Implementing FIFOs in XC4000 Series RAM

XAPP 053 July 7,1996 (Version 1.1)

Application Note by Lois Cartier

Summary

This Application Note demonstrates how to use the various RAM modes in XC4000-Series logic blocks. A simple FIFO is implemented in several different ways, using combinations of level-sensitive (asynchronous) and edge-triggered (synchronous), single-port and dual-port RAM.

Xilinx Family

XC4000E, XC4000L, XC4000EX, XC4000XL

Demonstrates

Edge-triggered and dual-port RAM options

Introduction

The XC4000E and XC4000EX families are similar to the XC4000 family but have additional features. The most significant of these features is the ability to optionally configure each function generator in the Configurable Logic Block (CLB) as edge-triggered or dual-port edge-triggered RAM. This versatile RAM capability is called Select-RAM™ memory. Level-sensitive RAM, which has always been available in XC4000 devices, is still an option in the XC4000E and XC4000EX.

Using the new capabilities, memory-based designs operate at much greater speeds. The First-In, First-Out Memory (FIFO) described in this application note doubles in speed when implemented with edge-triggered RAM compared to level-sensitive RAM. Designs can also take advantage of the edge-triggered functionality to eliminate input data registers, significantly reducing the size of the circuit.

Dual-port mode allows simultaneous read and write. This capability effectively quadruples the speed of the FIFO over level-sensitive implementations—a factor of two from edge-triggered write, and an additional factor of two due to the dual-port mode.

The FIFO application described in this application note was chosen as an example primarily because its simple control logic allows the discussion to concentrate on the memory implementation. Five versions of the design are presented, using different timing approaches and demonstrating the various Select-RAM memory modes with resulting speed improvements. Table 1 contains an information summary of the five designs. Reported operating frequencies are routinely met or exceeded with fully automated implementation of the top-level schematics.

Table 1: Available 16 x 32 FIFO Designs

FIFO Schematic	FIFOA	FIFOR	FIFOE	FIFOS	FIFOD
Top-Level Schematic	FTOPA	FTOPR	FTOPE	FTOPS	FTOPD
VIEWsim Command File	ftopa.cmd	ftopr.cmd	ftope.cmd	ftops.cmd	ftopd.cmd
FIFO Timing	Asynchronous	Asynchronous	Asynchronous	Synchronous	Synchronous
RAM Timing	Level-Sensitive	Level-Sensitive	Edge-Triggered	Edge-Triggered	Edge-Triggered
Number of Ports	Single-Port	Single-Port	Single-Port	Single-Port	Dual-Port
RAM Primitive	RAM16X1	RAM16X1	RAM16X1S	RAM16X1S	RAM16X1D
Size of RPM (in CLBs)	8 Rows x 6 Columns	8 Rows x 6 Columns	8 Rows x 6 Columns	8 Rows x 4 Columns	8 Rows x 6 Columns
Operating Frequency					(with simultaneous read & write)
XC4005E-3	33 MHz	32 MHz	66 MHz	63 MHz	67 MHz
XC4005E-2	~35 MHz	~34 MHz	~71 MHz	~73 MHz	~75 MHz
Supported Devices	XC4000/H, XC4000E/EX	XC4000/H, XC4000E/EX	XC4000E/EX	XC4000E/EX	XC4000E/EX

These simple FIFO designs use the same clock for the read and write cycles. A FIFO with asynchronous read and write clocks is demonstrated in the Xilinx application note “Synchronous and Asynchronous FIFO Designs.”

FIFO Using Level-Sensitive RAM

A typical FIFO consists of four separate logic blocks, as shown in Figure 1. Control logic generates read and write enables. Status logic warns of a full or empty FIFO, or any other desired state such as “half full” or “full minus one.” Read and write address pointers are stored in two up-counters. Data is stored in a RAM array. Control and status logic are very application-specific.

The circuit used throughout this application note is a 16 x 32 FIFO that can be used for a PCI interface. (A PCI interface includes two FIFOs, a Read FIFO and a Write FIFO, which are used as buffers for burst mode transfers.) The starting point is a FIFO implemented using level-sensitive (asynchronous) RAM (FIFOA) that can be implemented in any XC4000-family or XC4000-Series device.[1]

As it stands, the FIFOA design in an XC4005E-3 is barely fast enough to function in a PCI interface. (A speed of 33 MHz is required.) When modified using the Select-RAM memory special features, the design easily exceeds PCI performance.

The FIFOA schematic is shown in Figure 2. The following sections describe each of the four functional blocks.

FIFO Control

The POP and PUSH FIFO inputs are converted in this block into read and write enables, respectively. The arbitration logic, which decides the priority of simultaneous Pops and Pushes, is reduced in this simple example to a decision to ignore the Pop and perform the Push. PUSH is ignored for a full FIFO, and POP is ignored for an empty one. The control signals are registered.

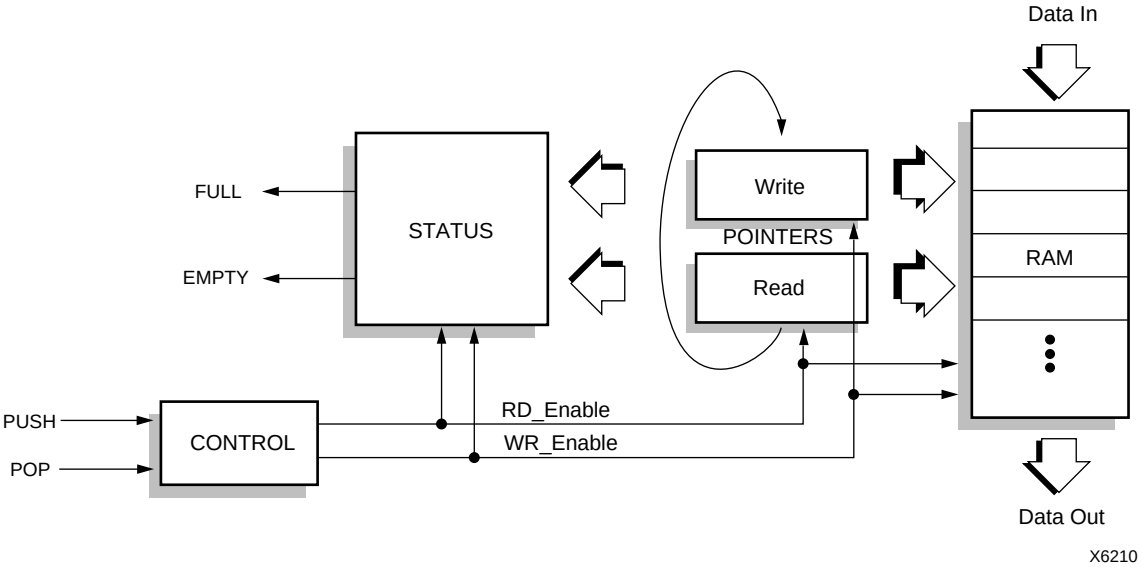

Figure 1: FIFO Block Diagram

1. The term XC4000-Series is applied to any XC4000E, XC4000EX, XC4000L, or XC4000XL device. The XC4000 family, an earlier version of these devices, is not included in the XC4000 Series.

FIFO Status

A common method of determining the FULL or EMPTY status of a FIFO is to compare the read and write pointers. If they are different, the FIFO is neither full nor empty. If they are the same and the last operation was a read, the FIFO is empty. If the pointers are the same and a write was last performed, the FIFO is full.

This method of determining FIFO status is not used in this case because the LAST, or "full minus one," status signal is also needed. Because the control signals and data are registered, one internal clock cycle of delay is inserted between the assertion of PUSH or POP and the actual performance of the indicated operation. There is therefore one clock cycle of delay after PUSH or POP is asserted before the status signals respond to the action. The "full minus one" state must be detected in order to indicate that at most one more byte—which may already be on the way—can be written.

The number of bytes in the FIFO is stored in a 4-bit up-down counter. The count is incremented by a write, decremented by a read, and not affected by a simultaneous read and write. (This last functionality is not necessary for this implementation, but is required when the dual-port RAM is used, enabling simultaneous read and write operations.) The up-down counter is shown in Figure 3.

There are seventeen different states for the FIFO: empty (0 bytes), 1-15 bytes, and full (16 bytes). However, a 5-bit counter is not needed for this application, because the full and empty states are distinguished by the FULL and EMPTY signals.

In this Write FIFO, the "empty plus one" signal is not needed. A PCI Read FIFO would require this signal instead of the "full minus one" generated here. If desired, any state can be generated in a similar manner.

Read and Write Pointers

The read and write pointers are stored in standard 4-bit binary up-counters. The read pointer is incremented by a read operation, the write pointer by a write operation.

The two pointer addresses are multiplexed to create the address for the RAM block.

FIFO Memory Array

The data is registered to synchronize it with the control and status logic. WE latches the data into the RAM on the falling edge of the pulse.

The RAM itself is 16 words deep by 32 bits wide. The number of bits is easily extended to create a wider RAM. Simply increase the number of CLBs used for RAM and use the same address lines and control signals. Increasing the number of words (FIFO depth) requires additional address lines, larger read and write pointer counters, and changes to the status logic.

Decreasing the number of words would make little difference to the size of the logic, as the smallest RAM unit is 16 x 1.

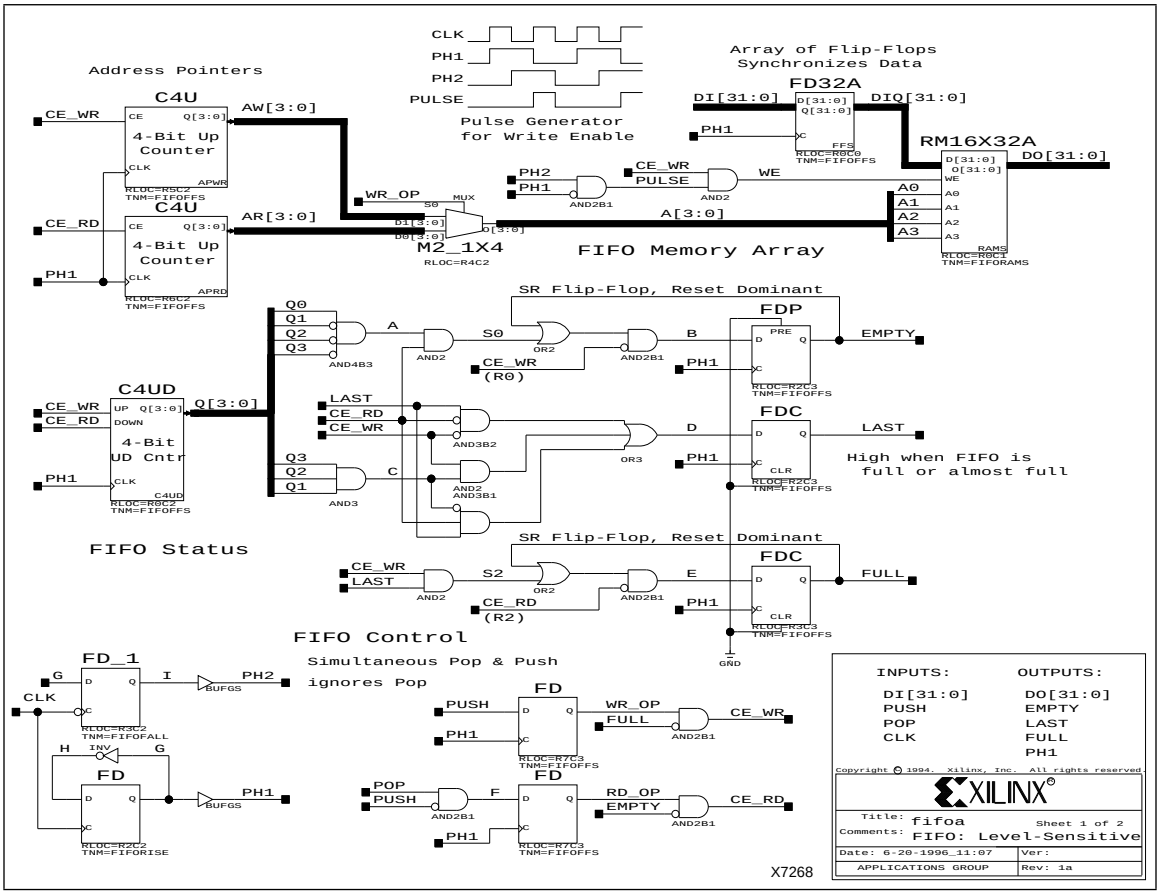

Figure 2: FIFO with Level-Sensitive RAM (FIFOA)

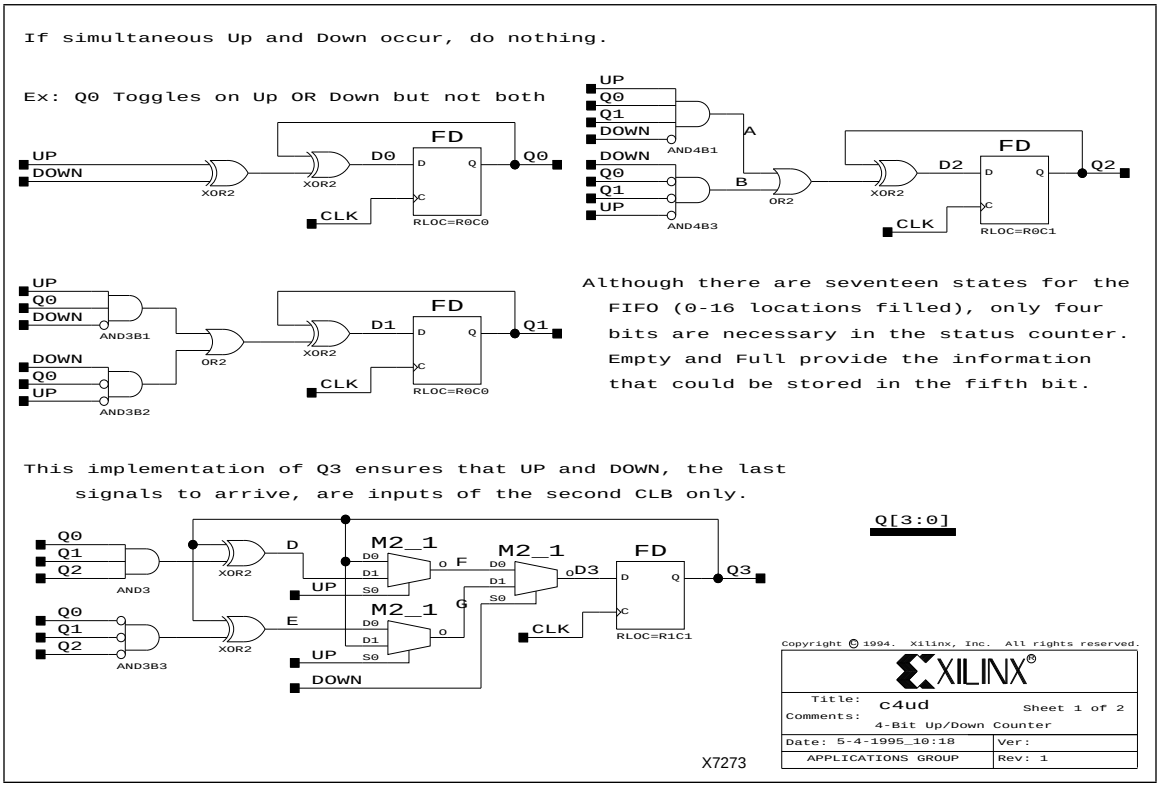

Figure 3: Up-Down Counter for FIFO Status

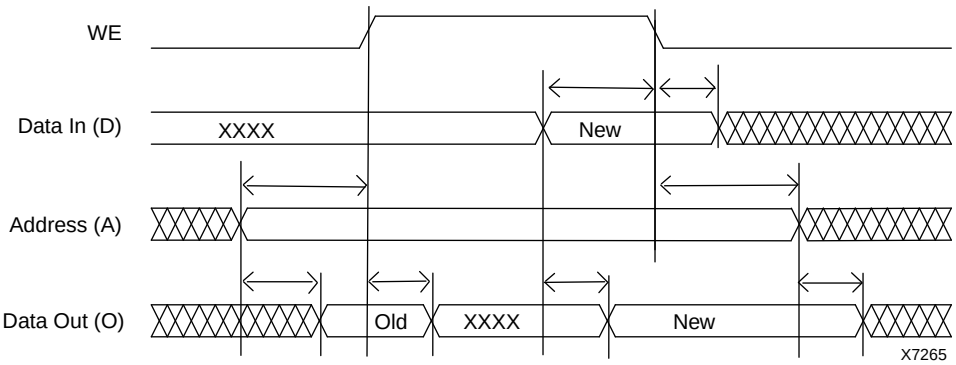

Figure 4: Level-Sensitive RAM Timing

FIFO Timing

The FIFOA implementation includes a clock divider, shown at lower left in Figure 2, that converts the input clock (CLK) into two internal clocks (PH1 and PH2). PH1 is the internal clock for the FIFO; PH2 is used in combination with PH1 to create the WE control signal. Therefore, this implementation clocks the FIFO at a rate half that of the external clock.

The reason for using this clocking scheme is that the RAM Write Enable signal it produces (WE) has controlled, predictable timing, and is glitch-free. Since PH1 and PH2 are both driven through BUFGS clock buffers, the WE pulse width is predictable and relatively independent of operating conditions. Even if the design is converted to HardWire, as long as the clock speed meets the specifications, the width of the WE pulse will never be reduced to the point where it fails to clock the data into the RAM. Alternative XC4000 implementations not requiring a 2X clock do not offer this advantage (see "A Risky Alternative" on page 9).

As with any level-sensitive RAM design, the delay on the WE signal and the address lines must be carefully verified to ensure that WE does not become active before the address lines have settled, and that WE goes inactive before the address lines change again (see Figure 4). During timing simulation, check for setup or hold time violations on address and data lines, or use XDelay to determine the maximum operating frequency. For FIFOs more than a few bits wide, the maximum frequency is usually determined by the delay on the address lines.

In the FIFOA design, the WE pulse is created by the overlap of the PH1 low pulse and the PH2 high pulse. WE therefore occurs in the third quarter of the PH1 cycle. See the small timing diagram at the top of Figure 2. The address lines have half of the PH1 cycle to settle before the rising edge of WE, and remain steady for approximately one fourth of the PH1 cycle after the WE falling edge. The exact timing is dependent on the net delays of WE and the address lines.

Xilinx recommends this method of creating WE, or a similar scheme, for all level-sensitive RAM designs implemented in the XC4000. Select-RAM memory has features that remove the need for a 2X system clock, as discussed in "Edge-Triggered RAM Solution" on page 9.

Physical Implementation

This design is implemented as a Relationally Placed Macro (RPM), occupying eight rows by six columns of CLBs. The data registers and RAM bits are placed in two columns on either side. This arrangement provides easy access for Data In and Data Out nets and allows the placement of address lines on vertical longlines. Pointers, control, and status logic are placed in the two central columns for minimum distribution time of write enable and address lines.

The top-level design used to test this circuit is FTOPA, shown in Figure 5. In FTOPA, each RAM output bit is registered in the same CLB. Placement of the output data registers is guaranteed using RLOC attributes. A TIMESPEC component specifies maximum delays, including delays from flip-flops to RAM address and WE pins.

When placed in an XC4005E-2 device, this RPM achieves a speed of approximately 35 MHz with fully automated placement and routing.

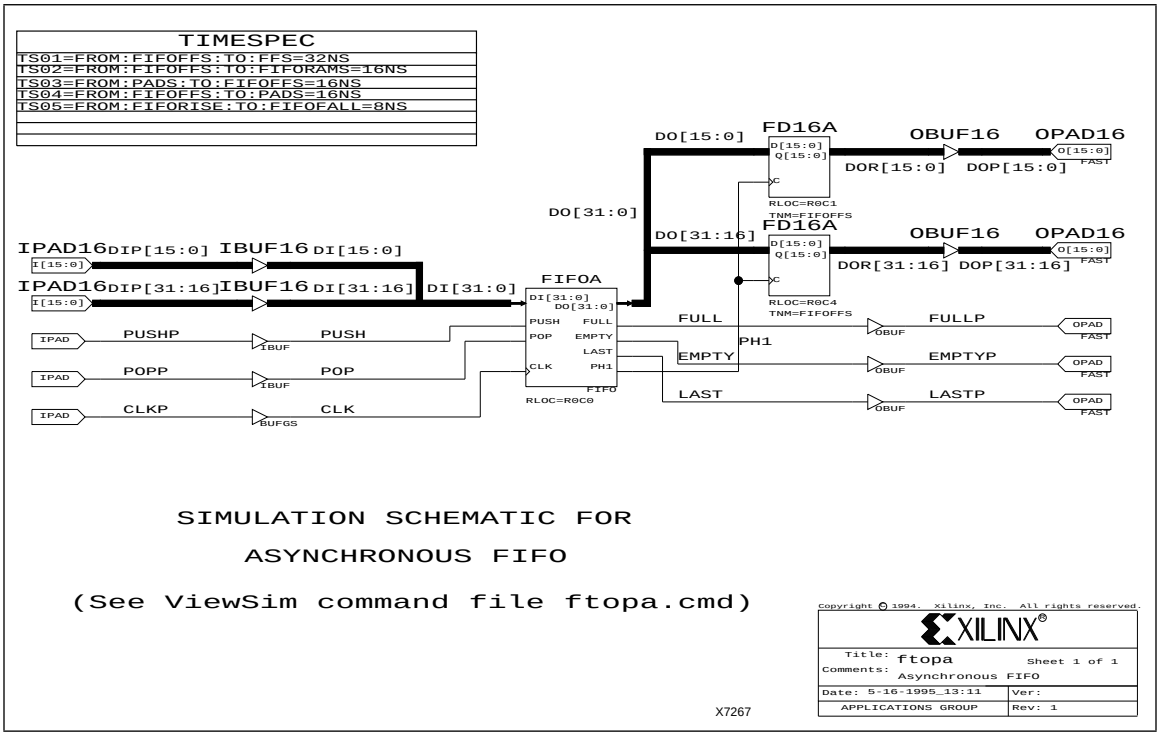

Figure 5: Top-Level FIFO with Level-Sensitive RAM (FTOPA)

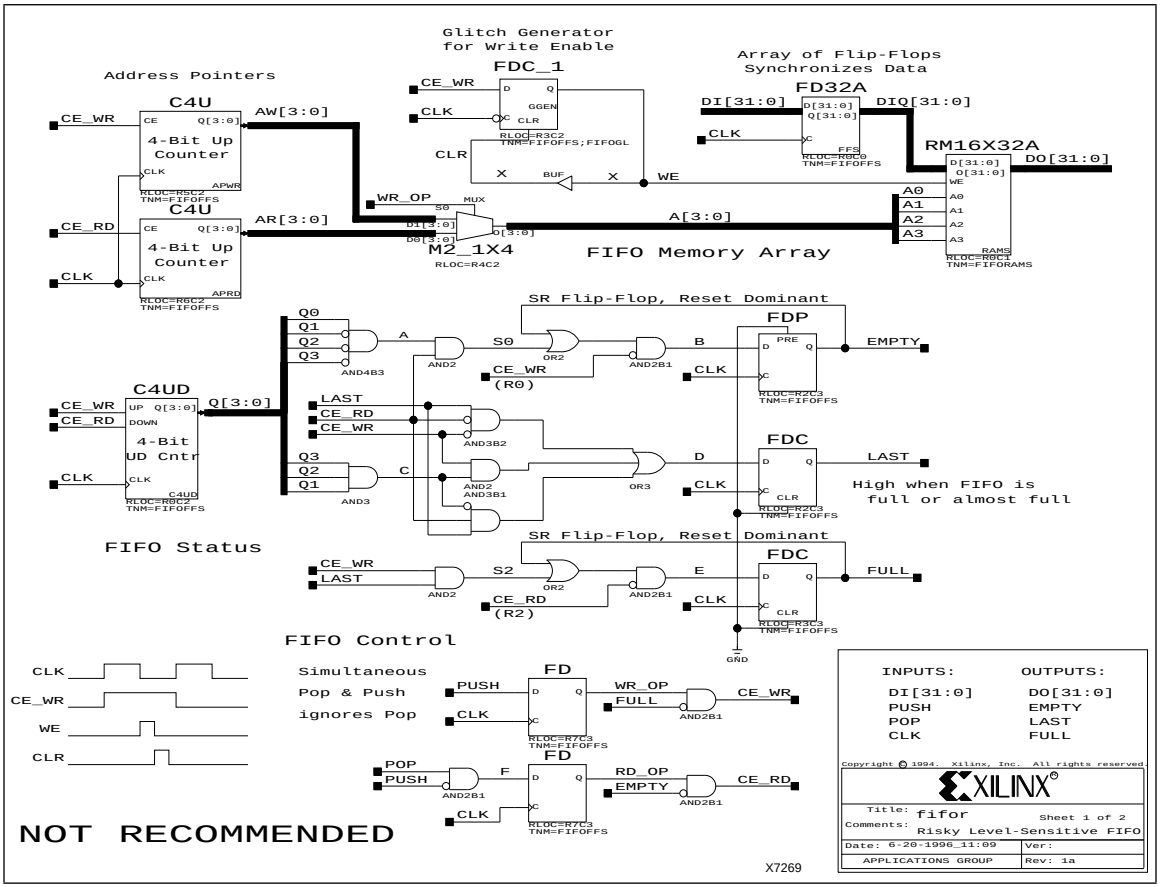

Figure 6: Risky FIFO with Level-Sensitive RAM (FIFOR)

A Risky Alternative

The FIFOA design has a drawback in that it requires a 2X clock to generate the Write Enable signal. In other words, the FIFO can only be clocked at half the speed of the input clock. This requirement is sometimes unacceptable.

Prior to the introduction of the XC4000 Series, when a 2X clock was not available it was sometimes necessary to resort to asynchronous design methods—strictly a last resort in FPGA designs. A pulse generator, triggered by the falling edge of the input clock, can be used to generate the write enable signal halfway through the clock period. The pulse generator and a small timing diagram are included in Figure 6.

This implementation (FIFOR) must be used with great care and is not recommended by Xilinx. It is not a good choice for use in a production design, although it may be an acceptable solution for prototyping. The risk is that under some conditions the width of the write enable pulse may drop below that required to initiate the Write—net delays do not always track logic delays and may swallow up a short WE glitch.

Asynchronous design methods such as this are particularly risky if the design is converted to a HardWire device, which in spite of the specifications will operate faster than the equivalent programmable device.

Alternatively, depending on net delays, the write enable pulse may be too long, so it is still active when the address lines begin to change. In this case, garbage data may be written into the FIFO. This possibility, since it depends on a maximum rather than a minimum delay, can be controlled using XACT Performance and verified using XDelay or timing simulation.

The timing of the FIFOR design is the same as that of FIFOA, except that the external clock is now used directly to clock the FIFO. The address lines have half a clock period to settle. The speed achieved by this design is therefore the same as that of FIFOA.

Fortunately, XC4000-Series Select-RAM memory provides a much better alternative to this potentially dangerous implementation. Much faster speeds, reliable performance, and even, in some cases, greater silicon efficiency can be achieved.

Edge-Triggered RAM Solution

With the advent of the XC4000 Series, it is no longer necessary to generate a write enable signal external to the RAM. The XC4000-Series function generator can optionally be configured as an edge-triggered RAM block. A write enable pulse internal to the CLB is generated by either the rising or falling edge of the RAM Write Clock (WCLK). The WE pin now functions as a true clock enable for WCLK. (See Figure 7.)

WCLK uses the same CLB pin as the flip-flop clock, but can be independently inverted. As a consequence, the RAM output can optionally be registered within the same CLB either by the same clock as the RAM, or by the opposite edge of this clock. The sense of WCLK applies to both function generators in the CLB when both are configured as RAM.

The WE pin is active-High and is not invertible within the CLB.

If one function generator in a given CLB is configured as a 16 x 1 edge-triggered RAM, the other function generator can be used either as a combinatorial logic function or as an additional 16 x 1 edge-triggered RAM. Both function generators can be combined to create a 32 x 1 edge-triggered RAM.

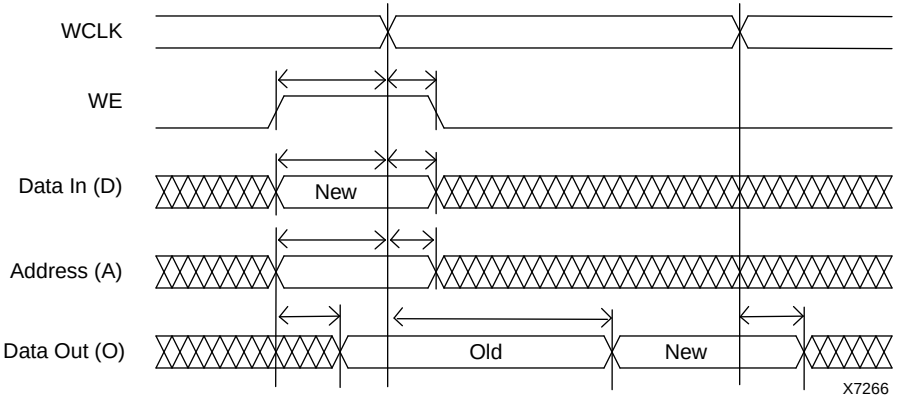

Figure 7: Edge-Triggered RAM Timing

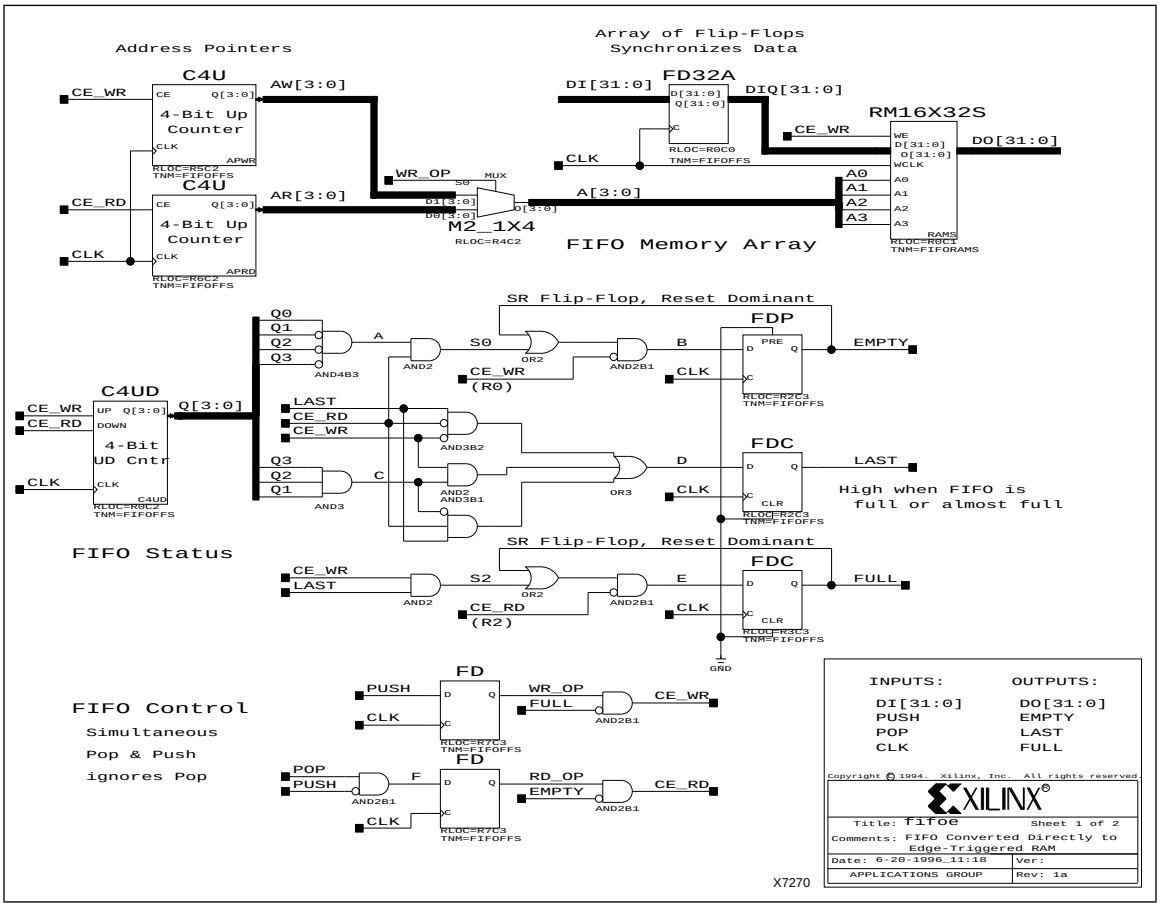

Figure 8: FIFO with Edge-Triggered RAM (FIFOE)

Increased FIFO Speed

Because the WE pulse is triggered by the active edge of the FIFO clock, the address and data lines now have a full clock cycle to settle, rather than the half-cycle available in the previous designs. Since net delay on the address lines is typically the factor determining the speed of operation, the FIFO now operates at twice the clock frequency that was previously possible. This modified design (FIFOE) is a drop-in replacement for FIFOA, but operates at twice the speed. See Figure 8 for a schematic of the FIFOE design.

Converting an Existing Design

To convert the FIFOA design to use edge-triggered RAM, simply replace each RAM16X1 component with a RAM16X1S component. The data and address pins are unchanged. The previous write enable signal, WE, is replaced by the new write clock, WCLK. This clock is gated by a true Write Enable pin, now connected to CE_WR. CE_WR is High throughout the clock cycle during a FIFO Write operation.

The gates previously used to create the WE clock pulse, including the clocks PH1 and PH2, are no longer necessary. They can be removed, and the external clock signal, CLK, can be used directly to clock the FIFO.

Potential Pitfalls

As with any memory design, there are some things to watch out for when using the XC4000-Series edge-triggered RAM.

- CLK must be routed through one of the global clock buffers—either BUFGP or BUFGS—to minimize clock skew within the FIFO. This buffer is not included in the FIFOE schematic; the FIFOE design assumes that CLK is already buffered outside the block.
- Do not violate the minimum active pulse-width specification for WCLK. The active edge of WCLK latches the input data. On the opposite edge of WCLK, the latches become transparent and the data is released. Sufficient time is required within the RAM to complete the Write operation before the data is released. If this specification is not met, incorrect data may be written to the RAM.
- The pulse following the active edge of WCLK has a maximum pulse width requirement, due to the construction of the RAM. The specification is on the order of milliseconds and should not be a serious restriction; however, it should not be forgotten.
- Use XDelay or careful timing simulation to verify that data, address lines, and write enable signals do not violate the setup or hold specifications with respect to WCLK. Using XACT Performance attributes to control the routing of these signals is very effective. RLOC attributes, as in this example, or the Xilinx Floorplanner can ensure optimal placement of the RAM cells and other logic as desired.

Simplified FIFO with Edge-Triggered RAM

With the edge-triggered RAM configuration, a full clock period is available for the data and address lines to settle. In many applications it is no longer necessary to register the data and control inputs as they enter the FIFO block. Eliminating these flip-flops dramatically reduces the size of the overall FIFO.

Figure 9 shows a fully synchronous FIFO (FIFOS) implemented with XC4000-Series edge-triggered RAM. This RPM is eight CLBs tall and four CLBs wide. It operates at the same speed as the FIFOE edge-triggered RAM; a maximum frequency of about 63 MHz is achieved using fully automatic tools. (Relative placement is pre-defined using RLOC attributes as in all Xilinx RPMs; routing is constrained by XACT Performance specifications.)

FIFOS is not a drop-in replacement for FIFOA, the original example design, because the one clock cycle delay on PUSH, POP, and data lines is no longer present.

The potential pitfalls mentioned in the discussion of FIFOE apply to this application, as well as to any other use of the XC4000-Series edge-triggered RAM option.

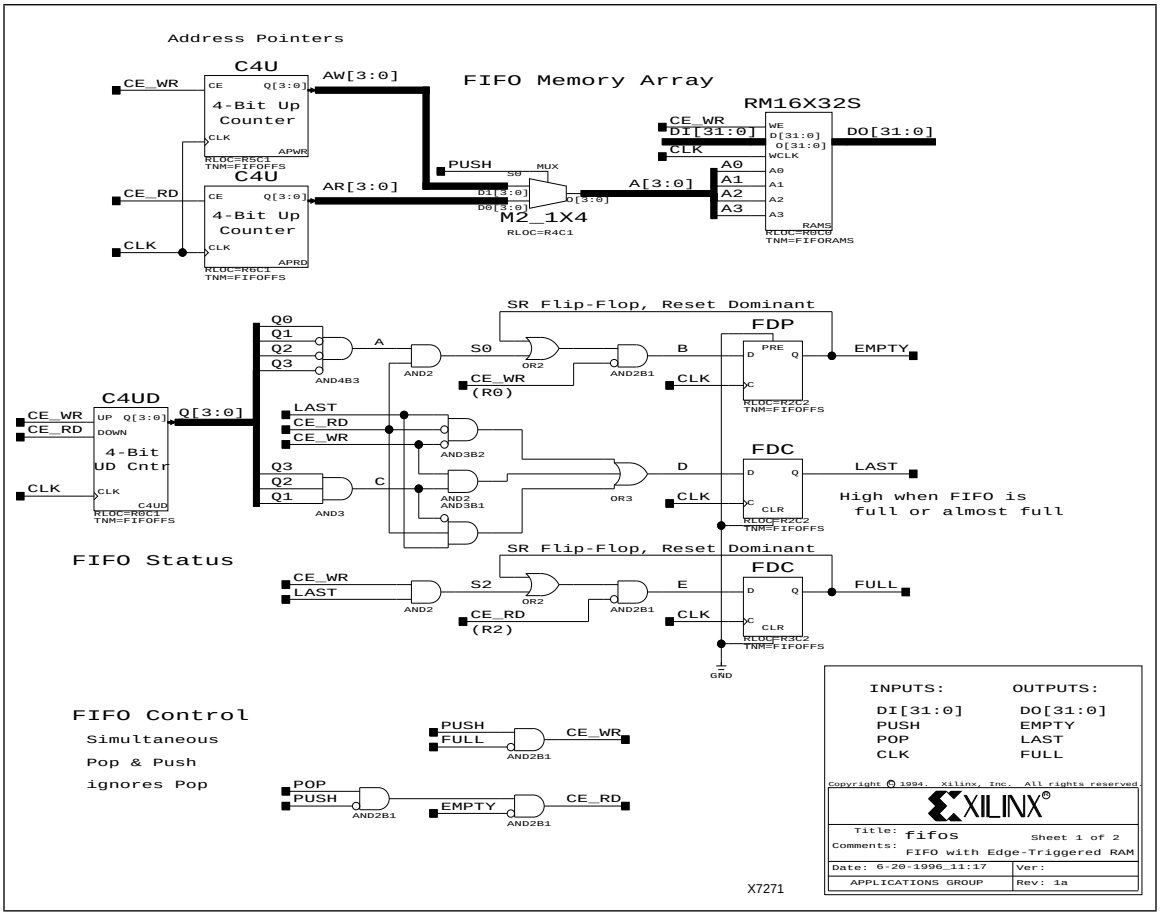

Figure 9: Simplified FIFO with Edge-Triggered RAM (FIFOS)

FIFO with Dual-Port Edge-Triggered RAM

The second major innovation introduced in XC4000-Series Select-RAM memory is the dual-port mode, shown in Figure 10. A single CLB can be configured as one 16 x 1 RAM block, with or without registered outputs, that can be read and written simultaneously at two different addresses. Simultaneous reads and writes at the same address are also supported, with the same timing.

XC4000-Series dual-port RAM is always edge-triggered.

Dual-Port RAM Block

Each of the two function generators has four dedicated inputs; these inputs function as the address lines for the two ports. The "A" address lines provide the write address for both ports. Every write cycle (WCLK active edge with WE high) writes to both ports. Data from address "A" is available at the Single Port Out pin (SPO); data from address "DPRA" is available at the Dual Port Out pin (DPO).

The F function generator, which generates the SPO output, behaves exactly the same as the single-port RAM described in "Edge-Triggered RAM Solution" on page 9. Both read and write addresses come from the F[4:1] inputs.

The G function generator generates the dual-port output. The write address comes from F[4:1], the read address from G[4:1].

All control signals are shared between the two function generators.

To place a RAM in dual-port, edge-triggered mode, use the RAM16X1D symbol.

Dual-Port FIFO

To implement a FIFO that can read and write at the same time, use only the dual port output. "A" is the write address, and "DPRA" is the read address. FIFOD, shown in Figure 11, is such a FIFO.

One advantage of this implementation is that the multiplexer on the address lines is no longer necessary. However, for this implementation the delay between flip-flops is almost as large as the delay on the address lines, so the maximum operating frequency is only slightly higher than the other edge-triggered designs.

FIFOD is not a drop-in replacement for FIFOA, the original asynchronous design, because the one clock cycle delay on PUSH, POP, and data lines is no longer present.

The potential pitfalls mentioned in the discussion of FIFOE apply to this application, as well as to any other use of the XC4000 Series dual-port, edge-triggered RAM option.

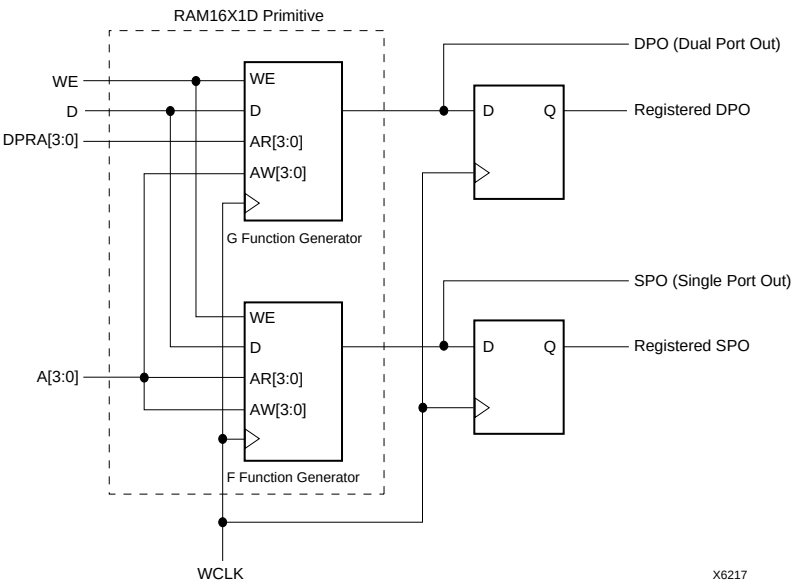

Figure 10: XC4000-Series Dual-Port RAM

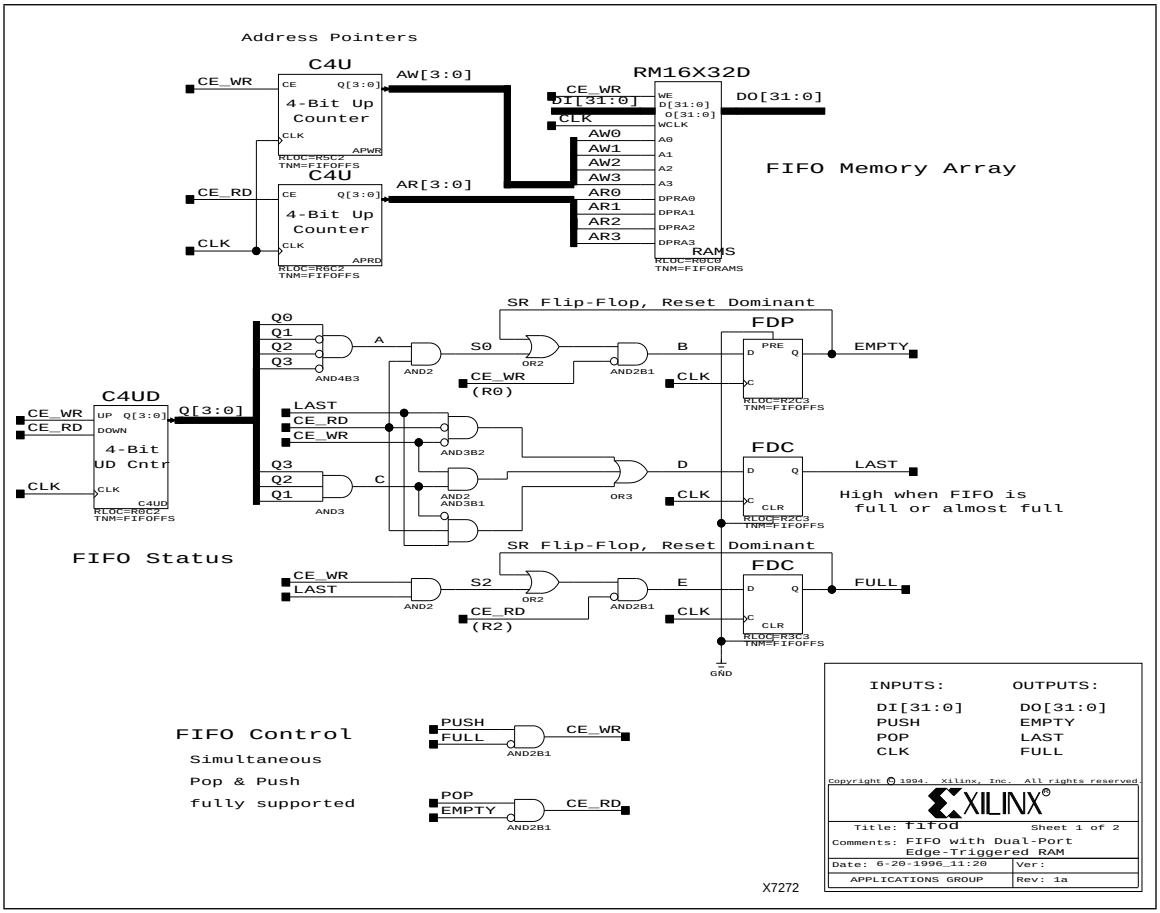

Figure 11: FIFO with Dual-Port Edge-Triggered RAM (FIFOD)

Additional Select-RAM Memory Features

Another feature added to the XC4000 Series is the ability to initialize RAM to a known value. Initialization of ROM has always been possible in XC4000 devices, but this is the first time that RAM is also initializable at configuration.

Use the INIT attribute to initialize RAM to a known value. Bear in mind that the initialization value is only written to the RAM during configuration; it is not renewed by a global reset. If no INIT value is specified, a default value of all zeros is used.

All configurations of Select-RAM memory can be initialized, including the XC4000-compatible level-sensitive mode.

Other applications that can benefit from the new Select-RAM features include shift registers, multipliers, counters implemented in RAM, and read-modify-write RAMs.

Edge-triggered synchronous and dual-port RAM are supported in X-BLOX with the SYNC_RAM and DP_RAM modules.

Selecting an Appropriate RAM Mode

This application note has demonstrated the three different RAM modes available in XC4000-Series devices: level-sensitive, edge-triggered single-port, and edge-triggered dual-port. The question may arise as to which mode is appropriate for a given application. Table 2 shows the recommended usage for each mode.

Frankly, there is little reason to use level-sensitive mode now that edge-triggered RAM is available. One case where it might be preferred is if an existing design is to be updated and a minimum number of design changes is desired.

In cases where CLB usage must be kept to a minimum, single-port edge-triggered mode is recommended.

If simultaneous read and write capability is an asset and silicon efficiency is not a priority, consider using dual-port edge-triggered mode. For designs with very large amounts of dual-port RAM, the increased routing required by doubling the number of address lines might become a problem.

Using the Design Files

All five FIFOs discussed in this Application Note are implemented as Xilinx RPMs.

VIEWlogic schematics are available from the Xilinx BBS. This section describes the available files and the software required to run the design. Also, please read through the “Limitations and Restrictions” section.

Table 2: RAM Mode Selection

	Level-Sensitive	Edge-Triggered	Dual-Port Edge-Triggered
Use for New Designs?	No	Yes	Yes
Density, 16 x 1 (Registered)	1/2 CLB	1/2 CLB	1 CLB
Simultaneous Read/Write?	No	No	Yes
Relative Performance	X	2X	2X-4X

Available Files

The RPMs have names starting with the letters “fifo”. Each RPM is placed in a top-level schematic starting with “ftop”. For example, “fifos” is placed in the top-level schematic “ftops”.

These top-level schematics include global clock buffers where necessary, appropriate XACT-Performance™ specifications for the block, and optional output registers. (For details on using XACT-Performance with Select-RAM memory, see the Xilinx application note “Using Select-RAM Memory in XC4000 Series FPGAs.”) Using RLOC attributes, the output registers are placed into the CLBs containing the corresponding RAM bits, and therefore incur no area penalty on the device. Figure 5 is an example of one of these top-level schematics.

VIEWsim simulation files are included with the data files (ftop*.cmd). Both functional and timing simulations were performed using these command files.

FTOPA and FTOPR can be compiled for XC4000 and XC4000H devices—although the XC4000E library is used in these schematics, the RAM16X1 primitive is the same as in the XC4000 library. If the XC4000E library is not available, use XAltran or Altran to convert the schematics to reference the XC4000 library.

Software Requirements

The following software is required to process this design:

PKUNZIP 2.04e, or later, unarchiving program.

VIEWdraw or VIEWdraw-LCA schematic editor. This software is required in order to make modifications to the schematics.

Xilinx XACT*step* software capable of processing XC4000E designs. This consists of the 5.2.1/6.0.1 release of XACT-*step*™ (or the V1.0.0 XC4000E Pre-Release software, which is obsoleted by XACT*step* V5.2.1/6.0.1).

Using the Design on Your System

- Create a new directory called 4KEFIFO on your hard disk.
- Copy the file called 4KEFIFO.EXE into the 4KEFIFO directory.
- Type 4KEFIFO.EXE on the command line. This command extracts a README.TXT file, and a hierarchical archive of the design files called FIFLES.ZIP.
- Invoke PKUNZIP -D FIFLES.ZIP to extract the files, including their hierarchical path names, onto your disk.
- Edit the VIEWDRAW.INI file. Make sure that the VIEWlogic design library pointers are set appropriately for your machine. You will find the library pointers near the end of the file. Be sure to replace the "xc4000" library with the "xc4000e" library, or the new RAM symbols will not be available.

Limitations and Restrictions

WARNING: THESE ARE UNTESTED DESIGNS.

Xilinx, Inc. does not make any representation or warranty regarding these designs or any item based on these designs. Xilinx disclaims all express and implied warranties, including but not limited to the implied fitness of these designs for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Xilinx does not make any warranty of any kind that any item developed based on these designs, or any portion of them, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is the responsibility of the user to seek licenses for such intellectual property rights where applicable. Xilinx shall not be liable for any damages arising out of or in connection with the use of the designs including liability for lost profit, business interruption, or any other damages whatsoever.

Design Support and Feedback

This application note may undergo future revisions and additions. If you would like to be updated with new versions of this application note, or if you have questions, comments, or suggestions please send an E-mail to

hotline@xilinx.com

or a FAX addressed to "XC4000-Series FIFO Application Note Developers" sent to

1+(408) 879-4442.

IMPORTANT: Please be sure to include which version of the application note you are using. The version number is on page 1.

XILINX® The Programmable Logic Company℠

Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
U.S.A.

Tel: 1 (800) 255-7778
or 1 (408) 559-7778
Fax: 1 (800) 559-7114

Net: hotline@xilinx.com
Web: http://www.xilinx.com

North America

Irvine, California
(714) 727-0780

Englewood, Colorado
(303)220-7541

Sunnyvale, California
(408) 245-9850

Schaumburg, Illinois
(847) 605-1972

Nashua, New Hampshire
(603) 891-1098

Raleigh, North Carolina
(919) 846-3922

West Chester, Pennsylvania
(610) 430-3300

Dallas, Texas
(214) 960-1043

Europe

Xilinx Sarl
Jouy en Josas, France
Tel: (33) 1-34-63-01-01
Net: frhelp@xilinx.com

Xilinx GmbH
Aschheim, Germany
Tel: (49) 89-99-1549-01
Net: dlhelp@xilinx.com

Xilinx, Ltd.
Byfleet, United Kingdom
Tel: (44) 1-932-349401
Net: ukhelp@xilinx.com

Japan

Xilinx, K.K.
Tokyo, Japan
Tel: (03) 3297-9191

Asia Pacific

Xilinx Asia Pacific
Hong Kong
Tel: (852) 2424-5200
Net: hongkong@xilinx.com