16M16FG-37E, available online at:

<http://www.micron.com/products/dram/ddr2sdram/partlist.aspx?density=256Mb>

Conclusion

It is possible to implement a high-performance DDR2 SDRAM memory interface for Spartan-3 FPGAs. This design has been simulated, synthesized (with Synplicity), and taken through the Xilinx Project Navigator flow.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/06/04	1.0	Initial Xilinx release.



XAPP723 (v1.3) February 8, 2006

DDR2 Controller (267 MHz and Above) Using Virtex-4 Devices

Author: Karthi Palanisamy

Summary

DDR2 SDRAM devices offer new features that go beyond the DDR SDRAM specification and enable the DDR2 device to operate at data rates of 666 Mb/s. High data rates require higher performance from the controller and the I/Os in the FPGA. To achieve the desired bandwidth, it is essential for the controller to operate synchronously with the operating speed of the memory.

Introduction

This application note describes a 267 MHz and above DDR2 controller implementation in a Virtex™-4 device interfacing to a Micron DDR2 SDRAM device. For performance levels of 267 MHz and above, the controller design outlined in this application note should be used along with the read data capture technique explained in a separate application note entitled [XAPP721, High-Performance DDR2 SDRAM Interface Data Capture Using ISERDES and OSERDES](#).

This application note provides a brief overview of DDR2 SDRAM device features followed by a detailed explanation of the controller operation when interfacing to high-speed DDR2 memories. It also explains the backend user interface to the controller. A reference design in Verilog is available for download from the Xilinx website:

<http://www.xilinx.com/bvdocs/appnotes/xapp721.zip>.

DDR2 SDRAM Overview

DDR2 SDRAM devices are the next generation devices in the DDR SDRAM family. DDR2 SDRAM devices use the SSTL 1.8V I/O standard. The following section explains the features available in the DDR2 SDRAM devices and the key differences between DDR SDRAM and DDR2 SDRAM devices.

DDR2 SDRAM devices use a DDR architecture to achieve high-speed operation. The memory operates using a differential clock provided by the controller. Commands are registered at every positive edge of the clock. A bidirectional data strobe (DQS) is transmitted along with the data for use in data capture at the receiver. DQS is a strobe transmitted by the DDR2 SDRAM device during Reads and by the controller during Writes. DQS is edge-aligned with data for Reads and center-aligned with data for Writes.

Read and write accesses to the DDR2 SDRAM device are burst oriented; accesses begin with the registration of an Active command, which is then followed by a Read or Write command. The address bits registered with the Active command are used to select the bank and row to be accessed. The address bits registered with the Read or Write command are used to select the bank and the starting column location for the burst access.

The DDR2 controller reference design includes a user backend interface to generate the Write address, Write data, and Read addresses. This information is stored in three backend FIFOs for address and data synchronization between the backend and controller modules. Based on the availability of addresses in the address FIFO, the controller issues the correct commands to the memory, taking into account the timing requirements of the memory. The implementation details of the logic blocks are explained in the following sections.

DDR2 SDRAM Commands Issued by the Controller

Table 1 explains the commands issued by the controller. The commands are detected by the memory using the following control signals: Row Address Select ($\overline{\text{RAS}}$), Column Address Select ($\overline{\text{CAS}}$), and Write Enable ($\overline{\text{WE}}$) signals. Clock Enable (CKE) is held High after device configuration, and Chip select ($\overline{\text{CS}}$) is held Low throughout device operation. The [Mode Register Definition](#) section describes the DDR2 command functions supported in the controller.

Table 1: DDR2 Commands

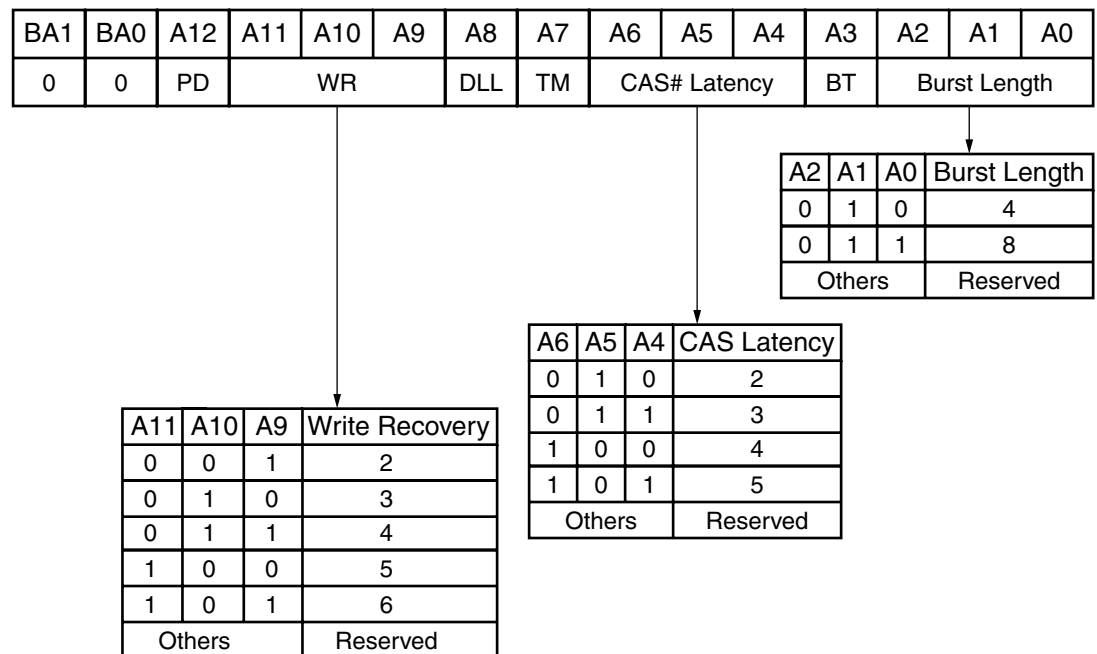
Step	Function	$\overline{\text{RAS}}$	$\overline{\text{CAS}}$	$\overline{\text{WE}}$
1	Load Mode	L	L	L
2	Auto Refresh	L	L	H
3	Precharge ⁽¹⁾	L	H	L
4	Bank Activate	L	H	H
5	Write	H	L	L
6	Read	H	L	H
7	No Operation/IDLE	H	H	H

Notes:

- Address signal A10 is held High during *Precharge All Banks* and is held Low during single bank precharge.

Mode Register Definition

The Mode register is used to define the specific mode of operation of the DDR2 SDRAM. This includes the selection of burst length, burst type, CAS latency, and operating mode. [Figure 1](#) explains the Mode register features used by this controller.



x723_01_091505

Figure 1: Mode Register

Bank Addresses BA1 and BA0 select the Mode registers. [Table 2](#) shows the Bank Address bit configuration.

Table 2: Bank Address Bit Configuration

BA1	BA0	Mode Register
0	0	Mode Register (MR)
0	1	EMR1
1	0	EMR2
1	1	EMR3

Extended Mode Register Definition

The extended Mode register ([Table 3](#)) controls functions beyond those controlled by the Mode register. These additional functions are DLL enable/disable, output drive strength, On Die Termination (ODT), Posted CAS Additive Latency (AL), off-chip driver impedance calibration (OCD), \overline{DQS} enable/disable, $\overline{RDQS}/\overline{RDQS}$ enable/disable, and OUTPUT disable/enable. Off-chip Driver Calibration (OCD) is not used in this reference design.

Table 3: Extended Mode Register

BA1	BA0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	Out	\overline{RDQS}	\overline{DQS}	OCD Program			R_{TT}	Posted \overline{CAS}			R_{TT}	ODS	DLL

Extended Mode Register 2 (EMR2)

Bank Addresses are set to 10 (BA1 is set High, and BA0 is set Low). The address bits are all set to Low.

Extended Mode Register 3 (EMR3)

Bank Address bits are set to 11 (BA1 and BA0 are set High). Address bits are all set Low, as in EMR2.

Initialization Sequence

The initialization sequence used in the controller state machine follows the DDR2 SDRAM specifications. The voltage requirements of the memory need to be met by the interface. The following is the sequence of commands issued for initialization.

1. After stable power and clock, a NOP or Deselect command is applied for 200 μ s.
2. CKE is asserted.
3. Precharge All command after 400 ns.
4. EMR (2) command. BA0 is held Low, and BA1 is held High.
5. EMR (3) command. BA0 and BA1 are both held High.
6. EMR command to enable the memory DLL. BA1 and A0 are held Low, and BA0 is held High.
7. Mode Register Set command for DLL reset. To lock the DLL, 200 clock cycles are required.
8. Precharge All command.
9. Two Auto Refresh commands.
10. Mode Register Set command with Low to A8, to initialize device operation.
11. EMR command to enable OCD default by setting bits E7, E8, and E9 to 1.
12. EMR command to enable OCD exit by setting bits E7, E8 and E9 to 0.

After the initialization sequence is complete, the controller issues a dummy write followed by dummy reads to the DDR2 SDRAM memory for the datapath module to select the right number of taps in the Virtex-4 input delay block. The datapath module determines the right number of delay taps required and then asserts the `dp_dly_slct_done` signal to the controller. The controller then moves into the IDLE state.

Precharge Command

The Precharge command is used to deactivate the open row in a particular bank. The bank is available for a subsequent row activation a specified time (t_{RP}) after the Precharge command is issued. Input A10 determines whether one or all banks are to be precharged.

Auto Refresh Command

DDR2 devices need to be refreshed every 7.8 μ s. The circuit to flag the Auto Refresh commands is built into the controller. The controller uses a system clock divided by 16 output as the refresh counter. When asserted, the `auto_ref` signal flags the need for Auto Refresh commands. The `auto_ref` signal is held High 7.8 μ s after the previous Auto Refresh command. The controller then issues the Auto Refresh command after it has completed its current burst. Auto Refresh commands are given the highest priority in the design of this controller.

Active Command

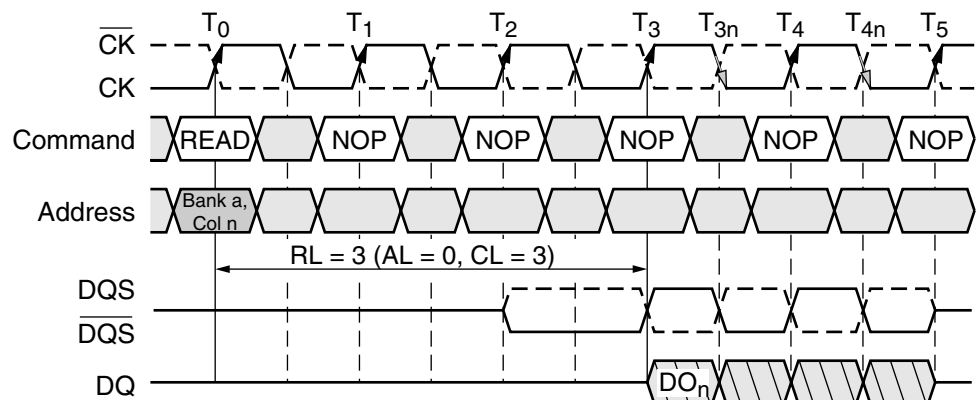
Before any Read or Write commands can be issued to a bank within the DDR2 SDRAM memory, a row in the bank must be activated using an Active command. After a row is opened, Read or Write commands can be issued to the row subject to the t_{RCD} specification. DDR2 SDRAM devices also support posted CAS additive latencies; these allow a Read or Write command to be issued prior to the t_{RCD} specification by delaying the actual registration of the Read or Write command to the internal device using additive latency clock cycles.

When the controller detects a conflict, it issues a Precharge command to deactivate the open row and then issues another Active command to the new row. A conflict occurs when an incoming address refers to a row in a bank other than the currently opened row.

Read Command

The Read command is used to initiate a burst read access to an active row. The values on BA0 and BA1 select the bank address, and the address inputs provided on A₀ - A_i select the starting column location. After the read burst is over, the row is still available for subsequent access until it is precharged.

Figure 2 shows an example of a Read command with an additive latency of zero. Hence, in this example, the Read latency is three, the same as the CAS latency.



x723_02_091505

Figure 2: Read Command Example

Write Command

The Write command is used to initiate a burst access to an active row. The value on BA0 and BA1 select the bank address while the value on address inputs A₀ - A_i select the starting column location in the active row. DDR2 SDRAMs use a write latency equal to read latency minus one clock cycle.

$$\text{Write Latency} = \text{Read Latency} - 1 = (\text{Additive Latency} + \text{CAS Latency}) - 1$$

Figure 3 shows the case of a Write burst with a Write latency of 2. The time between the Write command and the first rising edge of the DQS signal is determined by the WL.

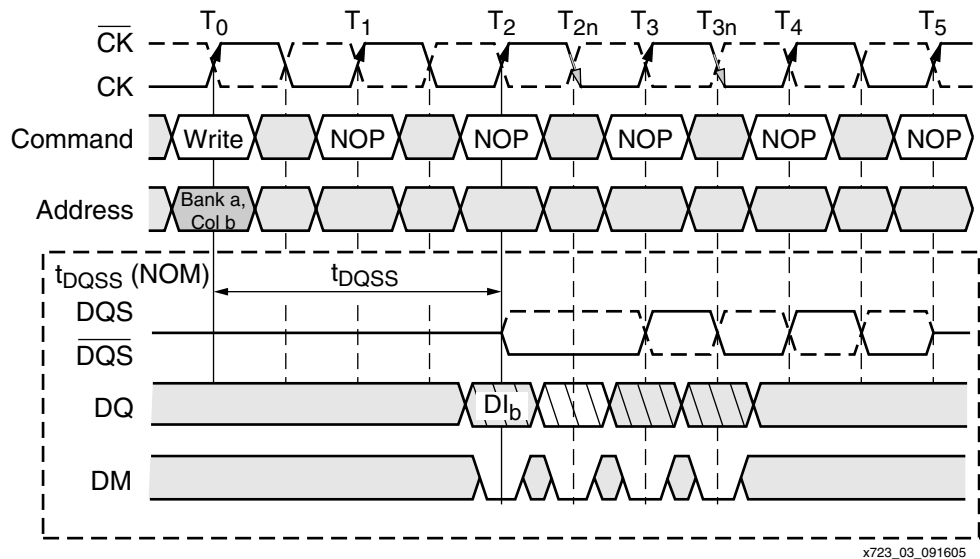


Figure 3: Write Command Example

DDR2 SDRAM Interface Design

The user interface to the DDR2 controller (Figure 4) and the datapath are clocked at half the frequency of the interface, resulting in improved design margin at frequencies above 267 MHz. The operation of the controller at half the frequency does not affect the throughput or latency. DDR2 SDRAM devices support a minimum burst size of 4, only requiring a command every other clock. For a burst of 4, the controller issues a command every controller clock (the slow clock). For a burst of 8, the controller issues a command every other controller clock (the slow clock). All the FIFOs in the user interface are asynchronous FIFOs, allowing the user's backend to operate at any frequency. The I/Os toggle at the target frequency.

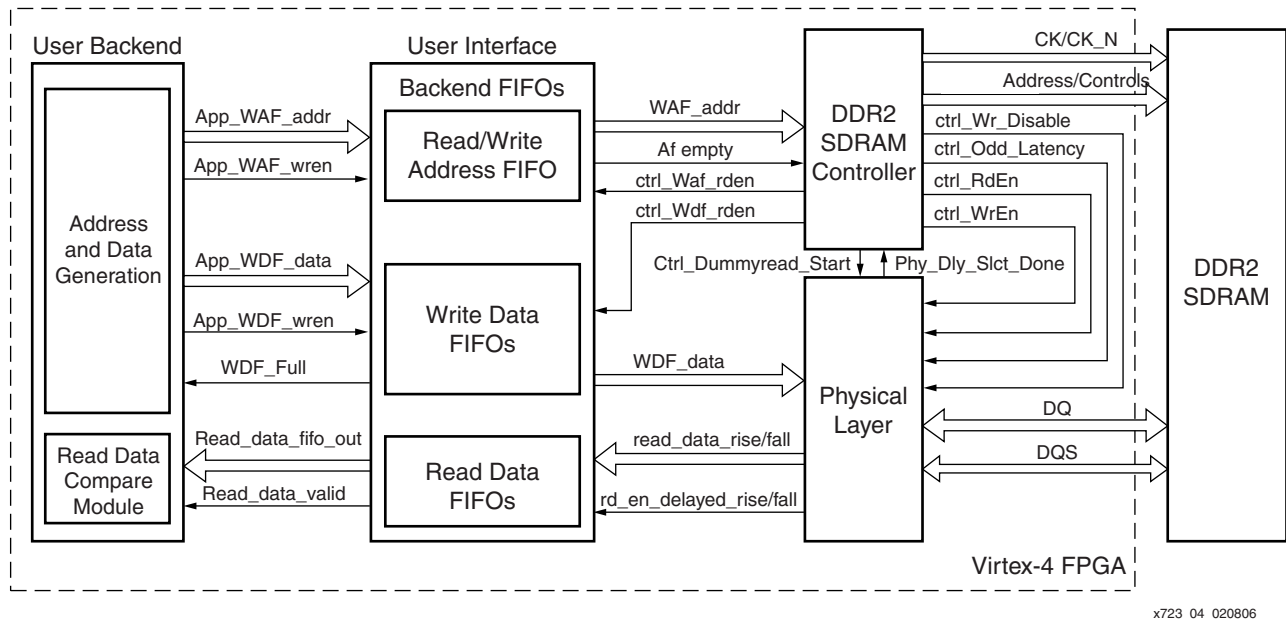


Figure 4: DDR2 Complete Interface Block Diagram

User Backend

The backend is designed to provide address and data patterns to test all the design aspects of a DDR2 controller. The backend includes the following blocks: backend state machine, read data comparator, and a data generator module. The data generation module generates the various address and data patterns that are written to the memory. The address locations are pre-stored in a block RAM, being used here as a ROM. The address values stored have been selected to test accesses to different rows and banks in the DDR2 SDRAM device. The data pattern generator includes a state machine issuing patterns of data. The backend state machine emulates a user backend. This state machine issues the write or read enable signals to determine the specific FIFO that will be accessed by the data generator module.

User Interface

The backend user interface has three FIFOs: the Address FIFO, the Write Data FIFO, and the Read Data FIFO. The first two FIFOs are accessed by the user backend modules, while the Read Data FIFO is accessed by the Datapath module used to store the captured Read data.

User-to-Controller Interface

Table 4 lists the signals between the user interface and the controller.

Table 4: Signals Between User Interface and Controller

Port Name	Port Width	Port Description	Notes
Af_addr	36	Output of the Address FIFO in the user interface. Mapping of these address bits: <ul style="list-style-type: none"> Memory Address (CS, Bank, Row, Column) - [31:0] Reserved - [32] Dynamic Command Request - [35:33] 	Monitor FIFO-full status flag to write address into the address FIFO.
Af_empty	1	The user interface Address FIFO empty status flag output. The controller processes the address on the output of the FIFO when this signal is deasserted.	FIFO16 Empty Flag.
ctrl_Waf_RdEn	1	Read Enable input to address FIFO in the user interface.	This signal is asserted for one clock cycle when the controller state is Write, Read, Load Mode Register, Precharge All, Auto Refresh, or Active resulting from dynamic command requests.
ctrl_Wdf_RdEn	1	Read Enable input to Write Data FIFO in the user interface.	The controller asserts this signal for one clock cycle after the first write state. This signal is asserted for two clock cycles for a burst length of 8. Sufficient data must be available in Write Data FIFO associated with a write address for the required burst length before issuing a write command. For example, for a 64-bit data bus and a burst length of 4, the user should input two 128-bit data words in the Write Data FIFO for every write address before issuing the write command.

The memory address (Af_addr) includes the column address, row address, bank address, and chip-select width for deep memory interfaces (Table 5).

Table 5: Af_addr Memory Address

Address	Description
Column Address	col_ap_width – 1:0
Row Address	col_ap_width + row_address – 1:col_ap_width
Bank Address	col_ap_width + row_address + bank_address – 1:col_ap_width + row_address
Chip Select	col_ap_width + row_address + bank_address + chip_address – 1:col_ap_width + row_address + bank_address

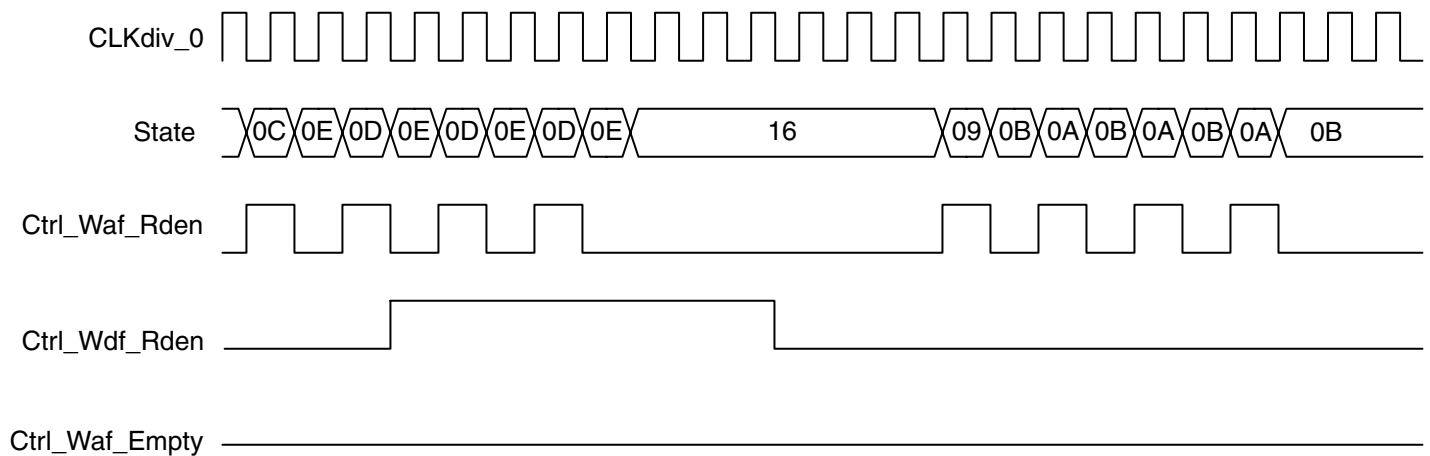
Dynamic Command Request

Table 6 lists the optional commands. These commands are not required for normal operation of the controller. The user has the option to request these commands when required by an application.

Table 6: Optional Commands

Command	Description
000	Load Mode Register
001	Auto Refresh
010	Precharge All
011	Active
100	Write
101	Read
110	NOP
111	NOP

Figure 5 describes four consecutive Writes followed by four consecutive Reads with a burst length of 8. Table 7 lists the state signal values for Figure 5.



X723_05_091905

Figure 5: Consecutive Reads Followed by Consecutive Writes with Burst Length of 8

Table 7: Values for State Signals in Figure 5

State	Description
0C	First Write
0E	Write Wait
0D	Burst Write
16	Write Read
09	First Write
0B	Read Wait
0A	Burst Read

Controller to Physical Layer Interface

Table 8 lists the signals between the controller and the physical layer. Figure 6 describes the timing waveform for control signals from the controller to the physical layer.

Table 8: Signals Between the Controller and Physical Layer

Signal Name	Signal Width	Signal Description	Notes
ctrl_WrEn	1	Output from the controller to the write datapath. Write DQS and DQ generation begins when this signal is asserted.	Asserted for two controller clock cycles for a burst length of 4 and three controller clock cycles for a burst length of 8. Asserted one controller clock cycle earlier than the WRITE command for CAS latency values of 4 and 5.
ctrl_wr_disable	1	Output from the controller to the write datapath. Write DQS and DQ generation ends when this signal is deasserted.	Asserted for one controller clock cycle for a burst length of 4 and two controller clock cycles for a burst length of 8. Asserted one controller clock cycle earlier than the WRITE command for CAS latency values of 4 and 5.
ctrl_Odd_Latency	1	Output from controller to write datapath. Asserted when the selected CAS latency is an odd number. Required for generation of write DQS and DQ after the correct write latency (CAS latency – 1).	
ctrl_Dummyread_Start	1	Output from the controller to the read datapath. When this signal is asserted, the strobe and data calibration begin.	This signal must be asserted when valid read data is available on the data bus. This signal is deasserted when the dp_dly_slct_done signal is asserted.
dp_dly_slct_done	1	Output from the read datapath to the controller indicating the strobe and data calibration are complete.	This signal is asserted when the data and strobe have been calibrated. Normal operation begins after this signal is asserted.
ctrl_RdEn	1	Output from the controller to the read datapath for a read-enable signal.	This signal is asserted for one controller clock cycle for a burst length of 4 and two controller clock cycles for a burst length of 8. The CAS latency and additive latency values determine the timing relationship of this signal with the read state.

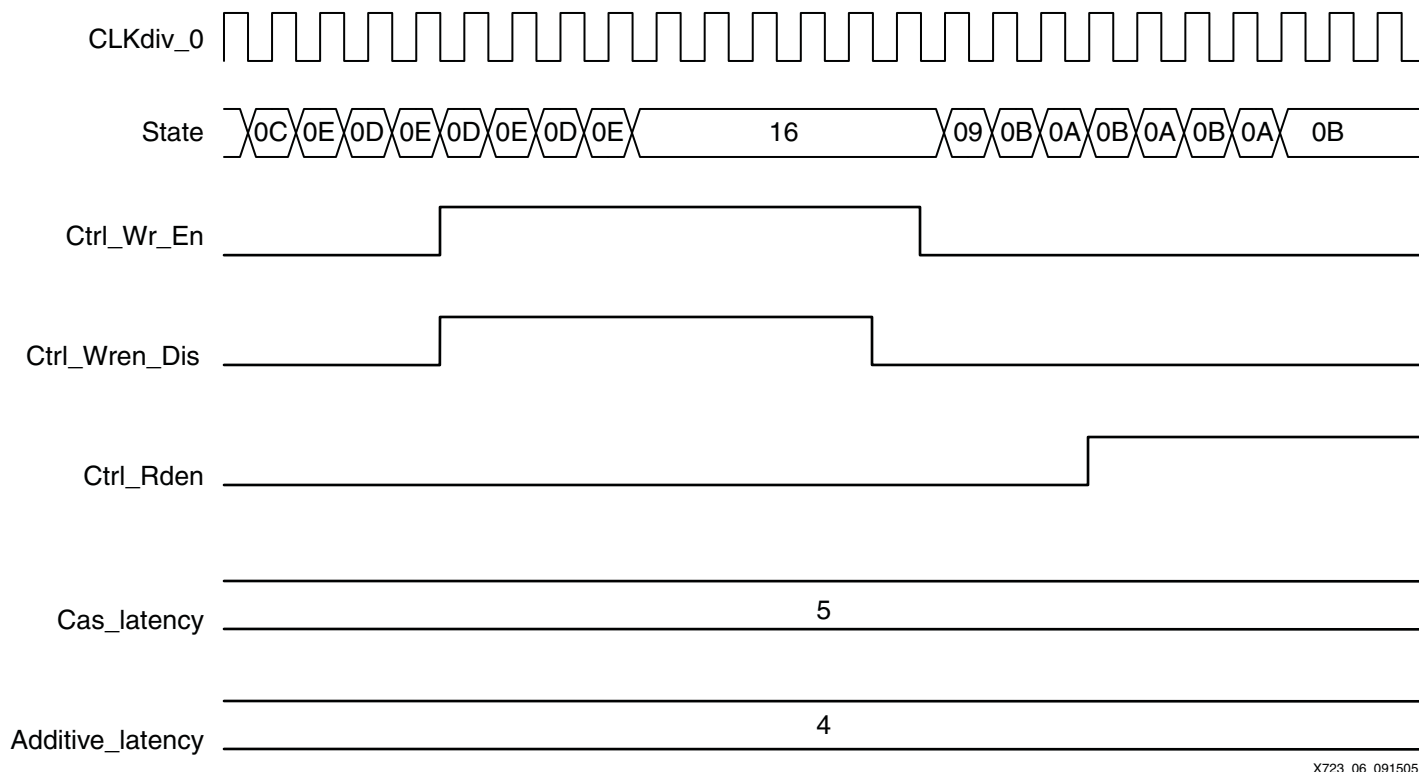


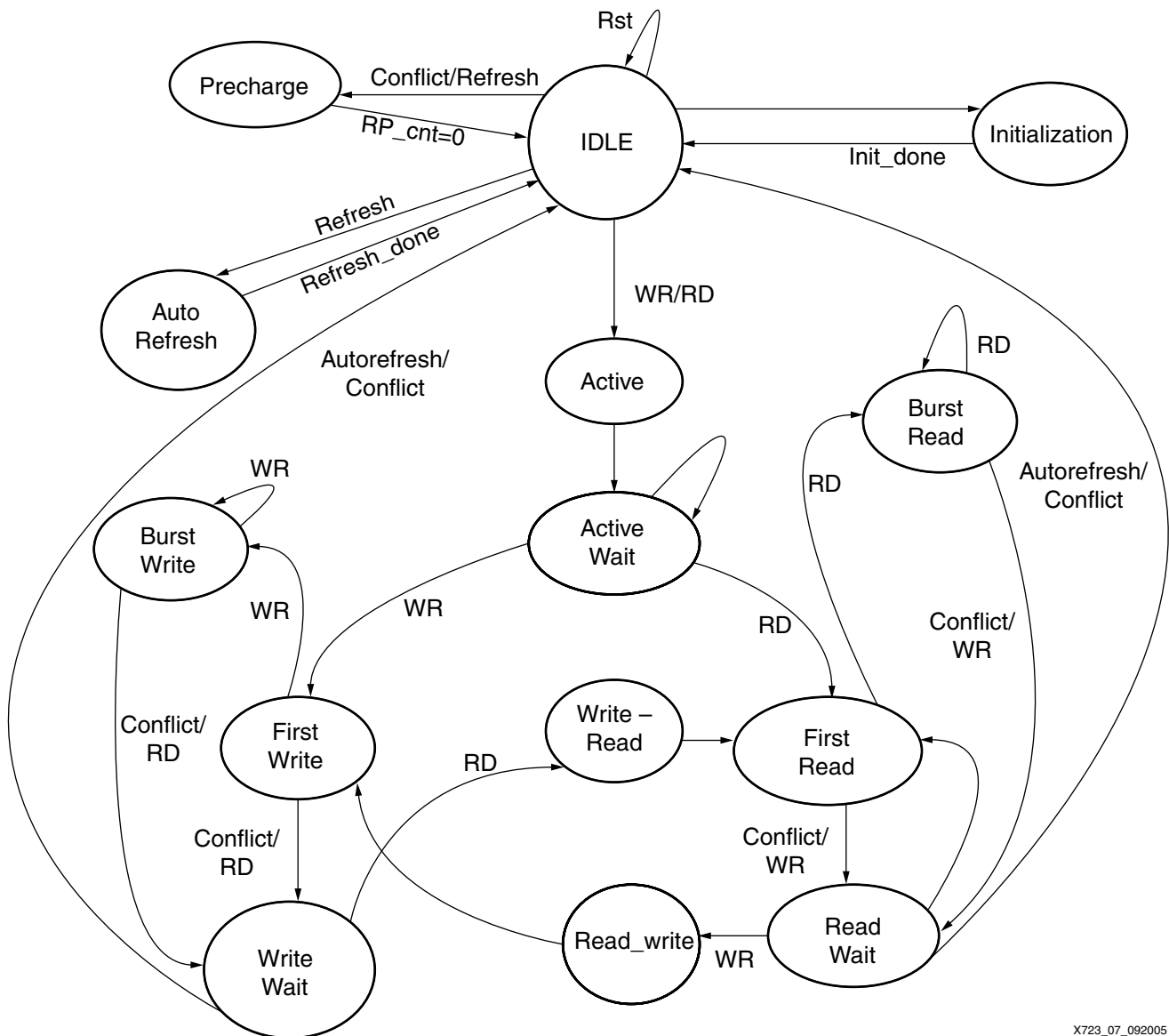
Figure 6: Timing Waveform for Control Signals from the Controller to the Physical Layer

Controller Implementation

The controller is clocked at the half the frequency of the interfaces. Therefore, the address, bank address, and command signals ($\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{WE}}$) are asserted for two clock cycles of the fast memory interface clocks. The control signals ($\overline{\text{CS}}$, CKE , and ODT) are DDR of the half frequency clocks, ensuring that the control signals are asserted for just one clock cycle of the fast memory interface clock.

The controller state machine manages issuing the commands in the correct sequencing order while determining the timing requirements of the memory.

Along with [Figure 7](#), the following sections explain in detail the various stages of the controller state machine.



X723_07_092005

Figure 7: DDR2 Controller State Machine

Before the controller issues the commands to the memory:

1. The address FIFO is in first-word-fall-through mode (FWFT). In FWFT mode, the first address written into the FIFO appears at the output of the FIFO. The controller decodes the address.
2. The controller activates a row in the corresponding bank if all banks have been precharged, or it compares the bank and row addresses to the already open row and bank address. If there is a conflict, the controller precharges the open bank and then issues an Active command before moving to the Read/Write states.
3. After arriving in the Write state, if the controller gets a Read command, the controller waits for the write_to_read time before issuing the Read command. Similarly, in the Read state, when the controller sees a Write command from the command logic block, the controller waits for the read_to_write time before issuing the Write command. In the read or write state, the controller also asserts the read enable to the address FIFO to get the next address.
4. The commands are pipelined to synchronize with the Address signals before being issued to the DDR2 memory.

Design Hierarchy

Figure 8 shows the design hierarchy beginning with a top-level module called `mem_interface_top`.

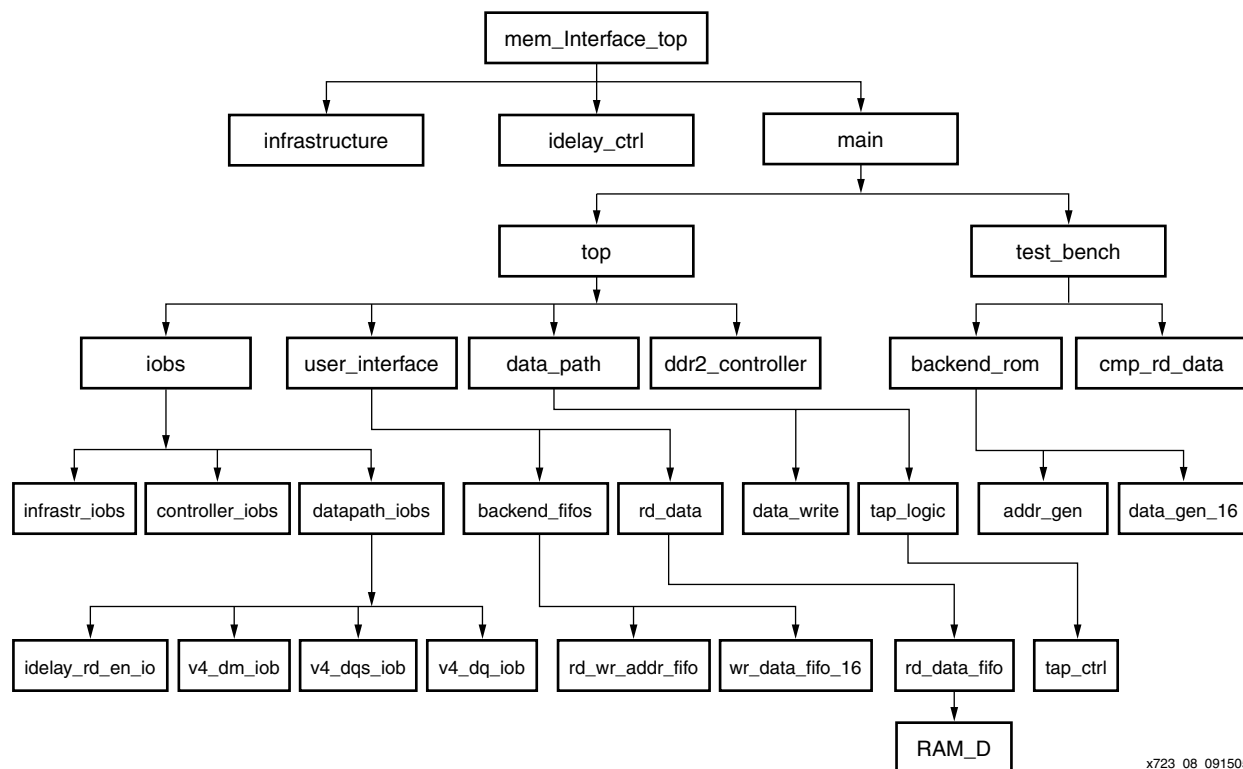


Figure 8: Reference Design Hierarchy

Resource Utilization

The resource utilization for a 64-bit DDR2 SDRAM interface including the synthesizable test bench is listed in Table 9.

Table 9: Resource Utilization

Resources	Utilization	Notes
Slices	3198	Includes the controller, synthesizable test bench, and the user interface.
BUFGs	6	Includes one BUFG for the 200 MHz IDELAY block reference clock.
BUFIOs	8	Equals the number of strobes in the interface.
DCMs	1	
PMCDs	2	
ISERDES	64	Equals the number of data bits in the interface.
OSERDES	90	Equals the sum of the data bits, strobes, and data mask signals.

The reference design for the 64-bit DDR2 SDRAM interface using the data capture technique is available for download on the Xilinx website at:

<http://www.xilinx.com/bvdocs/appnotes/xapp721.zip>.

Conclusion

The DDR2 controller described in this application note, along with the data capture method from XAPP721, provide a good solution for high-performance memory interfaces. This design provides high margin because all the logic in the FPGA fabric is clocked at half the frequency of the interface, eliminating critical paths. This design was verified in hardware.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/15/05	1.0	Initial Xilinx release.
12/16/05	1.1	Updated Table 8 and Table 9 .
02/02/06	1.2	Updated Figure 4 .
02/08/06	1.3	Updated Figure 4 .



XAPP721 (v1.3) February 2, 2006

High-Performance DDR2 SDRAM Interface Data Capture Using ISERDES and OSERDES

Author: Maria George

Summary

This application note describes a data capture technique for a high-performance DDR2 SDRAM interface. This technique uses the Input Serializer/Deserializer (ISERDES) and Output Serializer/Deserializer (OSERDES) features available in every Virtex™-4 I/O. This technique can be used for memory interfaces with frequencies of 267 MHz (533 Mb/s) and above.

Introduction

A DDR2 SDRAM interface is source-synchronous where the read data and read strobe are transmitted edge-aligned. To capture this transmitted data using Virtex-4 FPGAs, either the strobe or the data can be delayed. In this design, the read data is captured in the delayed strobe domain and recaptured in the FPGA clock domain in the ISERDES. The received serial, double data rate (DDR) read data is converted to 4-bit parallel single data rate (SDR) data at half the frequency of the interface using the ISERDES. The differential strobe is placed on a clock-capable IO pair in order to access the BUFIO clock resource. The BUFIO clocking resource routes the delayed read DQS to its associated data ISERDES clock inputs. The write data and strobe transmitted by the FPGA use the OSERDES. The OSERDES converts 4-bit parallel data at half the frequency of the interface to DDR data at the interface frequency. The controller, datapath, user interface, and all other FPGA slice logic are clocked at half the frequency of the interface, resulting in improved design margin at frequencies of 267 MHz and above.

Clocking Scheme

The clocking scheme for this design includes one digital clock manager (DCM) and two phase-matched clock dividers (PMCDs) as shown in [Figure 1](#). The controller is clocked at half the frequency of the interface using CLKdiv_0. Therefore, the address, bank address, and command signals (RAS_L, CAS_L, and WE_L) are asserted for two clock cycles (known as "2T" timing), of the fast memory interface clock. The control signals (CS_L, CKE, and ODT) are twice the rate (DDR) of the half frequency clock CLKdiv_0, ensuring that the control signals are asserted for just one clock cycle of the fast memory interface clock. The clock is forwarded to the external memory device using the Output Dual Data Rate (ODDR) flip-flops in the Virtex-4 I/O. This forwarded clock is 180 degrees out of phase with CLKfast_0. [Figure 2](#) shows the command and control timing diagram.

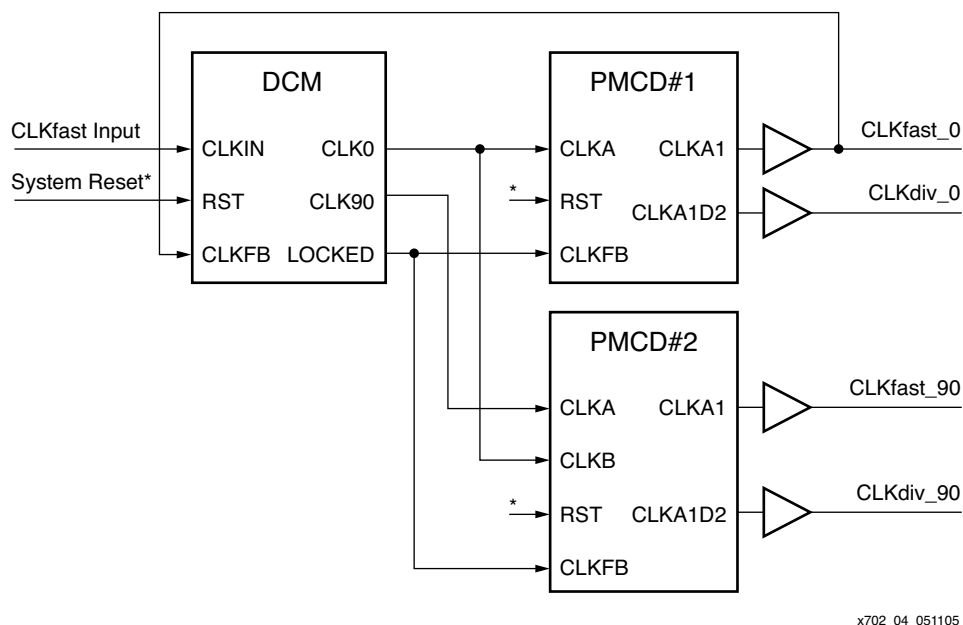


Figure 1: Clocking Scheme for the High-Performance Memory Interface Design

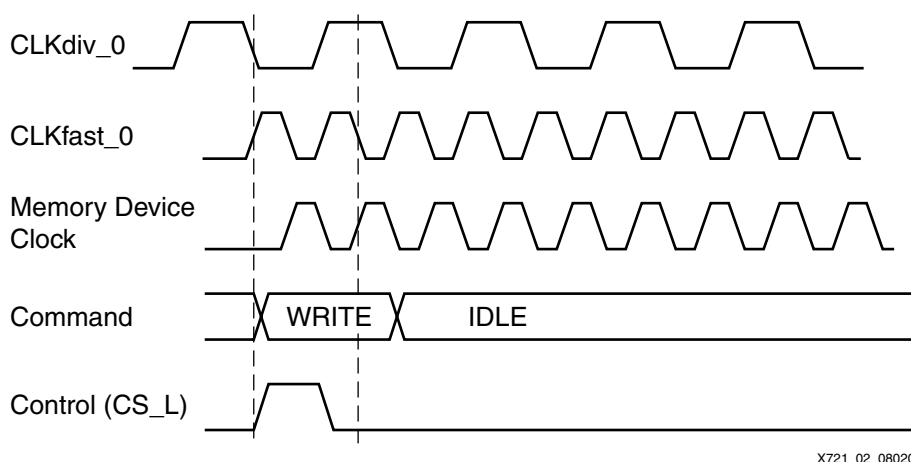
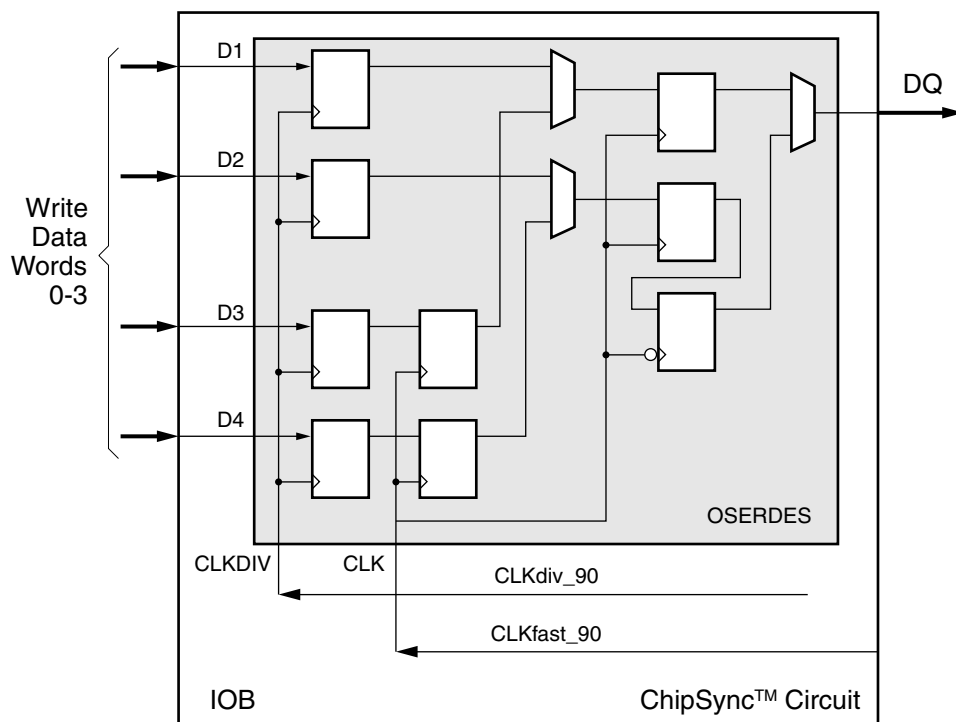


Figure 2: Command and Control Timing

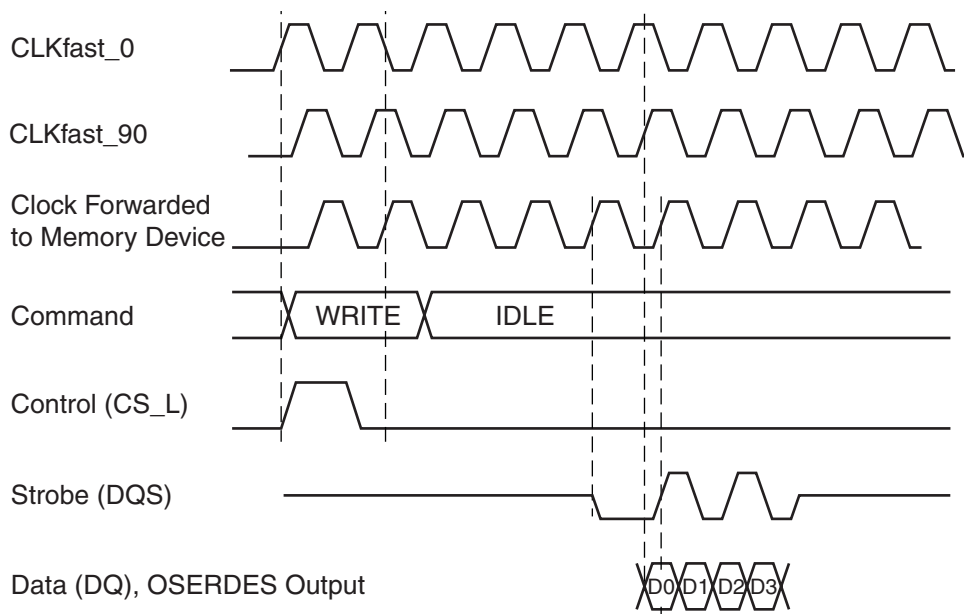
Write Datapath

The write datapath uses the built-in OSERDES available in every Virtex-4 I/O. The OSERDES transmits the data (DQ) and strobe (DQS) signals. The memory specification requires DQS to be transmitted center-aligned with DQ. The strobe (DQS) forwarded to the memory is 180 degrees out of phase with CLKfast_0. Therefore, the write data transmitted using OSERDES must be clocked by CLKfast_90 and CLKdiv_90 as shown in [Figure 3](#). The timing diagram for write DQS and DQ is shown in [Figure 4](#).



X721_03_080305

Figure 3: Write Data Transmitted Using OSERDES



X721_04_120505

Figure 4: Write Strobe (DQS) and Data (DQ) Timing for a Write Latency of Four

Write Timing Analysis

Table 1 shows the write timing analysis for an interface at 333 MHz (667 Mb/s).

Table 1: Write Timing Analysis at 333 MHz

Uncertainty Parameters	Value	Uncertainties before DQS	Uncertainties after DQS	Meaning
T_{CLOCK}	3000			Clock period.
$T_{\text{MEMORY_DLL_DUTY_CYCLE_DIST}}$	150	150	150	Duty-cycle distortion from memory DLL is subtracted from clock phase (equal to half the clock period) to determine $T_{\text{DATA_PERIOD}}$.
$T_{\text{DATA_PERIOD}}$	1350			Data period is half the clock period with 10% duty-cycle distortion subtracted from it.
T_{SETUP}	100	100	0	Specified by memory vendor.
T_{HOLD}	175	0	175	Specified by memory vendor.
$T_{\text{PACKAGE_SKEW}}$	30	30	30	PCB trace delays for DQS and its associated DQ bits are adjusted to account for package skew. The listed value represents dielectric constant variations.
T_{JITTER}	50	50	50	Same DCM used to generate DQS and DQ.
$T_{\text{CLOCK_SKEW-MAX}}$	50	50	50	Global Clock Tree skew.
$T_{\text{CLOCK_OUT_PHASE}}$	140	140	140	Phase offset error between different clock outputs of the same DCM.
$T_{\text{PCB_LAYOUT_SKEW}}$	50	50	50	Skew between data lines and the associated strobe on the board.
Total Uncertainties		420	495	
Start and End of Valid Window		420	855	
Final Window			435	Final window equals 855 – 420.

Notes:

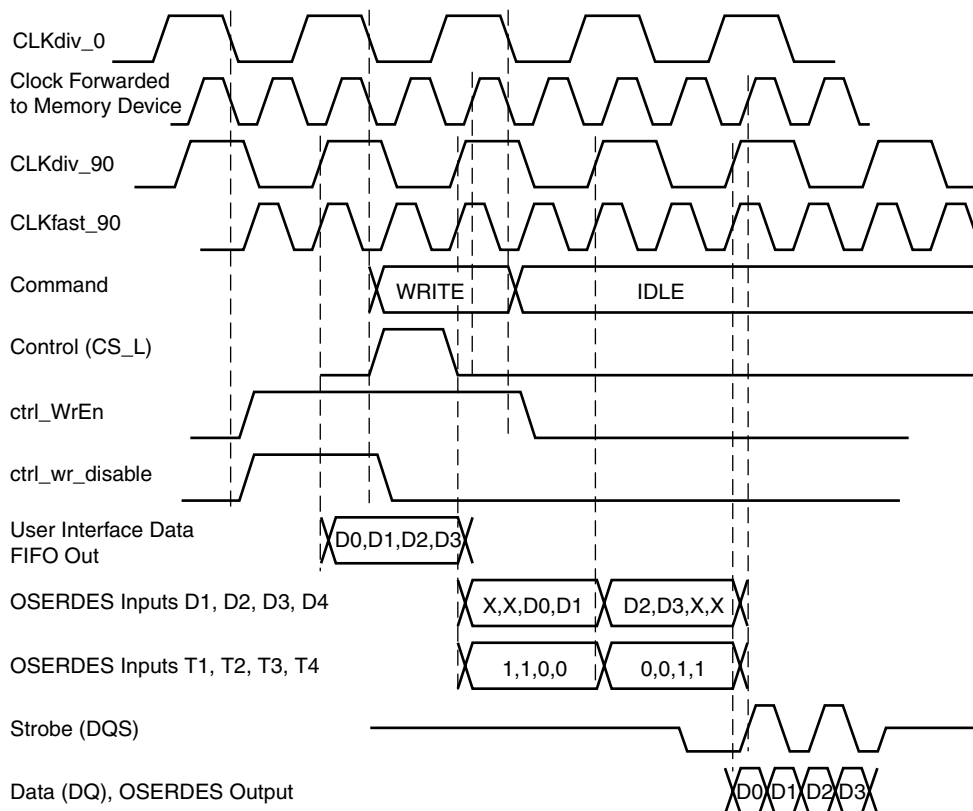
1. Skew between output flip-flops and output buffers in the same bank is considered to be minimal over voltage and temperature.

Controller to Write Datapath Interface

Table 2 lists the signals required from the controller to the write datapath.

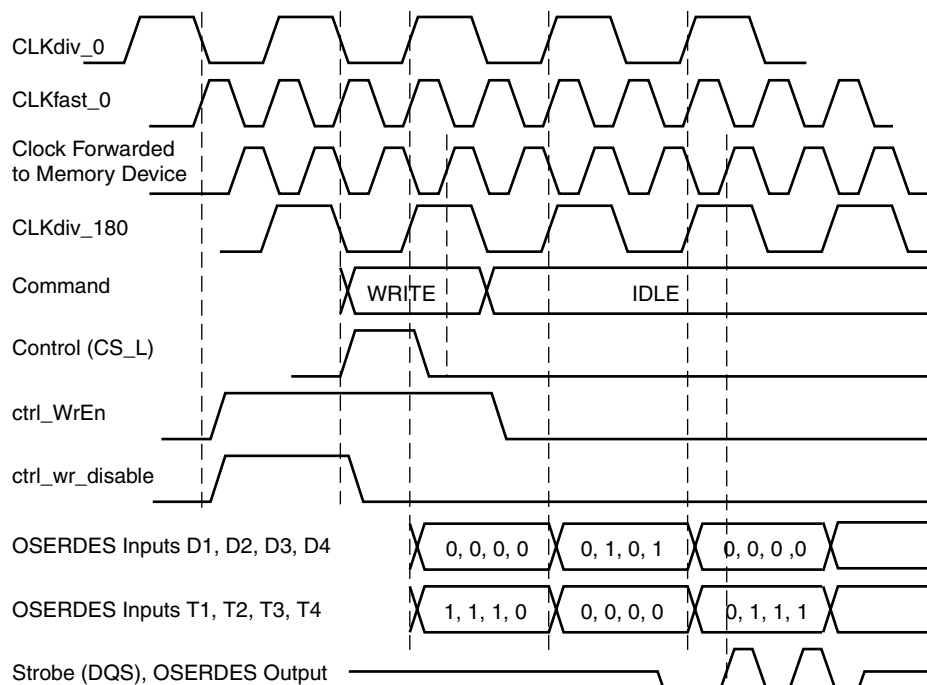
Table 2: Controller to Write Datapath Signals

Signal Name	Signal Width	Signal Description	Notes
ctrl_WrEn	1	Output from the controller to the write datapath. Write DQS and DQ generation begins when this signal is asserted.	Asserted for two CLKDIV_0 cycles for a burst length of 4 and three CLKDIV_0 cycles for a burst length of 8. Asserted one CLKDIV_0 cycle earlier than the WRITE command for CAS latency values of 4 and 5. Figure 5 and Figure 6 show the timing relationship of this signal with respect to the WRITE command.
ctrl_wr_disable	1	Output from the controller to the write datapath. Write DQS and DQ generation ends when this signal is deasserted.	Asserted for one CLKDIV_0 cycle for a burst length of 4 and two CLKDIV_0 cycles for a burst length of 8. Asserted one CLKDIV_0 cycle earlier than the WRITE command for CAS latency values of 4 and 5. Figure 5 and Figure 6 show the timing relationship of this signal with respect to the WRITE command.
ctrl_Odd_Latency	1	Output from controller to write datapath. Asserted when the selected CAS latency is an odd number, e.g., 5. Required for generation of write DQS and DQ after the correct write latency (CAS latency – 1).	



X721_05_080205

Figure 5: Write DQ Generation with a Write Latency of 4 and a Burst Length of 4



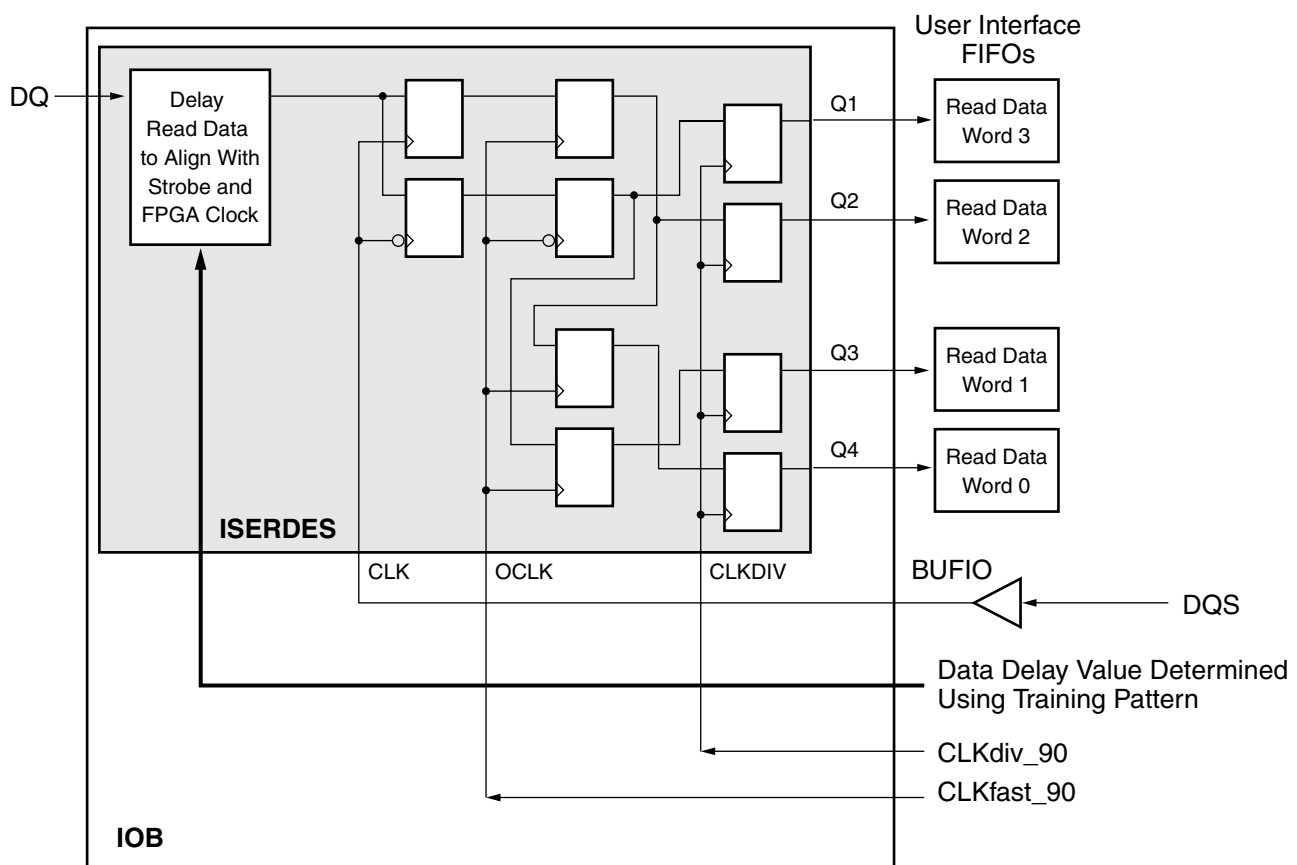
X721_06_080205

Figure 6: Write DQS Generation for a Write Latency of 4 and a Burst Length of 4

Read Datapath

The read datapath comprises the read data capture and recapture stages. Both stages are implemented in the built-in ISERDES available in every Virtex-4 I/O. The ISERDES has three clock inputs: CLK, OCLK, and CLKDIV. The read data is captured in the CLK (DQS) domain, recaptured in the OCLK (FPGA fast clock) domain, and finally transferred to the CLKDIV (FPGA divided clock) domain to provide parallel data.

- **CLK:** The read DQS routed using BUFIO provides the CLK input of the ISERDES as shown in [Figure 7](#).
- **OCLK:** The OCLK input of ISERDES is connected to the CLK input of OSERDES in hardware. In this design, the CLKfast_90 clock is provided to the ISERDES OCLK input and the OSERDES CLK input. The clock phase used for OCLK is dictated by the phase required for write data.
- **CLKDIV:** It is imperative for OCLK and CLKDIV clock inputs to be phase-aligned for correct functionality. Therefore, the CLKDIV input is provided with CLKdiv_90 that is phase-aligned to CLKfast_90.



X721_07_063005

Figure 7: Read Data Capture Using ISERDES

Read Timing Analysis

To capture read data without errors in the ISERDES, read data and strobe must be delayed to meet the setup and hold times of the flip-flops in the FPGA clock domain. Read data (DQ) and strobe (DQS) are received edge-aligned at the FPGA. The differential DQS pair must be placed on a clock-capable IO pair in order to access the BUFIO resource. The received read DQS is then routed through the BUFIO resource to the CLK input of the ISERDES of the associated data bits. The delay through the BUFIO and clock routing resources shifts the DQS to the right with respect to data. The total delay through the BUFIO and clock resource is 595 ps in a -11 speed grade device and 555 ps in a -12 speed grade device.

Table 3 shows the read timing analysis at 333 MHz required to determine the delay required on DQ bits for centering DQS in the data valid window.

Table 3: Read Timing Analysis at 333 MHz

Parameter	Value (ps)	Meaning
T_{CLOCK}	3000	Clock period.
T_{PHASE}	1500	Clock phase for DDR data.
$T_{\text{SAMP_BUFIO}}$	350	Sample Window from Virtex-4 data sheet for a -12 device. It includes setup and hold for an IOB FF, clock jitter, and 150 ps of tap uncertainty.
$T_{\text{BUFIO_DCD}}$	100	BUFIO clock resource duty-cycle distortion.
$T_{\text{DQSQ}} + T_{\text{QHS}}$	580	Worst case memory uncertainties that include VT variations and skew between DQS and its associated DQs. Because the design includes per bit deskew, realistically only a percentage of this number should be considered.
$T_{\text{MEM_DCD}}$	150	Duty-cycle distortion.
Tap Uncertainty	0	Tap uncertainty with 75 ps resolution. A window detection error of 75 ps can be on both ends of the window. This is already included in $T_{\text{SAMP_BUFIO}}$.
Total Uncertainties	1180	
Window	320	Worst-case window.

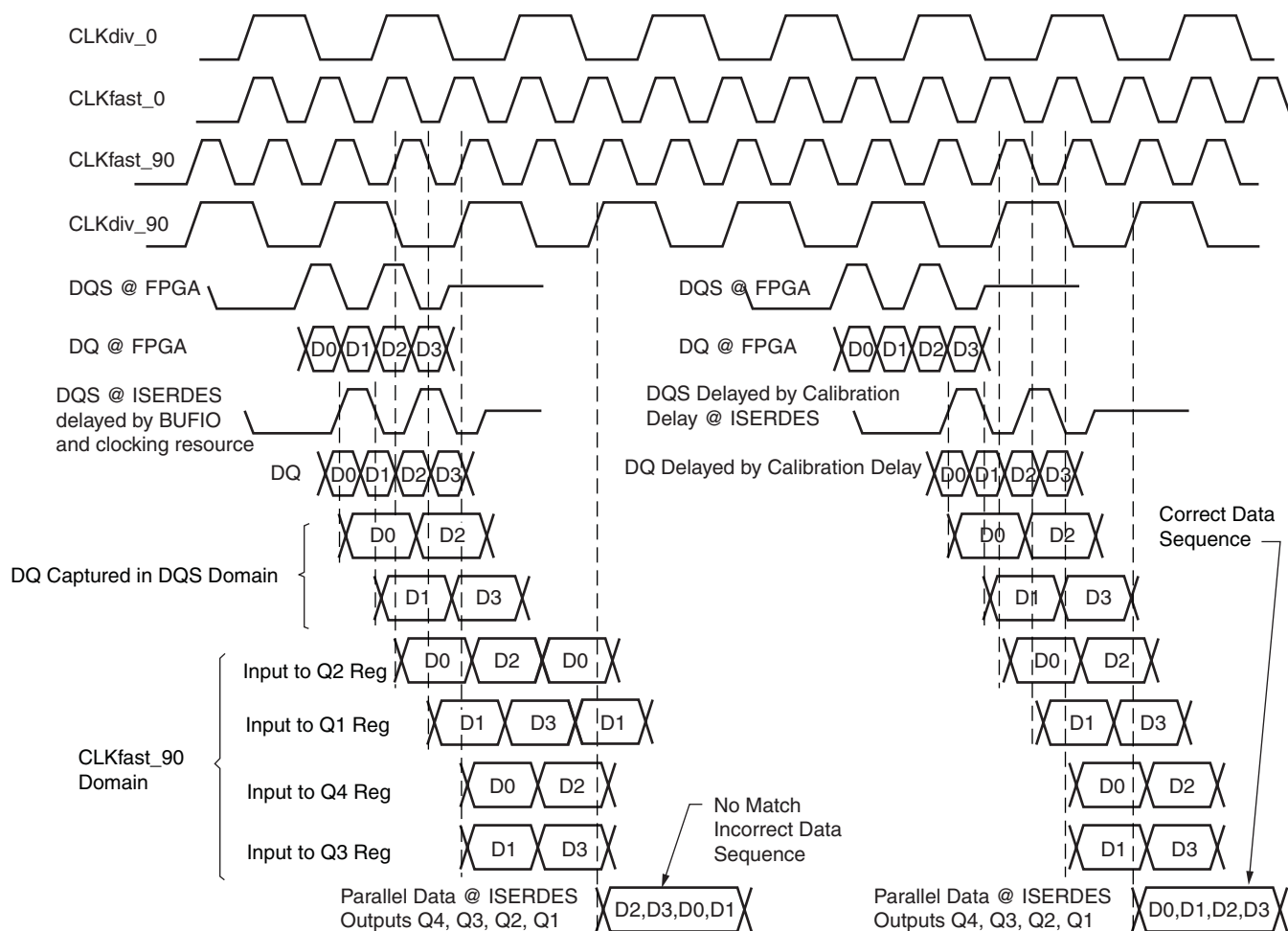
Notes:

1. $T_{\text{SAMP_BUFIO}}$ is the sampling error over VT for a DDR input register in the IOB when using the BUFIO clocking resource and the IDELAY.
2. All the parameters listed above are uncertainties to be considered when using the per bit calibration technique.
3. Parameters like BUFIO skew, package_skew, pcb_layout_skew, and part of T_{DQSQ} , and T_{QHS} are calibrated out with the per bit calibration technique. Inter-symbol interference and crosstalk, contributors to dynamic skew, are not considered in this analysis.

Per Bit Deskew Data Capture Technique

To ensure reliable data capture in the OCLK and CLKDIV domains in the ISERDES, a training sequence is required after memory initialization. The controller issues a WRITE command to write a known data pattern to a specified memory location. The controller then issues back-to-back read commands to read back the written data from this specified location. The DQ bit 0 ISERDES outputs Q1, Q2, Q3, and Q4 are then compared with the known data pattern. If they do not match, DQ and DQS are delayed by one tap, and the comparison is performed again. The tap increments continue until there is a match. If there is no match even at tap 64, then DQ and DQS are both reset to tap 0. DQS tap is set to one, and both DQS and DQ are delayed in unit tap increments and the comparison is performed after each tap increment until a match is found. With the first detected match, the DQS window count is incremented to 1. DQS continues to be delayed in unit tap increments until a mismatch is detected. The DQS window count is also incremented along with the tap increments to record the width of the data valid window in the FPGA clock domain. DQS is then decremented by half the window count to center DQS edges in the center of the data valid window. With the position of DQS fixed, each DQ bit is then centered with respect to DQS. The `dp_dly_slct_done` signal is asserted when the centering of all DQ bits associated with its DQS is completed.

Figure 8 shows the timing waveform for read data and strobe delay determination. The waveforms on the left show a case where the DQS is delayed due to BUFIO and clocking resource, and the ISERDES outputs do not match the expected data pattern. The waveforms on the right show a case where the DQS and DQ are delayed until the ISERDES outputs match the expected data pattern. The lower end of the frequency range useful in this design is limited by the number of available taps in the IDELAY block, the PCB trace delay, and the CAS latency of the memory device.



X721_08_112905

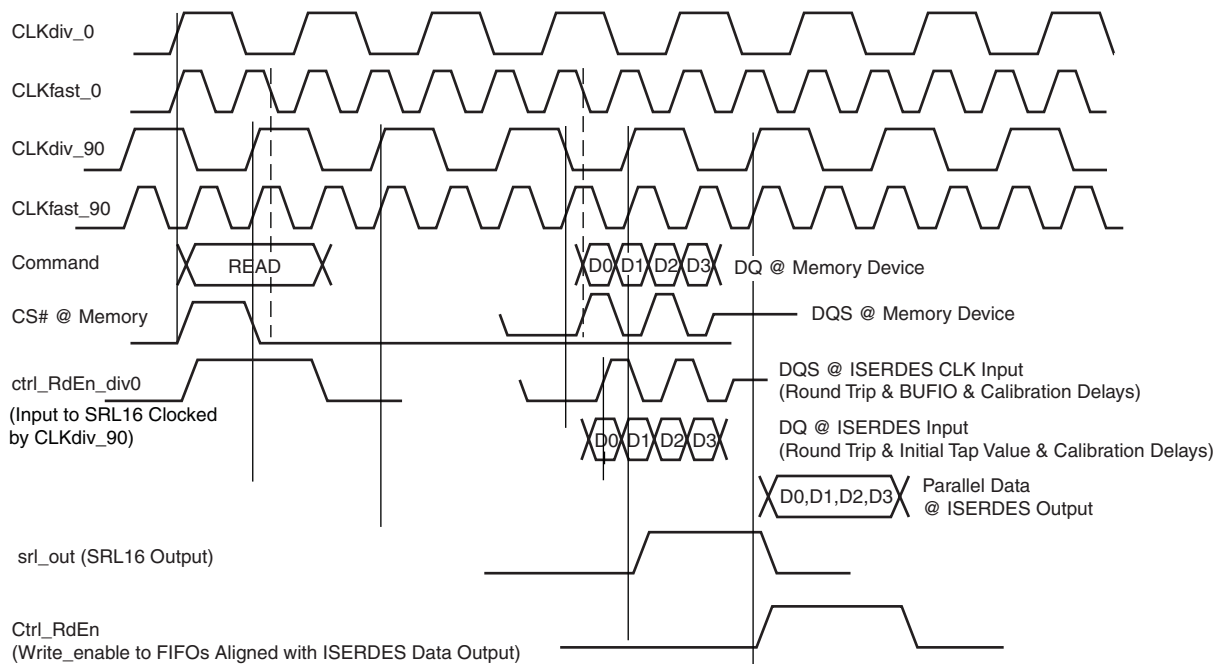
Figure 8: Read Data and Strobe Delay

Controller to Read Datapath Interface

Table 4 lists the control signals between the controller and the read datapath.

Table 4: Signals between Controller and Read Datapath

Signal Name	Signal Width	Signal Description	Notes
ctrl_Dummyread_Start	1	Output from the controller to the read datapath. When this signal is asserted, the strobe and data calibration begin.	This signal must be asserted when valid read data is available on the data bus. This signal is deasserted when the dp_dly_slct_done signal is asserted.
dp_dly_slct_done	1	Output from the read datapath to the controller indicating the strobe and data calibration are complete.	This signal is asserted when the data and strobe have been calibrated. Normal operation begins after this signal is asserted.
ctrl_RdEn_div0	1	Output from the controller to the read datapath used as the write enable to the read data capture FIFOs.	This signal is asserted for one CLKdiv_0 clock cycle for a burst length of 4 and two clock cycles for a burst length of 8. The CAS latency and additive latency values determine the timing relationship of this signal with the read state. Figure 9 shows the timing waveform for this signal with a CAS latency of 5 and an additive latency of 0 for a burst length of 4.



X721_09_113005

Figure 9: Read-Enable Timing for CAS Latency of 5 and Burst Length of 4

The ctrl_RdEn signal is required to validate read data because the DDR2 SDRAM devices do not provide a read valid or read-enable signal along with read data. The controller generates this read-enable signal based on the CAS latency and the burst length. This read-enable signal is input to an SRL16 (LUT-based shift register). The number of register stages required to align the read-enable signal to the ISERDES read data output is determined during calibration. One read-enable signal is generated for each data byte. Figure 10 shows the read-enable logic block diagram.

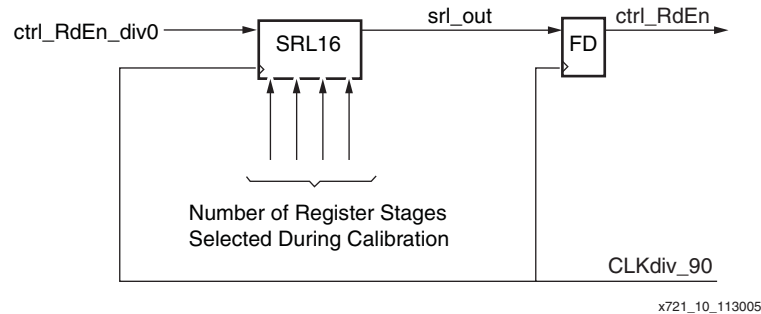


Figure 10: Read-Enable Logic

Reference Design

Figure 11 shows the hierarchy of the reference design. The mem_interface_top is the top-level module. This reference design is available on the Xilinx website at: <http://www.xilinx.com/bvdocs/appnotes/xapp721.zip>.

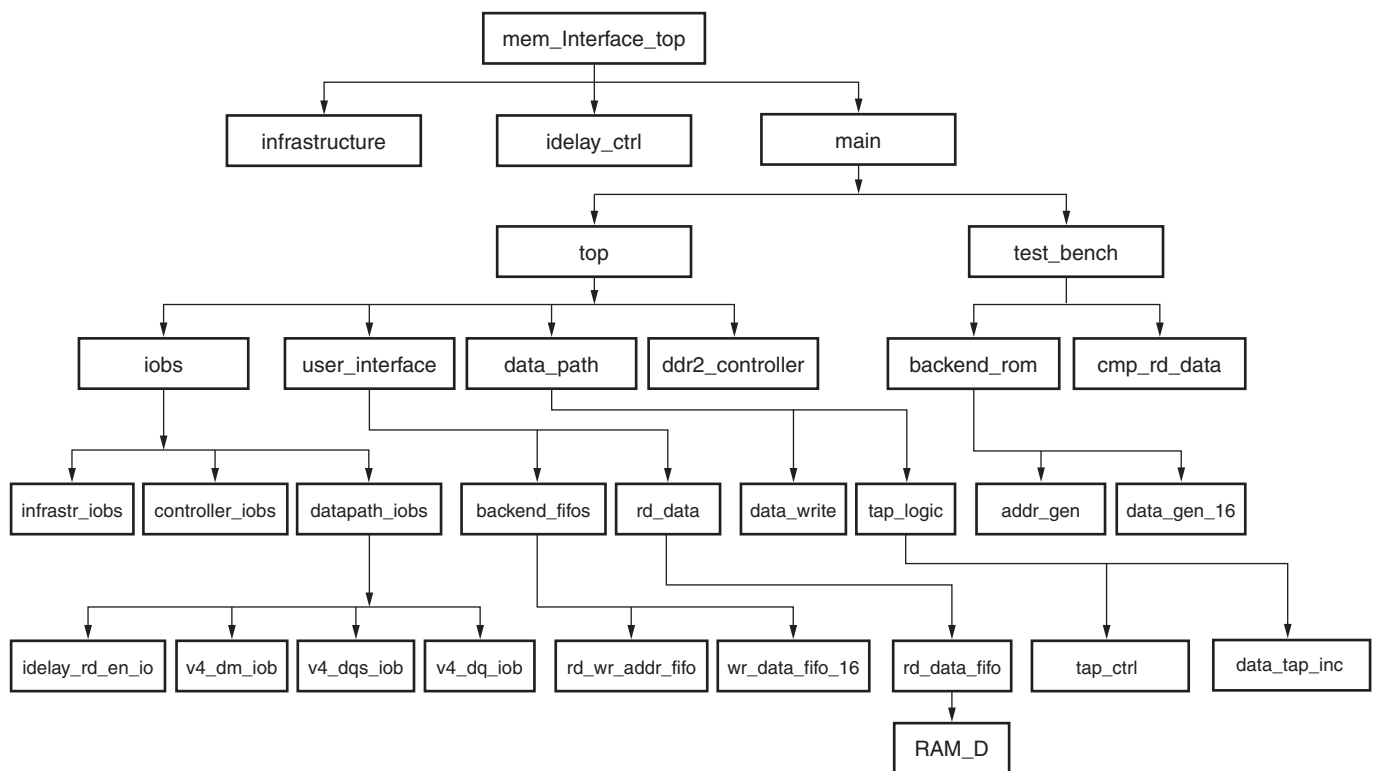


Figure 11: Reference Design Hierarchy

Reference Design Utilization

[Table 5](#) lists the resource utilization for a 64-bit interface including the physical layer, the controller, the user interface, and a synthesizable test bench.

Table 5: Resource Utilization for a 64-Bit Interface

Resources	Utilization	Notes
Slices	5861	Includes the controller, synthesizable test bench, and the user interface.
BUFGs	6	Includes one BUFG for the 200 MHz reference clock for the IDELAY block.
BUFIOs	8	Equals the number of strobes in the interface.
DCMs	1	
PMCDs	2	
ISERDES	64	Equals the number of data bits in the interface.
OSERDES	88	Equals the sum of the data bits, strobes, and data mask bits.

Conclusion

The data capture technique explained in this application note using ISERDES provides a good margin for high-performance memory interfaces. The high margin can be achieved because all the logic in the FPGA fabric is clocked at half the frequency of the interface, eliminating critical paths.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/15/05	1.0	Initial Xilinx release.
12/20/05	1.1	Updated Table 1 .
01/04/06	1.2	Updated link to reference design file.
02/02/06	1.3	Updated Table 4 .



Signal Integrity for High-Speed Memory and Processor I/O

SI20000-6-ILT (v1.0)

Course Specification

Course Description

Learn how signal integrity techniques are applicable to high-speed interfaces between Xilinx FPGAs and semiconductor memories. This course teaches you about high-speed bus and clock design, including transmission line termination, loading, and jitter. You will work with IBIS models and complete simulations using CAD packages. Other topics include managing PCB effects and on-chip termination. This course balances lecture modules and practical hands-on labs.

Level – Intermediate

Course Duration – 2 days

Price – \$1000 USD or 10 training credits

Course Part Number – SI20000-6-ILT

Who Should Attend? – Digital designers, board layout designers, or scientists, engineers, and technologists seeking to implement Xilinx solutions. Also end users of Xilinx products who want to understand how to implement high-speed interfaces without incurring the signal integrity problems related to timing, crosstalk, and overshoot or undershoot infractions.

Prerequisites

- Xilinx FPGA design experience preferred (equivalent of *Fundamentals of FPGA Design* course)

Software Tools

- Mentor Graphics HyperLynx®
- Cadence SPECCTRAQuest®

After completing this comprehensive training, you will have the necessary skills to:

- Identify when signal integrity is important and relevant
- Interpret an IBIS model and correct common errors
- Apply appropriate transmission line termination
- Understand the effect loading has on signal propagation
- Mitigate the impact of jitter
- Manage a memory data bus
- Understand the impact of selecting a PCB stackup
- Differentiate between on-chip termination and discrete termination

Course Outline

Day 1

- Introduction
- Transmission Lines
- **Mentor or Cadence Lab 1**
- IBIS Models
- **Mentor or Cadence Lab 2**
- **Mentor or Cadence Lab 3**
- High-Speed Clock Design
- **Mentor or Cadence Lab 4**
- SRAM Requirements
- **Mentor or Cadence Lab 5**

Day 2

- Physical PCB Structure
- On-Chip Termination
- SDRAM Design
- **Mentor Lab 6**
- Managing an Entire Design

Lab Descriptions

Note: Labs feature the Mentor Graphics or Cadence flow. For private training, please specify your flow to your registrar or sales contact. For public classes, flow will be determined by the instructor based upon class feedback.

- **Mentor Lab 1:** Opening the appropriate Mentor simulator
- **Mentor Lab 2:** Hands-on signal integrity observation of reflection and propagation effects
- **Mentor Lab 3:** Using an IBIS simulator to study basic transmission line effects
- **Mentor Lab 4:** Using saved simulation information to perform power calculation. Also, additional clock simulations
- **Mentor Lab 5:** Observing the effects of coupling on transmission lines
- **Mentor Lab 6:** Demonstrating how an SDRAM module can be handled with an EBD model
- **Cadence Lab 1:** Opening the appropriate Cadence simulator
- **Cadence Lab 2:** Analysis of a simple clock net
- **Cadence Lab 3:** Signal integrity effects caused by multidrop clock networks
- **Cadence Lab 4:** Crosstalk analysis
- **Cadence Lab 5:** Address and data analysis

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit www.xilinx.com/education, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "High-Speed" in the Training Catalog at <https://xilinx.onsaba.net/xilinx>.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.



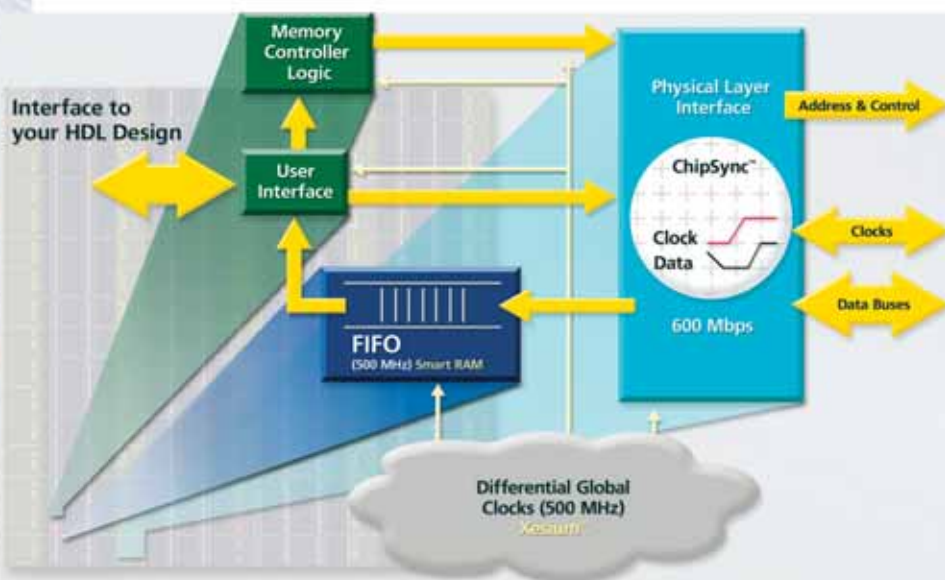
Virtex-4 Memory Interfaces

Hardware-Proven Solutions

Building interfaces to high-performance memory devices presents challenges such as high-speed synchronous data capturing, along with implementing complex physical-layer interfaces and control logic.

Virtex™-4 FPGAs solve these challenges with advanced silicon capabilities, including ChipSync™ source-synchronous technology, Xesium™ clocking, and Smart RAM. These features deliver the performance you need, while using minimal logic fabric resources.

To shrink design time, Xilinx provides expert guidance in the form of free, hardware-verified reference designs, application notes, user-friendly tools, and advanced development systems. This combination of unique silicon capabilities and comprehensive support enables you to build and verify robust memory interfaces quickly and easily.



High-Performance, Cost-Effective Solutions to Simplify Memory Interface Design

Optimize Your Design with Unique Built-in Silicon Features

- Revolutionary ChipSync technology provides 80 ps resolution for clock-to-data alignment, ensuring reliable data capture
- 500 MHz Xesium differential global clocks minimize skew and jitter, providing increased design margins
- 500 MHz Smart RAM blocks have built-in FIFO functionality, minimizing the design size
- Column-based I/O eliminates memory interface placement restrictions, alleviating board congestion

Finish Faster Using Proven Memory Interface Designs

- Comprehensive suite of free hardware-verified reference designs for high-speed memory interfaces (DDR2, DDR, QDR II, RLDRAM II, FCRAM II)
- Choose between fully synthesizable VHDL and Verilog reference designs

Customize with Ease Using Hardware Development Board

- Implement modifications with user-friendly tools
- Verify customizations with an advanced development board created specifically for memory interface design



Proven Memory Interface Solutions

Xilinx offers a complete suite of resources to support you in building and testing memory interfaces:

- **Advanced Memory Development System** – Develop and verify your design in hardware
- **Free Reference Designs and Detailed Application Notes**
- **Memory Interface Generator** – Customize your design with a simple menu-based tool
- **ChipScope™ Pro** – Debug, analyze and verify your design in real-time
- **Memory Corner Web Resources**
- **Education Services** – Increase your skill set and reduce your time-to-knowledge

Advanced Memory Development System



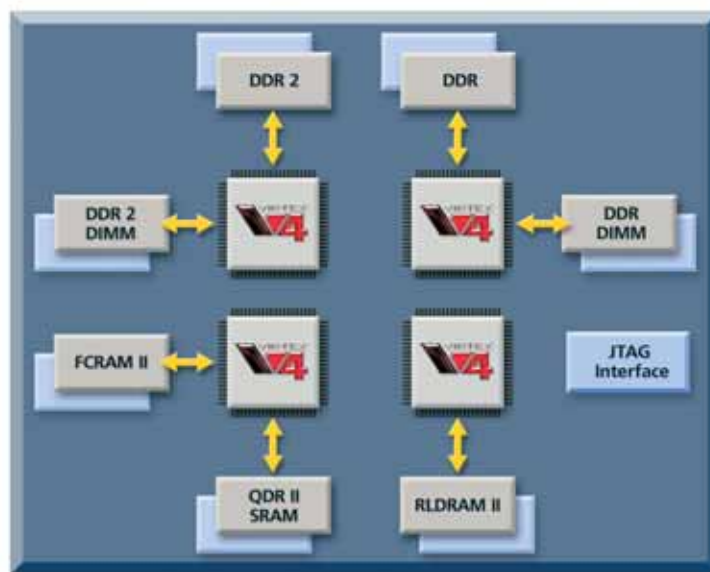
Memory Interface Generator



Memory System Features

Parameter	DDR2 SDRAM	DDR SDRAM	QDR II	RLDRAM II	FCRAM II
Data Rate	534 Mbps	400 Mbps	1.2 Gbps	600 Mbps	600 Mbps
CLK Rate	267 MHz	200 MHz	300 MHz	300 MHz	300 MHz
Data Width	144 bit (DIMM) 28 bit	144 bit (DIMM) 28 bit	(72+72) bit	36 bit	36 bit
I/O Standard	SSTL 18	SSTL 2	HSTL	HSTL	SSTL 18

ML461 Advanced Memory Development System Block Diagram



Take the Next Step

For the latest information or to find out more about our complete Virtex-4 FPGA memory solutions, visit www.xilinx.com/virtex4



Corporate Headquarters
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
Web: www.xilinx.com

European Headquarters
Xilinx
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan
Xilinx, K.K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific
Xilinx Asia Pacific Pte. Ltd.
No. 3 Changi Business Park Vista, #04-01
Singapore 486051
Tel: (65) 6544-8999
Fax: (65) 6789-8886
RCB no: 20-0312557-M
Web: www.xilinx.com

Distributed By:

FORTUNE 2005
500 BEST COMPANIES TO WORK FOR

XILINX
The Programmable Logic Company®

© 2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

WHAT'S NEW



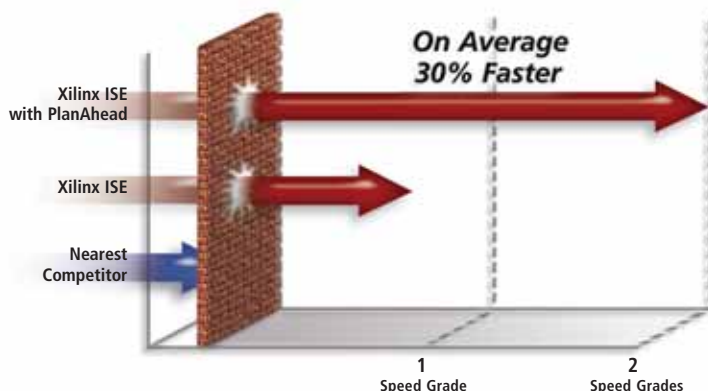
To complement our flagship publication *Xcell Journal*, we've recently launched three new technology magazines:

- *Embedded Magazine*, focusing on the use of embedded processors in Xilinx® programmable logic devices.
- *DSP Magazine*, focusing on the high-performance capabilities of our FPGA-based reconfigurable DSPs.
- *I/O Magazine*, focusing on the wide range of serial and parallel connectivity options available in Xilinx devices.

In addition to these new magazines, we've created a family of Solution Guides, designed to provide useful information on a wide range of hot topics such as *Broadcast Engineering*, *Power Management*, and *Signal Integrity*. Others are planned throughout the year.



READY! SET! GO WITH PLANAhead PERFORMANCE!



Based on benchmark data from a suite of 15 real-world customer designs targeting Xilinx and competing FPGA Solutions.



Two speed grades faster with PlanAhead software and Virtex-4

With our unique PlanAhead software tool, and our industry-leading Virtex-4 FPGAs, designers can now achieve a new level of performance. For complex, high-utilization, multi-clock designs, no other competing FPGA comes close to the Virtex-4 PlanAhead advantage:

- 30% better logic performance on average = 2 speed grade advantage
- Over 50% better logic performance for complex multi-clock designs

MEET YOUR TIMING BUDGETS ... BEAT YOUR COMPETITION TO MARKET

Meeting timing budgets is the most critical issue facing FPGA designers*. Inferior tools can hit a performance barrier, impacting your timing goals, while costing you project delays and expensive higher speed grades. To maximize the Virtex-4 performance advantage, the new PlanAhead software tool allows you to quickly analyze, floorplan, and improve placement and timing of even the most complex designs. Now, with ISE and PlanAhead you can meet your timing budgets *and* reduce design iterations, all within an easy-to-use design environment.

Download a free eval today at www.xilinx.com/planahead, view the TechOnline web seminar, and prevent your next FPGA design from stalling.

* CMP: June 2005 FPGA EDA Survey



The Programmable Logic Company™

www.xilinx.com/planahead



View The
TechOnline
Seminar Today

BREAKTHROUGH PERFORMANCE AT THE LOWEST COST

©2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Memory Interfaces

Solution Guide



"As designers of high-performance systems labor to achieve higher bandwidth while meeting critical timing margins, one consistently vexing performance bottleneck is the memory interface."

www.xilinx.com/xcell/memory1/

Published by



The Programmable Logic Company™

Corporate Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
Web: www.xilinx.com

European Headquarters

Xilinx
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan

Xilinx, K.K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific

Xilinx, Asia Pacific Pte. Ltd.
No. 3 Changi Business Park Vista, #04-01
Singapore 486051
Tel: (65) 6544-8999
Fax: (65) 6789-8886
RCB no: 20-0312557-M
Web: www.xilinx.com

Distributed By:

FORTUNE 2005
100 BEST COMPANIES TO WORK FOR

© 2006 Xilinx Inc. All rights reserved. The Xilinx name is a registered trademark; CoolRunner, Virtex, Spartan, Virtex-II Pro, RocketIO, System ACE, WebPACK, HDL Benchler, ChipScope, LogiCORE, AllianceCORE, MicroBlaze, and PicoBlaze are trademarks; and The Programmable Logic Company is a service mark of Xilinx Inc. PowerPC is a trademark of International Business Machines Corporation in the United States, or other countries, or both. All other trademarks are the property of their owners.