How to build reliable FPGA memory interface controllers without writing your own RTL code!

Adrian Cosoroaba, Xilinx - April 19, 2006

As FPGA designers strive to achieve higher performance while meeting critical timing margins, one consistently vexing performance bottleneck is the memory interface. Today's more advanced FPGAs provide embedded blocks in every I/O that make the interface design easier and more reliable. These I/O elements are building blocks that, when combined with surrounding logic, can provide the designer with a complete memory interface controller. Nonetheless, these I/O blocks – along with extra logic – must be configured, verified, implemented, and properly connected to the rest of the FPGA by the designer in the source RTL code.

But, what if these difficult tasks were taken care of by the FPGA vendor? What if a designer could simply use a GUI to input the memory system parameters and generate RTL code without writing it from scratch? Finally, what if the physical layer interface was based on hardware verified designs? All this is now possible using the Memory Interface Generator (MIG) from Xilinx. This "How To" article will discuss the various memory interface controller design challenges and how to use the MIG to build a complete memory interface solution for your own application on a Virtex-4 FPGA.

Memory trends and design challenges
In the late 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 per pin (where "Mbps" stands for "megabits-per-second"). Present trends indicate that these rates are likely to double every four years, potentially reaching 1.6 Gbps/pin by 2010.

This trend presents a serious problem to designers in that the data valid window – that period within the data period during which Read Data can be reliably obtained – is shrinking faster than the data period itself. This is because the various uncertainties associated with system and device performance parameters, which impinge upon the size of the data valid window, do not scale down at the same rate as the data period.

This trend is readily apparent when comparing the data valid windows of the earlier-generation DDR SDRAMs running at 400 Mbps and the current DDR2 memory technology which runs at 667 Mbps. The DDR device with a 2.5 ns data period has a data valid window of 0.7 ns, while the DDR2 device with a 1.5 ns period has a mere 0.14 ns. Clearly, this accelerated erosion of the data valid window introduces a new set of design challenges for the FPGA designer that require a more effective means of establishing and maintaining reliable memory interface performance.

The challenge to keep pace with the increase in data rate is compounded by the expansion of the data buses employed by these high-performance memories. Wider buses at these speeds require more bandwidth, making chip-to-chip interfaces all the more challenging. The designer must
therefore resolve a new and more problematic set of signal integrity, I/O placement and board routing issues.

Along with the performance issues that attend the new breed of high-performance memories, the designer faces 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 which, for the FPGA designer, suggest the need for a new level of integration support from the tools that accompany the FPGA.

**Memory interface and controller design**

There are three fundamental building blocks that comprise a memory interface and controller for an FPGA-based design: the physical layer interface, the memory controller state machine, and the user interface that bridges the memory interface design to the rest of the FPGA design. Customer surveys reveal a consensus that the physical layer interface (comprising the Read and Write interface logic and I/O blocks) is one of the most challenging parts of the overall design.

Memory interface clocking requirements are typically more difficult to meet when reading from memory, as compared with writing to memory. This is because the 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, Digital Clock Manager (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.

**Traditional method for Read Data capture**

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. The designer inserts 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, different data strobe (DQS) signals and changing ambient system conditions (i.e., voltage, temperature) can easily create skew whereby the predetermined phase shift is ineffectual.

1. The traditional fixed delay Read Data capture method is prone to errors.

Traditional techniques have allowed FPGA designers to implement DDR SDRAM memory interfaces. But very high-speed 333-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 set up (leading-edge) and hold (trailing-edge) uncertainties further minimize the valid window available for reliable Read Data capture. For example, the 333-MHz (667 Mbps) DDR2 Read interface timings require FPGA clock alignment within a 0.2 ns window.

Other issues also demand the designer's attention, including chip-to-chip signal integrity, simultaneous switching constraints, and board layout constraints. Pulse-width distortion and jitter on the clock and/or data strobe signals also cause data and address timing problems at the input to the RAM and the FPGA's Input/Output Blocks (IOBs) flip-flops. Furthermore, as a bidirectional and non-free-running signal, the data strobe has an increased jitter component, unlike the clock signal. *Adaptive Clock-to-Data centering built into every I/O*

Virtex-4 FPGAs with dedicated delay and clocking resources in the I/O blocks – called ChipSync – 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 input clock to the Read Data upon system initialization. This proven methodology ensures optimum performance, reduces engineering costs, and increases design reliability.

ChipSync enables clock-to-data centering without consuming CLB resources. Designers can use the I/O logic to determine the exact phase relationship between clock to data. The Read Data is then delayed to center-align the input 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 input clock to the center of the Read Data.

2. Clock-to-Data centering using 75 ps tap delays. *(click this image to see a larger, more detailed version)*

Delaying the data by the appropriate number of taps aligns the center of the data window with the edge of the input clock. The tap delays generated by this precision delay block allow alignment of the data and clock to within 75 ps resolution.

**Signal integrity challenge**

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

The column-based architecture used in the Virtex-4 FPGA enables I/O, clock, power, and ground pins to be located anywhere on the silicon chip and 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. Additionally, the special packaging technology – known as SparseChevron – used in the Virtex-4 FPGA enables distribution of power and ground pins evenly across the package.
The benefit to designers is improved signal integrity. Finally, a differential DCM clock output that delivers extremely low-jitter performance necessary for very short data valid windows and diminishing timing margins ensures a robust memory interface design.

**Memory controller design and integration**

In addition to the challenge of designing the physical layer interface, integrating all of the building blocks - including the memory controller state machine - is essential for the completeness of one's design. Controller state machines vary with the memory architecture and system parameters. State machine code can also be complicated as it can be a function of many variables, such as:

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

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

The complete design can be generated with the MIG - a software tool freely available from Xilinx as part of the ISE CORE Generator suite of reference designs and IP. A depiction of the MIG design flow is presented in Fig 3.

![Memory Interface Generator (MIG) design flow](click this image to see a larger, more detailed version)

The designer uses the MIG's GUI (Fig 4) to set system and memory parameters. After selecting the FPGA device and speed grade, for example, the designer may select the memory architecture and pick the actual memory device or module. The same GUI provides selection of the bus width and clock frequency and also offers the option to have more than one controller for multiple memory bus interfaces. Other advanced options provide control of the clocking method and pin assignments.
Finally, the MIG tool generates the *.rtl and *.ucf files, which are the HDL code and constraints files, respectively. These files are generated from a library of hardware-verified designs.

The designer also has an additional option. Unlike other solutions that offer only "black box" implementations, this code is not encrypted, thereby providing the designer with complete flexibility to change and customize a design. The output files are categorized in modules that apply to different building blocks of the design: user interface, physical layer, controller state machine, etc. For example, one may wish to use the physical layer interface "as-is", but to customize the state machine that controls the bank access algorithm. After these optional code changes, the designer can perform additional simulations.

The next step is to import the files in the ISE project followed by synthesis, place and route, additional timing simulations (if needed), and - finally - verification in hardware.

**Conclusion**

Equipped with the right FPGA and software tools, memory interface controller design can be a reasonably pain-free process, even when attempting to access the added performance benefits of today's DDR2 SDRAM with the latest 667 Mbps speed grade. More information and details on these memory interfaces solutions can be found in the Memory section of the Xilinx website.

**Adrian Cosoroaba** joined Xilinx in 2004 as Marketing Manager for Virtex Solutions Marketing, responsible for worldwide marketing activities related to Virtex Solutions. Cosoroaba brings to Xilinx over 19 years of semiconductor experience in memory, microprocessor applications and marketing. Prior to joining Xilinx, Cosoroaba held a range of positions including Strategic Marketing and Applications Manager at Fujitsu Microelectronics. Cosoroaba holds a M.S. degree in electrical engineering from Ohio State University and a B.S. degree in engineering physics from University of California at Berkeley. He can be reached at adrian.cosoroaba@xilinx.com.