WP333 (v1.0) March 24, 2008

FIFOs in Virtex-5 FPGAs

By: Peter Alfke

In a First-In First-Out (FIFO) memory subsystem, data is written and retrieved in exactly the same order; the first data written into the memory is the first data read out of the memory. Explicit addressing is not required. Write and read operations are independent and can use unrelated clocks. In 1970, Fairchild Semiconductor introduced the first integrated circuit FIFO, the 3341.

From the user's perspective, a FIFO is an ideal memory subsystem because it is simple and easy to use. A well-designed FIFO memory never becomes full, and it only becomes empty when the last word is read from the buffer.

From the system designer's perspective, FIFO implementation can be complex and demanding. This paper explores some of these problems and their solutions.

Introduction

On the surface, designing a FIFO memory subsystem seems simple and straightforward because it consists of a Random Access Memory (RAM) with two independently clocked ports. One port is used for writing. The other port is used for reading. In addition, the FIFO has two independent address counters to steer write and read data.

Independently clocked means that there are no restrictions on the relationship between read and write clock frequency and phase. Independently clocked is also referred to as multi-rate or asynchronous FIFO operation. Plesiochronous or plesiosynchronous describes read and write clocks that are almost, but not precisely, synchronous or in-phase. The Virtex™-5 FIFO operates flawlessly with any phase or frequency relationship between the read and write clocks.

Even experienced designers can be challenged by the difficulty of reliably decoding and synchronizing the essential FIFO Empty and FIFO Full status signals. This is especially true when the FIFO uses two independent clocks that operate at several hundred MHz. Fast asynchronous design is notoriously difficult.

The Virtex-5 FPGA FIFO

To meet the challenge of fast, asynchronous FIFO design, the Virtex-5 family includes a dedicated, hard-coded FIFO controller inside each block RAM. The controller allows reliable FIFO operation at a clock rate of 500 MHz without using any extra logic.

The designer supplies 4-bit, 9-bit, 18-bit, or 36-bit parallel input data, a continuously running write clock, a write clock enable signal, a continuously running read clock, and a read clock enable signal. Output data is always the width of input data.

The FIFO Empty signal goes High as a result of the read clock when the last data entry is read out of the FIFO. The designer must disable reads until the FIFO Empty signal goes Low again.

The rising and the falling edges of the FIFO Empty signal are synchronous with the read clock, providing a totally synchronous interface. If the read clock enable signal is accidentally kept active after the FIFO is empty, a read error flag is asserted, but the FIFO contents and addressing are not affected.

FIFO Almost Empty and FIFO Almost Full are programmable status signals. These two signals can be used as a warning to slow the read or the write operation, or they can be used to indicate the data level in the FIFO.

Implementation Details

The FIFO controller has two binary counters that serve as block RAM address pointers. One counter points to the current read address. The other counter points to the current write address. The arithmetic difference between the output values of these counters equates to the quantity of data in the FIFO. When the FIFO is full or the FIFO is empty, these counters are pointing at the same address and their arithmetic difference is zero.

Generating reliable FIFO Full and FIFO Empty status signals requires quickly and accurately comparing the outputs of these counters, which do not typically share a common clock. Furthermore, binary counters can generate unacceptable glitches at the comparator output.

The solution to this problem is a Gray code counter. The simplest way to build a Gray code counter is to start with a binary counter, and then synchronously convert its binary output to Gray code. The binary address counter values can be used to calculate a programmable offset for generating FIFO Almost Full and FIFO Almost Empty.

Synchronization Issues

The assertion of FIFO Empty results from a read operation. Therefore, the leading edge of FIFO Empty is naturally synchronous with the read clock. Because the trailing edge of FIFO Empty results from a write operation, the trailing edge of FIFO Empty is synchronous with the write clock, which is the wrong clock domain. Moving the trailing edge of FIFO Empty to the read clock domain requires using flip-flops, which can introduce metastable delays.

The Virtex-5 family uses a conservative synchronizer design that has been demonstrated to work reliably at a 500 MHz clock rate. Xilinx conducted a week-long test at clock rates (asynchronous) of 200 MHz and 500 MHz. During this test, FIFO Empty was asserted more than 10^{14} times without a single failure.

The synchronizer design delays the trailing edge of FIFO Empty by a few read-clock periods, which is acceptable because it does not adversely affect performance. Using a similar method, the trailing edge of FIFO Full is synchronized to the write clock.

Summary

Today, mid-sized FIFOs are easily implemented in the Virtex-5 family. The dedicated FIFO controller inside every block RAM of every Virtex-5 FPGA uses no extra FPGA logic, and it insulates the designer from complex and demanding task of designing a reliable FIFO memory subsystem.

Revision History

The following table shows the revision history of this document.

Date	Version	Description of Revisions
03/24/08	1.0	Initial Xilinx release. Derived from a TechXclusive.

Notice of Disclaimer