FIFO Buffer Designs in The XC4000E/EX FPGA Families

Many XC4000 designs use the distributed RAM feature to implement First-In-First-Out (FIFO) elastic buffers to form a bridge between subsystems with different clock rates and access requirements. However, the non-synchronous nature of the single-port RAM in the original XC4000 architecture leaves the designer with several challenges. Addresses must be multiplexed, independent read and write clocks must be synchronized, and access requests must be arbitrated.

The synchronous dual-port option RAM in the new XC4000E and XC4000EX FPGAs eliminates most of these problems.

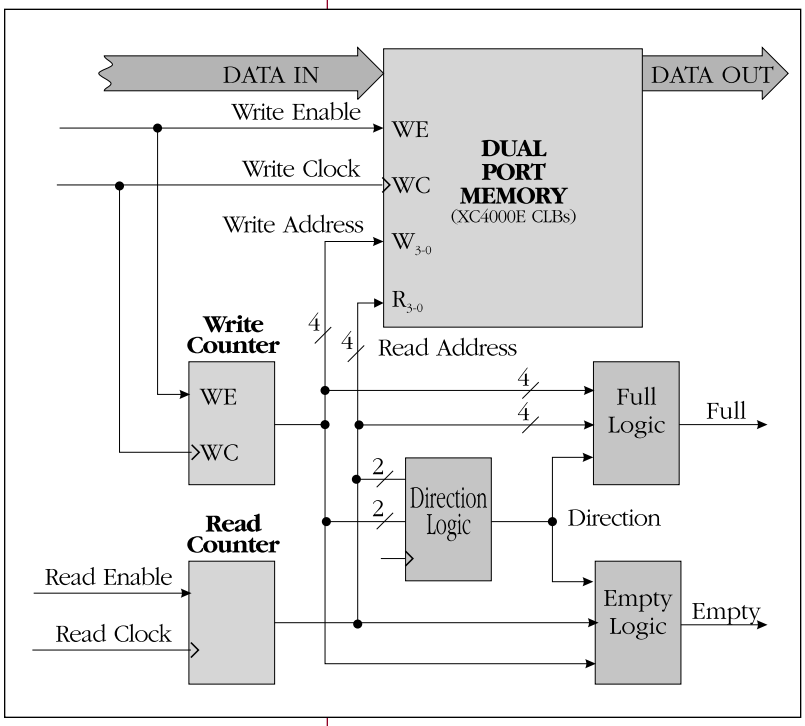


Figure 1 - 16x16 FIFO Block Diagram

Since the dual-port RAM in each CLB has independent write and read addresses, there is no need to multiplex addresses and arbitrate their selection. The synchronous write mechanism simplifies write timing and contributes to much faster operation. The FIFO design effort can now be concentrated on achieving high throughput and low cost, and on solving the fundamental timing problems created by asynchronous read and write clocks.

Synchronous and Asynchronous FIFOs is an application note now available on the Xilinx WebLINX World Wide Web site (http://www.xilinx.com). This application note describes six complete design examples of RAM-based FIFO designs using the dual-port RAM feature of the XC4000E and XC4000EX FPGAs. Three synchronous designs with a common read/write clock are described, as well as the corresponding three asynchronous designs with independent read and write clocks. Emphasis is on the fast, efficient and reliable generation of the handshake signals FULL and EMPTY that determine design performance.

The first design is a synchronous 16x16 FIFO buffer, where the depth of the basic CLB-RAM is sufficient. This leads to a very fast and efficient implementation that can run at, or close to, the maximum write speed of 70 MHz (-3 speed grade), even for simultaneous read and write operations. This is the simplest and fastest design because it avoids the more challenging issues of asynchronous clocking. The design occupies 23 CLBs.

Figure 1 shows the basic block diagram of the 16x16 FIFO buffer. In the synchronous version of this design, read and write clocks are identical. The 4-bit read counter and the 4-bit write counter (**Figure 2**) are implemented as two cascaded 2-bit Grey or Johnson counters. In a fully synchronous design, this choice is not mandatory, but it has advantages in the non-synchronous implementation. It is, however, mandatory that the upper two bits always stay constant for four consecutive counts. Therefore, Linear-Feedback-Shift-Register (LFSR) counters cannot be used.

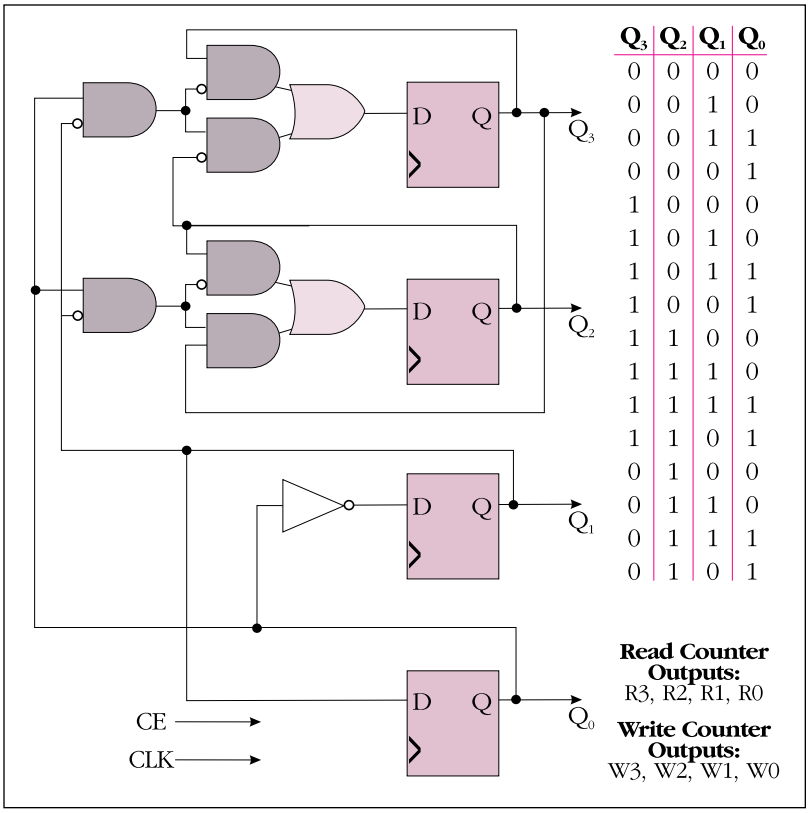
The two 4-bit counters address the RAM in the conventional way. Seen as a “black

box," the FIFO buffer behaves like an elastic shift register: input data is latched by an active edge on the Write Clock, and output data is always available at the output port. FULL and EMPTY signals must be interpreted by external logic to prevent a Write operation during FULL, or a Read operation during EMPTY.

Most of the design effort is spent on the control logic (**Figure 3**) that detects the FULL and EMPTY conditions. Since the easily decoded signal for these two abnormal conditions is the same — read address is identical with write address — an additional signal must be created that distinguishes between the the two very different conditions of FULL and EMPTY. For this purpose, an auxiliary signal, called DIRECTION is created to indicate whether the Write counter is about to catch up with the Read counter, or whether the Read counter is about to catch up with the Write counter. The two most significant bits of both counters are compared, since they indicate in which quadrant of the 16-position circular address space the present address resides. These two most significant bits of both address counters together are used to address two 4-input look-up tables in parallel. The look-up tables (LUTs) decode the relative quadrant position of the two counters.

The 4-bit LUT address describes one of 16 possible conditions:

- Four addresses describe the situation where the write counter is in the quadrant immediately behind the read counter. This is decoded as a "possibly going full" condition, and sets the DIRECTION latch or flip-flop.
- Another four addresses describe the situation where the write counter is in the quadrant immediately ahead of the read counter. This is decoded as a "possibly going empty" condition, and it resets the DIRECTION latch or flip-flop.
- Four other addresses indicate that the two counters are in the same quadrant, and another four addresses indicate that the two counters are in opposite quadrants. These eight addresses provide no useful information about the relative address position, and thus do not affect DIRECTION. Note that DIREC-



Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	1	0
0	0	1	1
0	0	0	1
1	0	0	0
1	0	1	0
1	0	1	1
1	0	0	1
1	1	0	0
1	1	1	0
1	1	1	1
1	1	0	1
0	1	0	0
0	1	1	0
0	1	1	1
0	1	0	1

Figure 2 - Read Counter or Write Counter

Continued on the next page

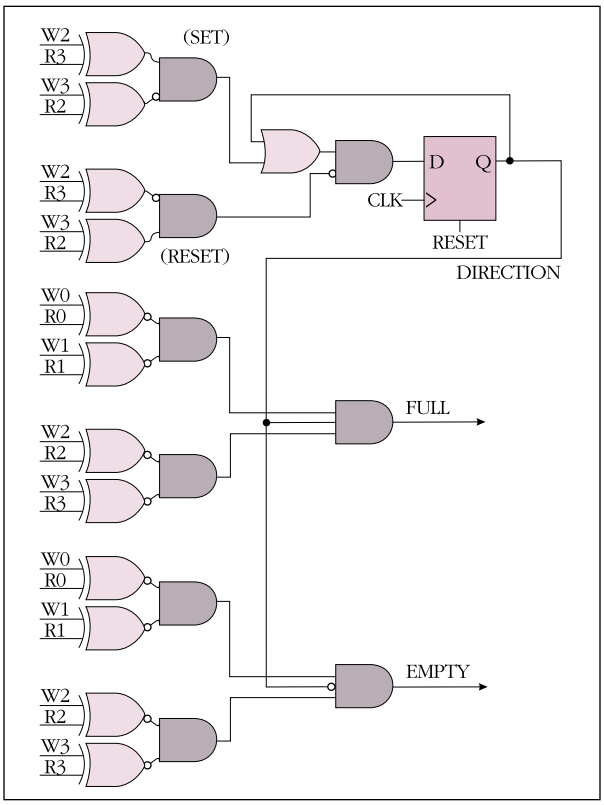


Figure 3 - 16x16 FIFO Synchronous Control

Distributed Arithmetic Laplacian Filter

A common practice in image processing involves convolving an image with a Laplacian operator. **Figure 1** shows a typical Laplacian operator that might be used for edge enhancement. To convolve it with an image, the operator is moved over the image, and centered over each pixel in turn. In each position, the 25 weights in the matrix are multiplied and accumulated with the 25 pixels that the matrix covers. This operation yields one pixel in the resulting image.

-16	-7	-13	-7	-16
-7	-1	12	-1	-7
-13	12	160	12	-13
-7	-1	12	-1	-7
-16	-7	-13	-7	-16

Figure 1 - Example Laplacian operator for edge enhancement in an image processing system.

This is an ideal application for "distributed arithmetic" techniques that exploit the lookup-table (LUT) architecture of the XC4000E™ FPGA family. **Figure 2** shows the basic approach. Four external line buffers plus the incoming video data provide simultaneous access to five lines of the image. Inside the FPGA, each of the video streams is serialized and passed through four 1-bit-wide shift registers, each of which delay the data by one pixel. This provides simultaneous bit-serial access to five adjacent pixels from five adjacent lines — the region covered by the Laplacian filter. The shift registers can be implemented very efficiently using the CLB RAM feature of the XC4000E FPGA architecture.

In the most basic distributed arithmetic approach, the 25 signals address a 2^{25}-word LUT which, in turn, feeds a shifting accumulator. This is obviously impractical. A typical cost-reduction measure would be to partition the problem, segmenting the addresses into multiple smaller LUTs. The outputs of these smaller LUTs would be combined in an adder tree to provide the input to the accumulator.

In this particular case, however, the weighting values involved permit the use of more efficient techniques. Except for the values 160 and -7, each of the coefficients is used in four places.

FIFO Buffer

Continued from previous page

TION must start in the reset state when the FIFO is initiated with both counters at zero.

DIRECTION is thus established well before the actual FULL or EMPTY condition can occur. There will be at least four, and usually many more, consecutive set or reset inputs to the DIRECTION latch or flip-flop before it is being used to discriminate between FULL or EMPTY.

FULL goes active as a result of the write clock edge that writes data into the last available location. FULL goes inactive as a result of the first read clock that reads one word out of the previously full FIFO buffer. EMPTY goes active as a result of the read clock edge that reads the last available data from the FIFO buffer. EMPTY goes inactive as a result of the first write clock that writes one word into the previously empty FIFO buffer. In a synchronous design, FULL and EMPTY are synchronous control signals, to be used appropriately by the logic external to the FIFO buffer.

The application note goes on to describe an asynchronous version of the 16x16 FIFO buffer, and 32x8 and 64x8 FIFO buffers with both synchronous and asynchronous read and write clocks. The larger FIFO buffer designs include input and output data multiplexing between multiple RAM banks. The asynchronous 32x8 FIFO buffer requires 28 CLBs and the 64x8 FIFO buffer needs 48 CLBs; both can perform simultaneous read and write operations at 40 MHz. ◆