High Speed FIFOs In Spartan-II FPGAs

XAPP175 (v1.0) November 23, 1999 Application Note

Summary

This application note describes how to build high-speed FIFOs using the Block SelectRAM+ memory in the Spartan™-II FPGAs. Verilog and VHDL code is available for the design. The design is for a 512x8 FIFO, but each port structure can be changed if the control logic is changed accordingly. Both a common-clock version and an independent-clock version are described.

Introduction

Many designs require First-In-First-Out (FIFO) elastic buffers to form a bridge between subsystems with different clock rates and access requirements. The Spartan-II FPGAs provide dedicated on-chip blocks of 4096 bit dual-port synchronous RAM, which are ideal for use in FIFO applications. As a result, the low-cost Spartan-II FPGAs can significantly reduce system cost by integrating discrete FIFOs along with other complex logic while providing the speed and I/O levels necessary for interface to larger external memories.

The Block SelectRAM+ memory extends the capability of the distributed SelectRAM memory in CLBs, which requires input and output data multiplexing for depths greater than 16 words. Each Block RAM port can be configured independently as 4Kx1, 2Kx2, 1Kx4, 512x8 or 256x16. The Block RAM is fully synchronous for both writing and reading. For more detail on the Block SelectRAM+ memory, see the Spartan-II FPGA Family datasheet and *XAPP173, Using Block SelectRAM+ Memory in Spartan-II FPGA*s, both available from www.xilinx.com.

This application note describes a 512x8 FIFO, with the depth and width being adjustable within the HDL code. First the design for a FIFO with common read and write clocks (synchronous) is described. Then the design changes required for the more difficult case of independent read and write clocks are presented. This is referred to as "asynchronous" or "unsynchronized" in reference to the two clocks, although the FIFO logic itself is always synchronous. The design files are available from the Xilinx web site in both VHDL and Verilog. Signal names in parentheses are a reference to the name in the HDL code.

Synchronous Design (Using Common Clocks)

The first design is synchronous, which means it uses common clocks for Read and Write. When both the Read and Write clocks originate from the same source, the FIFO operation and arbitration are simplified, and the Empty and Full flags are easily generated. See Table 1 for the port definitions for this design.

Linear Feedback Shift Registers (LFSRs) are used for both the read (read_addr) and write (write_addr) address counters. Because they are addressing a RAM, a binary counting sequence is not necessary, and the pseudo-random sequence of the LFSRs is acceptable. They use very little logic, and are therefore much faster than a standard binary implementation. The only drawback is that the FIFO size is reduced by one, to 511x8. The fifo_gsr signal resets all counters.

The synchronous nature of the Block SelectRAM+ memory simplifies the timing requirements to meeting setup times. To perform a read, Read Enable (read_enable) is driven High prior to a rising clock edge, and the Read Data (read_data) is presented on the outputs during the next clock cycle (Figure 1). To do a Burst Read, simply leave Read Enable High for as many clock cycles as desired. If Empty goes active after reading, then the last word has been read, and the next Read Data would be invalid.

To perform a write, the Write Data (write_data) must be present on the inputs, and Write Enable (write_enable) is driven High prior to a rising clock edge. (See Figure 2.) As long as the Full flag is not set, the Write will be executed. To do a Burst Write, the Write Enable is left High, and new Write Data must be available every cycle.

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	8
read_enable_in	input	1
read_data_out	output	8
full_out	output	1
empty_out	output	1
fifocount_out	output	4

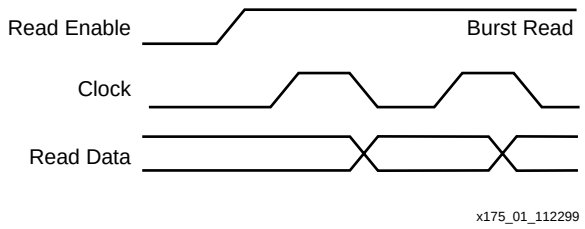

Figure 1: Read Cycle

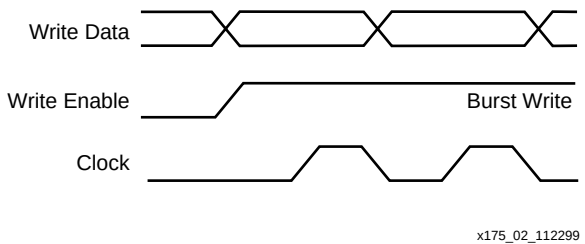

Figure 2: Write Cycle

The Empty flag is set when the Next Read Address (next_read_addr) is equal to the current Write Address, 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 Next Write Address (next_write_addr) is equal to the current Read Address, 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.

A FIFO count (fifocount) is added for convenience, to determine when the FIFO is 1/2 full, 3/4 full, etc. It is a binary count of the number of words currently stored in the FIFO. It is incremented on Writes, decremented on Reads, and stays the same 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.

Changes Required for Independent Clocks

Now we will examine the situation where the read and write clocks are independent or "asynchronous" to each other. The port definitions for this version of the design are shown in Table 2. Keep in mind that the Block RAM itself is still fully synchronous, and the read and write timing is identical to the 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	8
read_enable_in	input	1
read_data_out	output	8
full_out	output	1
empty_out	output	1
fifostatus_out	output	5

In order to operate a FIFO with independent Read and Write clocks, some asynchronous arbitration logic is needed to determine the status flags. The previous Empty/Full generation logic and associated flip-flops are no longer reliable, because they are now asynchronous with respect to one another, since Empty is clocked by the Read Clock, and Full is clocked by the Write Clock.

To solve this problem, and to maximize the speed of the control logic, additional logic complexity is accepted for increased performance. There are primary 9-bit Read and Write binary address counters, which drive the address inputs to the Block RAM. The binary addresses are converted to Gray-code, and pipelined for a few stages to create several address pointers (read_addrgray, read_nextgray, read_lastgray, write_addrgray, write_nextgray) which are used to generate the Full and Empty flags as quickly as possible. Gray-code addresses are used so that the registered Full and Empty flags are always clean, and never in an unknown state due to the asynchronous relationship of the Read and Write clocks. In the worst case scenario, Full and Empty would simply stay active one cycle longer, but this would not generate an error.

When the Read and Write Gray-code pointers are equal, the FIFO is empty. When the Write Gray-code pointer is equal to the next Read Gray-code pointer, the FIFO is full, having 511 words stored. Additional comparators are used to determine when the FIFO is Almost Empty and Almost Full, so that Empty and Full can be generated on the same clock edge as the last operation. (Traditional control logic uses an asynchronous signal to set the flags, but this is much slower and limits the overall performance).

Unlike the common-clock version, it is not possible to keep a reliable count of the number of words in the FIFO, so a FIFO status output is used instead. It is five bits wide, with the signals representing various ranges of fullness, as seen in Table 3.

Table 3: FIFO Status

FIFO Status Bit	Description
fifostatus[0]	FIFO is between Empty and 1/4 Full
fifostatus[1]	FIFO is between 1 word and 1/2 Full
fifostatus[2]	FIFO is between 1/4 Full and 3/4 Full
fifostatus[3]	FIFO is between 1/2 Full and Full
fifostatus[4]	FIFO is between 3/4 Full and Full

The FIFO status outputs are mutually exclusive, meaning only one will be High at any one time, but the ranges that they cover overlap. They are based on the Gray-code pointers, and the quadrant deltas that exist between the Read and Write addresses. Because of the nature of Gray-code counting, more precision (such as one based on octants) can be easily added, because the upper bits of a Gray-code address are themselves Gray-coded, so there will not be any incorrect status registered.

Design Files

Both the common-clock and independent-clock designs are available on the Xilinx web site as xapp175.zip for PC and xapp175.tar.Z for UNIX. Implementation requires the Xilinx Foundation 2.1i (or later) development system with the latest Service Pack. The designs are available in both VHDL and Verilog and can be customized for different FIFO sizes or other requirements. Note that Block RAM is always instantiated in HDL code. The RAM is initialized to zeroes by default but an INIT attribute can be used to initialize to a different value. The common-clock design requires one Block RAM and approximately 22 slices, or about 10% of the smallest Spartan-II device, the XC2S15 FPGA.

Revision History

Date	Version	Revision
11/23/99	1.0	Initial Xilinx release.

© 1999 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.