XAPP688 (v1.2) May 3, 2004

Creating High-Speed Memory Interfaces with Virtex-II and Virtex-II Pro FPGAs

Author: Nagesh Gupta, Maria George

Summary

Designing high-speed memory interfaces is a challenging task. Xilinx has invested time and effort to make it simple to design such interfaces using the Virtex-II™ and Virtex-II Pro™ FPGAs. This application note discusses the challenges presented by this task together with various techniques that can be used to overcome them, while illustrating the key concepts in implementing any memory interface. All examples used in this application note assume a DDR-1 interface on an XC2VP20FF1152-6 Virtex-II Pro FPGA. The interface speed is 200 MHz.

Introduction

High-speed memory interfaces are typically source-synchronous and double-data rate. In a source-synchronous design, the clocks are generated and transmitted along with the data, as shown in Figure 1. The data is typically received using the received clock and then transferred into the receivers' clock domain.

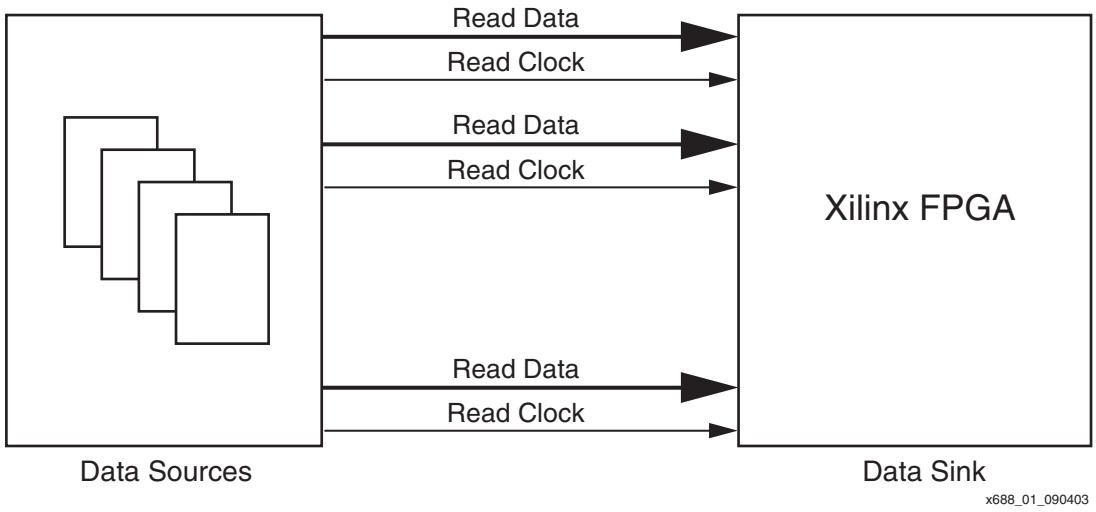

Figure 1: **Source Synchronous Memory Interface**

A memory interface can be modularly represented as shown in Figure 2. Creating a modular interface has many advantages. It allows designs to be ported easily. It also makes it possible to share parts of the design across different types of memory interfaces.

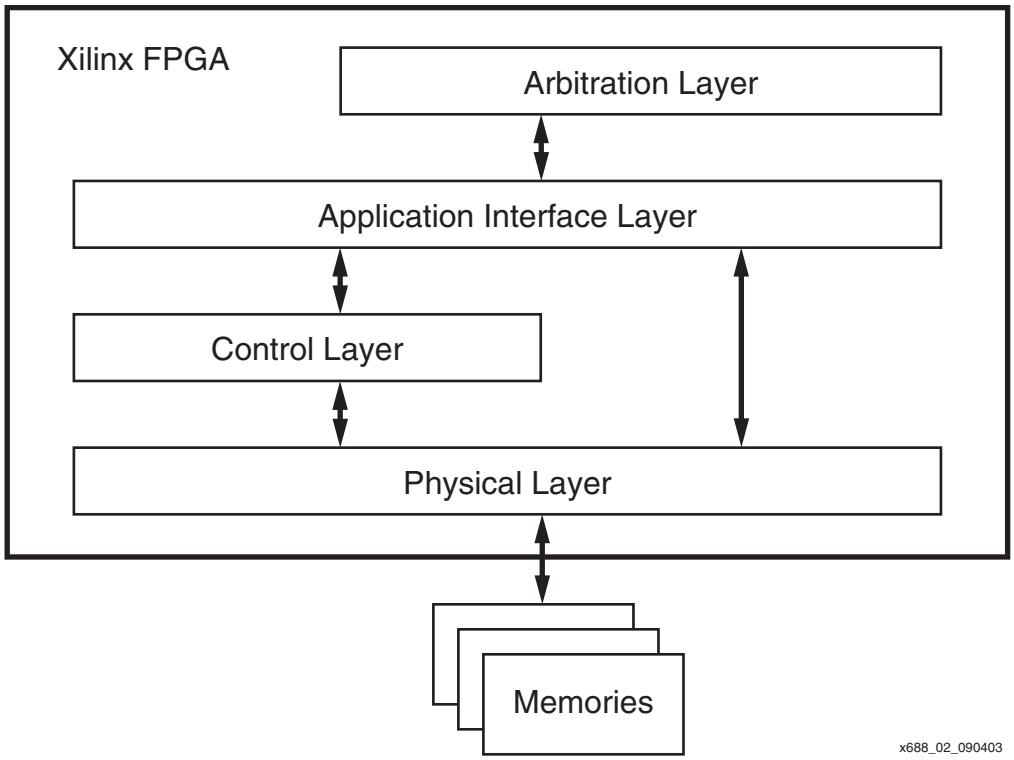

Figure 2: **Modular Memory Interface Representation**

Key Challenges

High-speed controllers and interfaces are challenging to design. Designing high-speed memory interfaces are particularly challenging due to various factors. Some of the key challenges are

- Source-synchronous data transmit (data write function).
- Source-synchronous data receive (data read function).

While these functions are common in all high-speed interfaces, memory interfaces make it particularly challenging due to the following factors:

- Single-ended standards. Memories typically use HSTL or SSTL type input/output standard.
- Non-free-running clocks. Some memories such as DDR SDRAM provide a non free-running strobe clock for data reads.
- Timing parameters. The input and output timing specifications for memories take away significant amount of the available data valid window.

We have solved these key problems with patent-pending innovations. These innovations have been proven in hardware and are available for use in the form of reference designs.

Physical Layer

The Physical layer is responsible for transmitting and receiving all the signals to and from the memories. The major functions of the physical layer include:

- Write data into the memory.
- Read data from the memory.
- Provide all the necessary control signals
- Transfer the read data clock domain from memory domain to FPGA domain.

The physical layer constitutes a major challenge in memory interface design. This section elaborates on the different aspects of the physical layer.

Clocks, Address and Control Signals

The FPGA generates all the clocks and control signals for reads and writes to memory. The memory clocks are typically generated using a Double Data Rate (DDR) register.

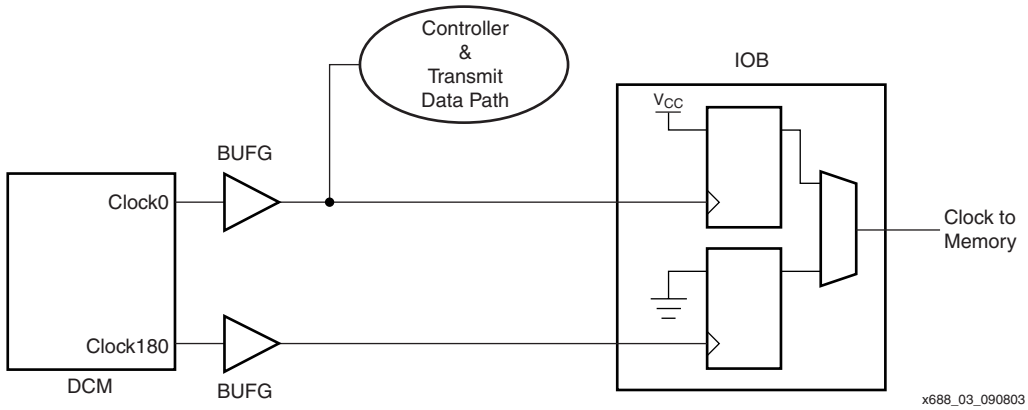

Figure 3: **Clock Generation for Memory Device**

As shown in Figure 3, a Digital Clock Manger (DCM) generates the clock and its inverted version. Generating the clock this way has a couple advantages:

- The data, control and clock signals all go through similar delay elements while exiting the FPGA.
- Clock duty cycle distortion is minimal when global clock nets are used for the clock and the 180° phase-shifted clock.

All of the address and control signals are registered and output at the IOB. The address and control signals are registered using a clock that is 180° shifted from the clock signal to the memory. Such an approach enables the address and control signals to have additional time margin before they are registered. The address and control signals meet the required timing with ease. An example timing analysis for a DDR-1 interface implemented using an XC2VP20FF1152 FPGA, -6 speed grade, is shown inTable 1.

Table 1: **Address and Control Signal Margins**

Parameter	Value	Leading-Edge Uncertainties	Trailing-Edge Uncertainties	Meaning
T_{CLOCK}	5000			Clock period
T_{CLOCK_SKEW}	50	50	50	Minimal skew, since right/left sides are being used and the bits are close together
$T_{PACKAGE\_SKEW}$	65	65	65	Using same bank reduces package skew
T_{SETUP}	750	750	0	Setup time from memory data sheet
T_{HOLD}	750	0	750	Hold time from memory data sheet
$T_{PCB\_LAYOUT\_SKEW}$	50	50	50	Skew between layout lines on the board
$T_{PHASE\_OFFSET\_ERROR}$	140	140	140	Offset between different phases of the clock. Taken from the parameter CLKOUT_PHASE
$T_{DUTY\_CYCLE\_DISTORTION}$	0	0	0	Duty-cycle distortion does not apply
T_{JITTER}	0	0	0	Since the clock and address are generated using the same clock, the same jitter exists in both. Therefore, it does not need to be included
Total Uncertainties		1055	1055	
Command Window	2500	1445	3945	Worst-case window of 2500 ps

Figure 4 illustrates the address and control signal margin.

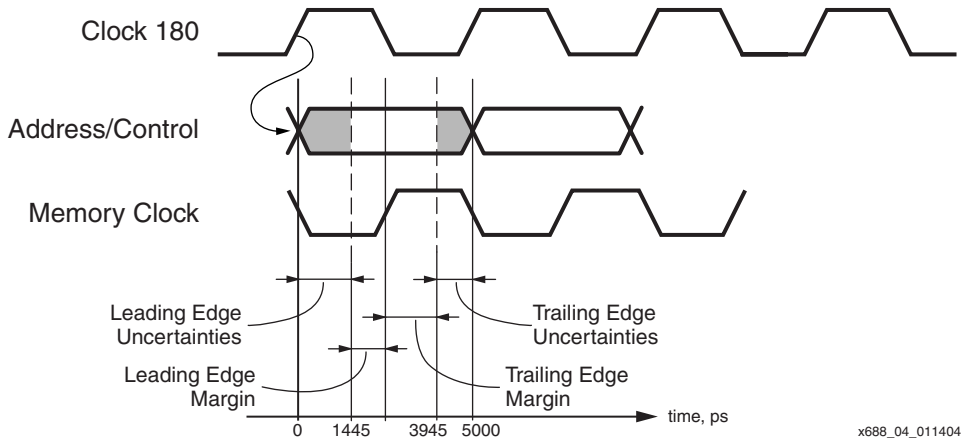

Figure 4: **Address and Control Signal Margins**

Data Write

The write data and strobe are clocked out of the FPGA. The strobe is center-aligned with respect to the data. For DDR-1, the strobe is required to be non-free-running. A strobe for DDR-1 is required only when valid data is being written into the memory. In addition, the DDR-1 strobe is bi-directional. The same strobe is used for reads and writes.

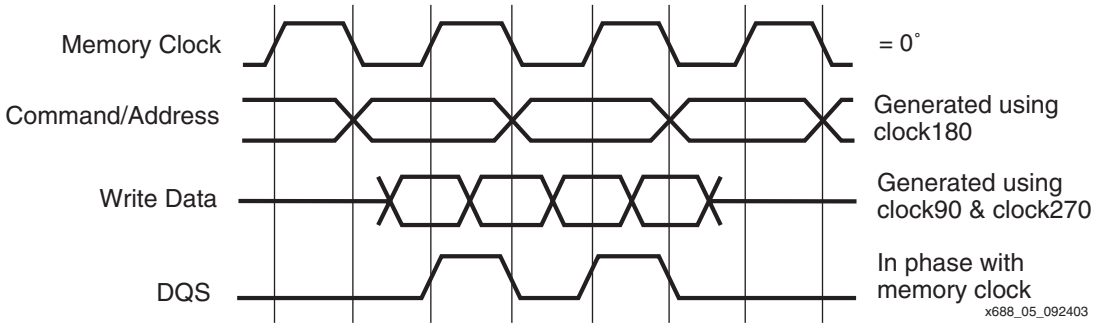

Figure 5: **Write Data, Command, Address, Strobe, and Clock Signals**

To meet the requirements specified above, the write data is clocked out using a clock that is 90 degrees and 270 degrees shifted from the primary clock going to the memory. The DDR registers in the IOB are used to clock the write data out. This is shown in Figure 6.

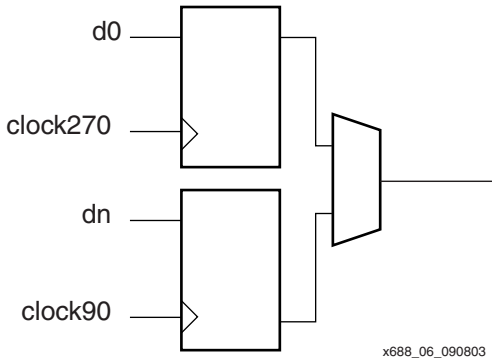

Figure 6: **Using IOB DDR Registers to Drive the Write Data**

Table 2 shows a calculation of the extra margin available for write data capture using the techniques just described. An illustration of the calculated time margin window is shown in Figure 7.

Table 2: **Write Data Timing Margin Calculation**

Parameter	Value	Leading-Edge Uncertainties	Trailing-Edge Uncertainties	Meaning
T_{CLOCK}	5000			Clock period
T_{CLOCK_PHASE}	2500			Clock phase
T_{DCD}	250			Duty cycle distortion of clock to memory
T_{DATA_PERIOD}	2250			Total data period, $T_{CLOCK\_PHASE} - T_{DCD}$
T_{CLOCK_SKEW}	50	50	50	Minimal skew, since right/left sides are being used and the bits are close together
$T_{PACKAGE\_SKEW}$	65	65	65	Skew due to package pins and board layout. This can be reduced further with tighter layout
T_{SETUP}	450	450	0	Setup time from memory data sheet
T_{HOLD}	450	0	450	Hold time from memory data sheet
$T_{PCB\_LAYOUT\_SKEW}$	50	50	50	Skew between layout lines on the board
$T_{PHASE\_OFFSET\_ERROR}$	140	140	140	Offset error between different clocks from the same DCM. Taken from the parameter CLKOUT_PHASE
T_{JITTER}	0	0	0	The same DCM is used to generate the clock and data. Hence they jitter together
Total Uncertainties	1205	755	755	Worst case for leading and trailing can never happen simultaneously
Window	740	755	1495	Total worst-case window is 740 ps

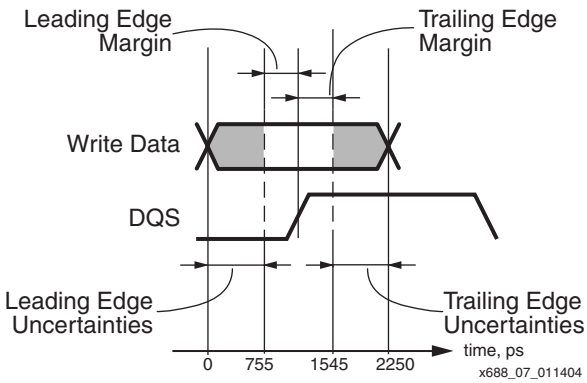

Figure 7: **Write Data Margin**

Strobe Delay Circuit

The strobe is delayed such that it falls within the data valid window. There are several mechanisms that can be used to delay the strobe.

1. Introduce an external board trace delay. This mechanism has been explained in earlier Xilinx Application Notes such as XAPP253.
2. Use a delay line on the board as opposed to a trace delay. This mechanism is similar to the external trace delay, but will use a delay line to create the required delay.

3. Delay a free-running strobe using the DCM. This mechanism can be used in memories such as QDR-2, FCRAM-2 and RLDRAM-2. The disadvantage of this mechanism is it requires additional DCM and clocking resources for each strobe. A QDR-2 interface using this mechanism is explained in XAPP 262.
4. Use an internal delay element with precisely controlled delays. An internal delay element has been built of LUTs(Look Up Tables) and other resources in the Xilinx FPGA fabric. Selecting the elements and all routing resources used within the elements defines the delay values precisely.

Methods (3) and (4) above are the preferred methods. Method (3) cannot be used if the FPGA is short on clocking resources or if a free-running strobe is not available. Method (4) has been demonstrated successfully for DDR-1 interface at 200 MHz.

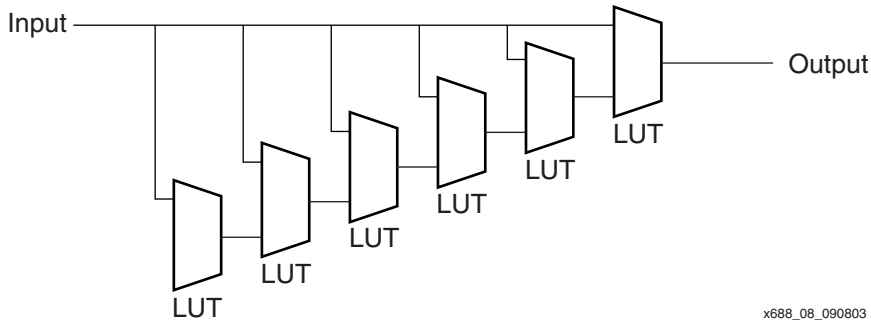

Figure 8: **Building Delay Elements Using LUTs in a Xilinx FPGA**

Figure 8 shows the way in which the LUTs are configured to create selectable delay elements. The delay value of each MUX element is between 200 and 300 picoseconds. The extra delay of the strobe with respect to data is calculated using a calibration circuit that ensures that the strobe is within the worst-case data valid window.

The total extra delay in the strobe path (compared to the data path) should fall within the valid data window. Calculation of the valid data window is described in data read section.

Data Read

Receiving the source-synchronous data from memory is the most interesting part of the memory interface. There are several novel techniques we have developed in this area.

Memory read data is edge aligned with a source-synchronous clock. In case of DDR-1, the clock is a non-free running strobe. The challenge is to receive the data using the strobe and transfer it to the FPGA clock domain. Since the memory clock is not free running, there can only be one stage of registers using this clock.

The input side of the data bits uses exactly the same resources as used by the input side of the strobe. This ensures matched delays on the strobe and data signals until the strobe is delayed in the strobe delay circuit. That is one of the reason why this design captures the data directly into registers in the FPGA fabric. A separate circuit transfers the output of these registers into FIFOs clocked using the FPGA clock.

Data Capture

Data is captured using the delayed strobe mentioned previously. Data is directly captured in the FPGA fabric slices. By using novel techniqies, it is possible to keep the captured data valid for two entire clock cycles. The timing calculation for data capture is shown in Table 3. An illustration of the calculated data valid window is shown in Figure 9.

Table 3: **Data Valid Window Calculation**

Parameter	Value	Leading-Edge Uncertainties	Trailing-Edge Uncertainties	Meaning
T_{CLOCK}	5000			Clock period
T_{PHASE}	2500			Clock phase
T_{MEM_DCD}	250			Duty cycle distortion from memory DLL
T_{DATA_PERIOD}	2250			Total data period, $T_{PHASE} - T_{MEM\_DCD}$
T_{DQSQ}	500	500	0	Strobe-to-data distortion
$T_{PACKAGE\_SKEW}$	65	65	65	This parameter depends on the exact package. Since the 8 data bits are close together, skew is less than this
T_{SETUP}	240	240	0	Setup time from Virtex-II Pro data sheet for -6 part. Taken from T_{DICK} parameter.
T_{HOLD}	–50	0	–50	Hold time from Virtex-II Pro data sheet. Taken from T_{CKDI} parameter.
T_{JITTER}	100	0	0	Data and strobe jitter together, since they are generated off the same clock
$T_{LOCAL\_CLOCK\_LINE}$	25	25	25	Observed skew is lower than this value, since loading is light and all bits are close together
$T_{PCB\_LAYOUT\_SKEW}$	50	50	50	Skew between data lines on the board
T_{QHS}	450	0	450	Hold skew factor for DQ
Uncertainties		880	540	
Window	830	880	1710	Worst-case window of 830 ps

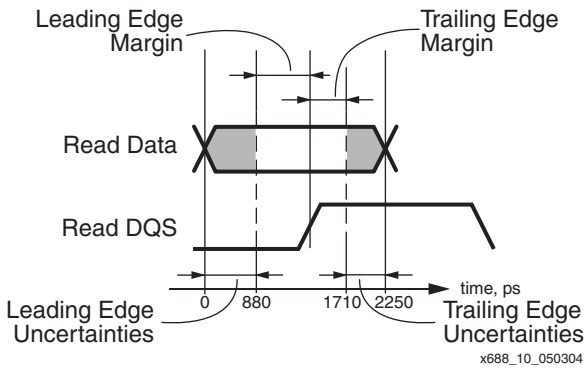

Figure 9: **Data Valid Window**

A timing diagram of data capture is shown in Figure 10.

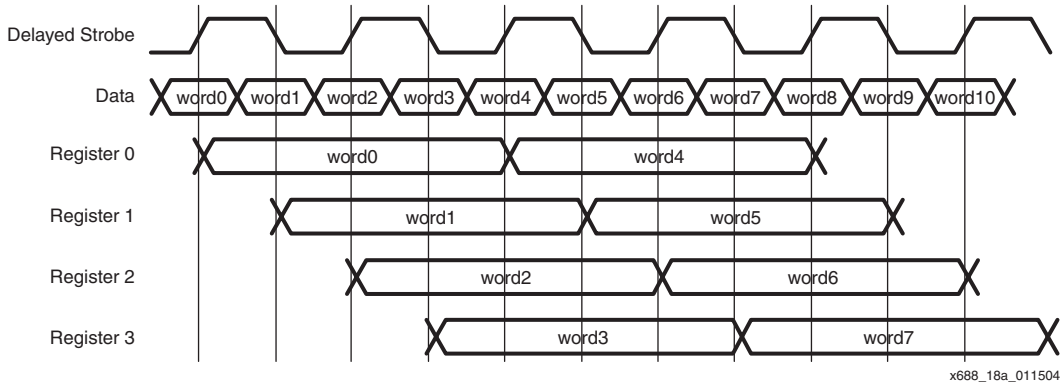

Figure 10: **Data Capture Timing**

Data Recapture

Data recapture is the process of transferring the data into the FPGA clock domain. The recapture circuit works for any system-level timing. No system-level calculations are required for the circuit to work. The extended period for which each piece of data is valid makes the task of recapture easier. The data is directly transferred from the data capture registers into separate FIFOs.

Delay Calibration Circuit

The delay calibration circuit selects the number of delay elements to be used to create extra delay through the strobe line. The delay calibration circuit calculates the delay of a circuit that is identical in all respects to the strobe delay circuit. All aspects of delay are considered for calibration. This includes all the component delays, as well as the route delays. The calibration circuit selects the number of delay elements to be used at any given time. After the required number of delay elements is determined, the calibration circuit asserts the appropriate select lines to the strobe delay circuit. The select lines of the strobe delay circuit can be changed only when no read data is being received.

Selecting the required number of delay elements as shown avoids any uncertainty due to process, voltage, and temperature (PVT) variance. The positive and negative hysterisys for transition from selecting one set of delay elements to another set is configurable. The number of contiguous selection transitions required to make the transition is also configurable.

During operation, the temperature of the die can cause the delay elements to get slower. If this happens, the number of selected delay elements changes automatically.

Control Layer and Application Interface Layer

The control and application layers are simple to design in the Xilinx FPGA fabric. The control layer is responsible to generate all control signals according to a memory protocol. The application interface layer is used to transfer data and commands between the user application and the memory interface blocks. A simple user application interface is as shown in Figure 11.

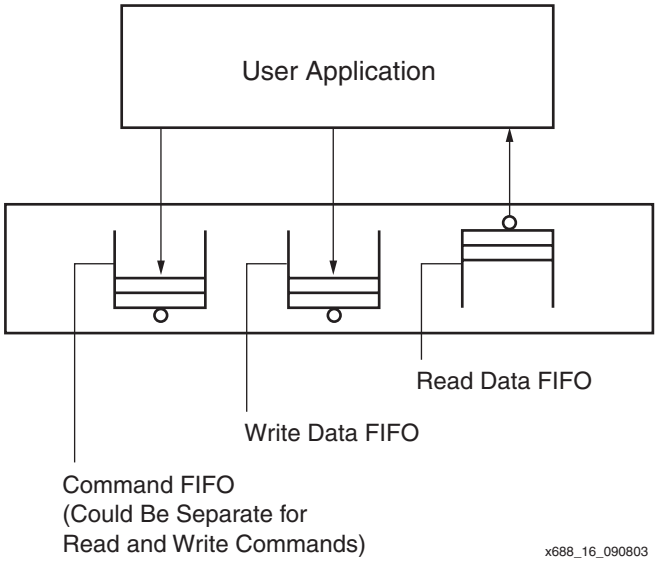

Figure 11: **User Application Interface**

Tools

Implementation of the interface described in this application note is quite intricate. There are several instances where very precise routing delays and placements are required. To enable customers to design effectively using the methods described, a tool is provided that works with various Virtex-II and Virtex-II Pro devices and generates all the required constraints for the design. The tool can also generate the recommended pinouts for the data bus. The constraints generated by the tool must be consistent with the RTL and the RTL hierarchy.

Implementation

This section describes various implementation considerations. Some differences exist between different memory types. The techniques described in this application are applicable in general to all of the different memory types. This section talks about the differences between different memories and how the techniques can be applied to different memories.

Board Design Considerations

Board design considerations for a memory interface are not very different from designing other high-speed interfaces. Several Xilinx application notes describe the intricacies of building high-speed boards. XAPP623 describes advanced techniques in power system design. Xilinx user guide UG060 describes a memory board designed for Virtex-II Pro using most of the techniques described in this application note.

DDR-1 SDRAM Interface

DDR-1 is arguably the most complex memory interface. All of the examples in this application note were for DDR-1. DDR-1 controller has been illustrated in other Xilinx application notes such as XAPP253.

QDR-2 SRAM Interface

QDR-2/DDR-2 SRAM is different in many aspects compared to DDR-1 SDRAM. This interface is described in detail in XAPP262. The controller is much simpler, since there is no need for separate row and column addresses.

The physical layer is simpler since these memories have a contiguous or free-running strobe. A contiguous strobe can be easily used to capture the data in multiple stages and finally into an asynchronous FIFO. One complexity is the lack of a data valid signal. The absence of a data valid signal leaves the state machine to predict when valid data will be available to register. The state machine predicts the return of valid data by counting the number of cycles after the read command is issued. However, the input and output buffer speeds of different FPGAs or different speed grades of FPGAs can potentially cause the data to return at different cycle boundaries. This reduces the maximum frequency of operation for this kind of interface.

Another novel innovation solves this problem, and makes it possible to determine the exact time when valid data will be received.

RLDRAM-2/FCRAM-2 Interfaces

Both RLDRAM-2 and FCRAM-2 interfaces are relatively simple. The controller design for either of these memories is comparable in complexity to QDR-2 controller. There are no separate row and column addresses to deal with. In addition, the strobes are contiguous and there is a separate valid signal that indicates the arrival of valid data.

Xilinx will publish an RLDRAM-2 interface application note shortly.

References

1. Xilinx Application Notes XAPP253, XAPP609, XAPP262, XAPP678C, and XAPP688C.
2. Micron data sheet for DDR-1 DIMMs, MT4VDDT1664 A.
3. Micron data sheets for DDR-1 MT46V16M16.

Conclusion

The innovative techniques described in this application note make it easy to design any memory interface with Xilinx Virtex-II and Virtex-II Pro FPGAs.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/15/04	1.0	Initial Xilinx release.
02/02/04	1.1	Removed Figure 12 (screenshot of unreleased tool).
03/01/04	1.1.1	Table 2: Corrected description for T_{SETUP} and T_{HOLD}.
05/03/04	1.2	• Table 1 and Table 2: Added derivation of parameter to "Meaning" column for $T_{PHASE\_OFFSET\_ERROR}$. • Table 3: Added derivation of parameters to "Meaning" column and revised parameter values for T_{SETUP} and T_{HOLD}. Recalculated derived values (Uncertainties and Window). • Figure 9: Modified data window parameters in accord with changes in Table 3.