XAPP205 (v1.3) August 10, 2000

Data-Width Conversion FIFOs Using the Virtex Block SelectRAM Memory

Author: Nick Camilleri

Summary

Virtex FPGAs provide dedicated on-chip blocks of 4096-bit dual-port synchronous RAM (block SelectRAM+™ memory). The block SelectRAM feature is ideal for use in FIFO applications. This application note describes how to create a common-clock (synchronous) version and an independent-clock (asynchronous) version of a FIFO for data-width conversion with different width read and write data ports.

Introduction

This application note is an enhancement to XAPP131 (170MHz Synchronous and Asynchronous FIFOs using the Virtex block SelectRAM memory). In general, only the changes from XAPP131 will be covered. The reader is strongly encouraged to read XAPP131 before proceeding.

Four different data-width conversion FIFOs are described in this document. The first has a common clock with a 255 x 16 write port and a 1020 x 4 read port. The second has a common clock, a 1020 x 4 write port, and a 255 x 16 read port. The last two are similar FIFOs with independent read and write clocks. The signal names appearing in parentheses reference the names in the Verilog code.

Table 1: **Port Definitions - Common-Clock Design**

Signal Name	Port Direction	Port Width
clock_in	input	1
fifo_gsr_in	input	1
write_enable_in	input	1
write_data_in	input	16 or 4
read_enable_in	input	1
read_data_out	output	4 or 16
full_out	output	1
empty_out	output	1
fifocount_out	output	4

Synchronous Design using Common Clocks

For the synchronous version of the data-width conversion FIFO the control logic is very similar to the synchronous control logic in application note XAPP131. Binary counters are used for both the read (read_addr) and write (write_addr) address counters. Because the address widths are different, the read and write addresses can not be mapped one for one. The difference in width between these two addresses is referred to as a minor address (read/write_addr_minor). The higher eight bits of the addresses are the same on the write side and on the read side. For example, for a 255 x 16 read port and a 1020 x 4 write port FIFO, the read address requires eight bits, and the write address requires 10 bits. Therefore, the upper 8 bits of each address are compared in the flag control logic, and the extra two bits are the minor address. A block diagram of the common clock design is shown in Figure 1. The timing diagram is shown in Figure 2.

© 2000 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at http://www.xilinx.com/legal.htm. All other trademarks and registered trademarks are the property of their respective owners.

To perform a read, the read enable signal (read_enable) is driven high prior to a rising clock edge. The read data signal (read_data) will be presented on the outputs during the next clock cycle. To do a burst read, simply leave read_enable high for as many clock cycles as desired. The last word has been read when Empty becomes active after a read. If another read is attempted, read_data will be invalid.

To perform a write, the write data signal (write_data) must be present on the inputs, and the write enable signal (write_enable) must be driven high prior to a rising clock edge. As long as the Full flag is not set, data will be written into the FIFO. To do a burst write, the write_enable is left high, and new write_data must be available every cycle.

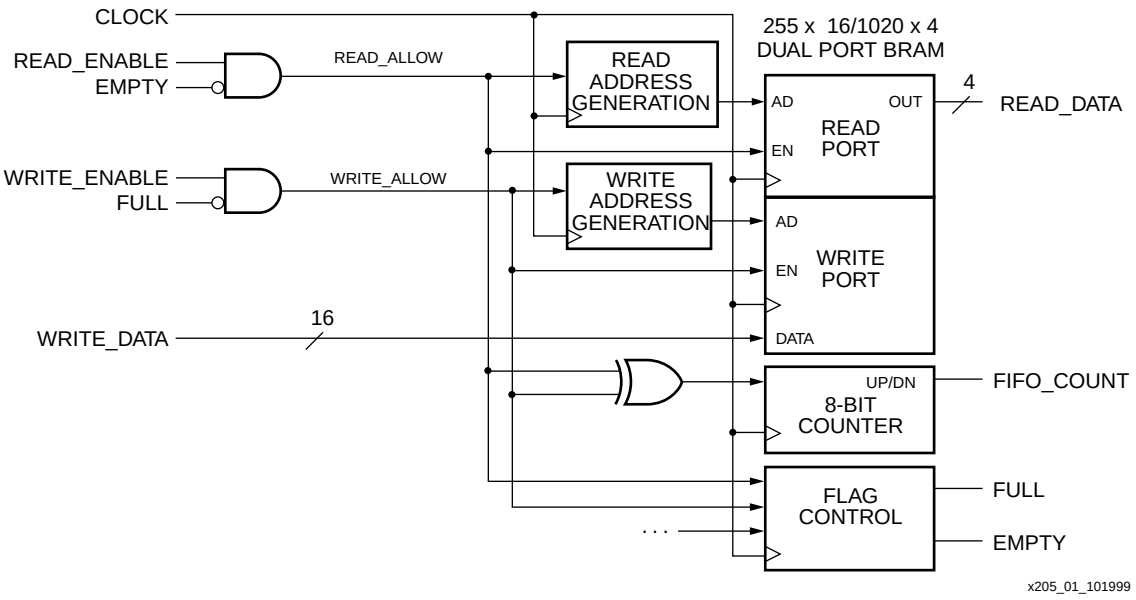


Figure 1: **Block Diagram for a Common Clock Design**

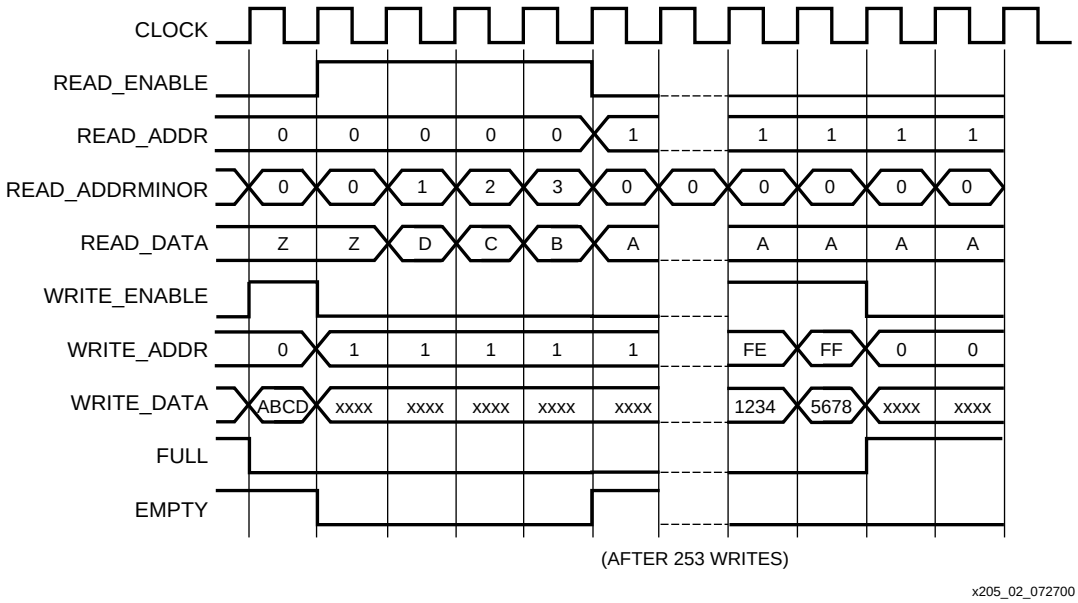


Figure 2: **Timing Diagram for Common Clock Design**

Table 2: **Port Definitions - Independent-Clock design**

Signal Name	Port Direction	Port Width
write_clock_in	input	1
read_clock_in	input	1
fifo_gsr_in	input	1
write_enable_in	input	1
write_data_in	input	16 or 4
read_enable_in	input	1
read_data_out	output	4 or 16
full_out	output	1
empty_out	output	1
fifostatus	output	4

A FIFO count (fifocount) is added for convenience, to determine when the FIFO is 1/2 full, 3/4 full, etc, as shown in Table 3. It is a binary count of the number of words currently stored in the FIFO. It is incremented on Writes, decremented on Reads, and left alone if both operations are performed within the same clock cycle. In this application, only the upper four bits are sent to I/O, but that can easily be modified.

The Empty flag is set when either the fifocount is zero, or when the fifocount is one and only a Read is being performed. This early decoding allows Empty to be set immediately after the last Read. It is cleared after a Write operation (with no simultaneous Read). Similarly, the Full flag is set when the fifocount is 255, or when the fifocount is 254 and only a write is being performed. It is cleared after a Read operation (with no simultaneous Write). If both a Read and Write are done in the same clock cycle, there is no change to the status flags. During global reset (fifo_gsr), both these signals are driven High, to prevent any external logic from interfacing with the FIFO during this time.

To read or write 4-bit data, a secondary enable signal for the control logic is needed (read/write_allow_minor). The corresponding read/write_allow signal for that port will only go active when every 4th nibble is read/written. This signal controls the read/write address counter.

All the initialization values for the address registers have been updated for the change to 8-bit addressing.

FIFO Design Using Independent Clocks

The control logic for an independent clock version relies heavily on the Gray code address counters and equality comparators to perform the deceptively tricky empty/full flag generation. The minor addresses are also included in the control logic, for the previously mentioned 4-bit operations. Figure 3 is a block diagram of the Independent Clock Design. The corresponding timing diagram is shown in Figure 4.

The fifostatus signal indicates 1/2 full, 1/4 full, etc., as shown in Table 3. The task of generating fifostatus in the asynchronous version is more complex, and therefore requires more logic. The overall performance can be improved if this signal is trimmed. The fifostatus outputs have a one-cycle latency for write operations, and a two-cycle latency for reads.

Table 3: **FIFOCount and FIFOStatus Signal Description**

Bit 3	Bit 2	Bit 1	Bit 0	FIFOCount/FIFOStatus
1	1	1	1	15/16 full
1	1	1	0	7/8 full
1	1	0	1	13/16 full
1	1	0	0	3/4 full
1	0	1	1	11/16 full
1	0	1	0	5/8 full
1	0	0	1	9/16 full
1	0	0	0	1/2 full
0	1	1	1	7/16 full
0	1	0	0	3/8 full
0	1	0	1	5/16 full
0	1	0	0	1/4 full
0	0	1	1	3/16 full
0	0	1	0	1/8 full
0	0	0	1	1/16 full
0	0	0	0	< 1/16 full

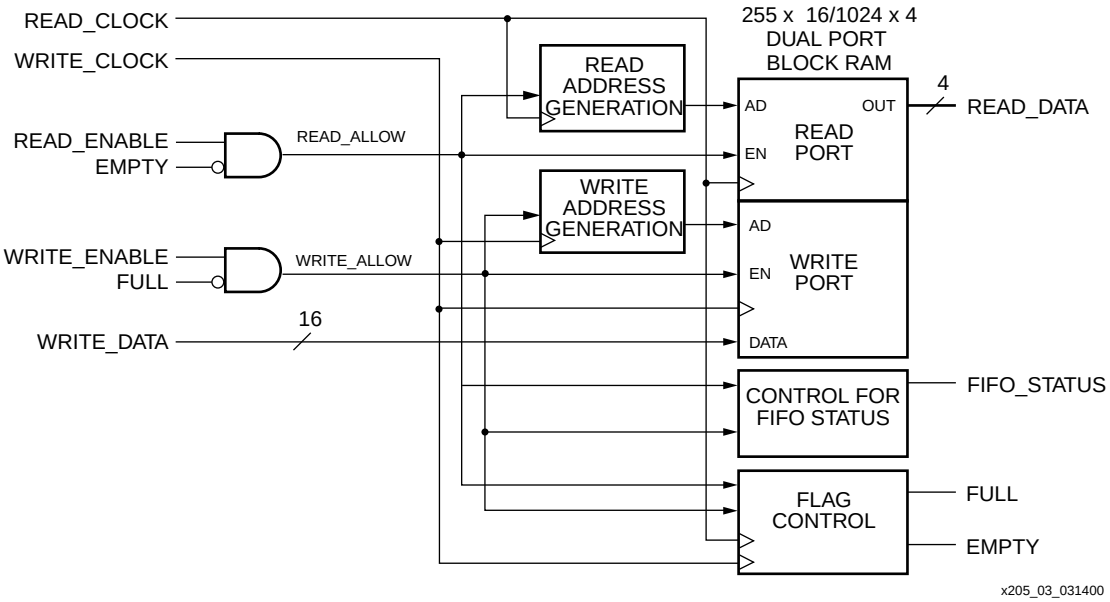
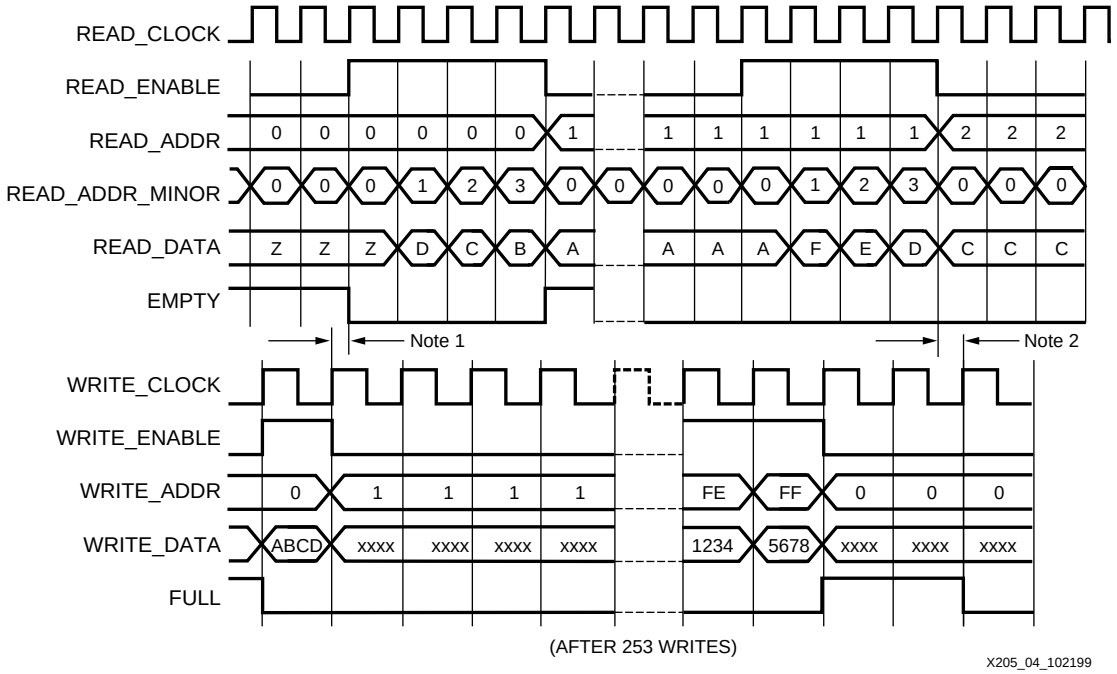


Figure 3: **Block Diagram for an Independent Clock Design**



Notes:

1. EMPTY will go low if the Write Gray Addresses meet the set-up time before the rising edge of READ_CLOCK. If not, then EMPTY will go low when this requirement is met.
2. FULL will go low if the Read Gray Addresses meet the set-up time before the rising edge of the WRITE_CLOCK. If not, then FULL will go low when this requirement is met.

Figure 4: **Timing Diagram for an Independent Clock**

Reference Designs

Each of the four different data-width conversion FIFOs have been simulated using the ModelTech Simulator with individual test bench designs. The designs have not been tested in hardware.

The reference design is available on the Xilinx web site in both VHDL and Verilog files (**xapp205.zip**). Table 4 lists the file names and descriptions for the reference design.

Table 4: **Reference Design Names and Descriptions**

Design	Description
fifoctlr_ccmw1.v	common clocks, 1020 x 4 read, 255 x 16 write
fifoctlr_ccmw2.v	common clocks, 255x16 read, 1020 x 4 write
fifoctlr_icmw1.v	independent clocks, 1020 x 4 read, 255 x 16 write
fifoctlr_icmw2.v	independent clocks, 255 x 16 read, 1020 x 4 write
tb_x205v_ccmw1.v	testbench for fifoctlr_ccmw1.v
tb_x205v_ccmw2.v	testbench for fifoctlr_ccmw2.v
tb_x205v_icmw1.v	testbench for fifoctlr_icmw1.v, with read clock faster than write clock
tb_x205v_icmw1b.v	testbench for fifoctlr_icmw1.v, with write clock faster than read clock
tb_x205v_icmw2.v	testbench for fifoctlr_icmw2.v, with read clock faster than write clock
tb_x205v_icmw2b.v	testbench for fifoctlr_icmw2.v, with write clock faster than read clock

Conclusion

Effective conversion FIFOs can be implemented using the block SelectRAM+ memory available in all Virtex devices.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
9/21/99	1.0	Initial release.
10/25/99	1.1	Revised timing diagrams of Figure 2, Figure 4
3/14/00	1.2	Reformatted text.
8/10/00	1.3	Revised Figure 2 and Table 4