FIFO Buffer Module with Watermarks (Verilog and VHDL)

- Features
- Introduction
- Background
- Application
- Design Software
- Conclusion
- Additional Information
- Appendix
    - Verilog Files:
    - VHDL Files:
- Feedback for Our Sponsor

Features

The following topics are covered via the Lattice Diamond ver.2.0.1 Design Software.

- Overview of the FIFO Buffer Module and common usage
- Watermark implementation
- Configuration of FIFO
- FIFO Buffer Module Testbenches

Introduction

This module (in both Verilog and VHDL) is a First-in-First-Out (FIFO) Buffer Module commonly used to buffer variable-rate data transfers or to hold/buffer data used in digital communication and signal processing algorithms. For example, a FIFO module can be used as a circular buffer or delay line in a FIR filter.

If available, the tools will use the embedded block RAM resources within the FPGA. The FIFO.vhd and FIFO_v.v modules are verified in testbenches by writing and reading values to and from the FIFO while observing the RAM data and the condition of the output flags.

Background

The FIFO module is a variable-length buffer with scalable register word-width and address space, or depth. There are watermark flags available for “almost full” and “almost empty” conditions. The depth of the “almost full” and “almost empty” flags can be adjusted within the module’s parametrization, or generic block in the case of the VHDL version. The FIFO also has flags for empty, full and error. There is an output port for reading out the data count. As the port name suggest, this tells the world (outside the module) how many words are currently stored between the read and write pointers within the RAM.

The Software required/used for this design:

- Lattice Diamond Design Software version 2.0.1 with third party software Synplify Pro for Lattice and Active-HDL Lattice Edition.

Application

Building the Circuit

The FIFO_v.v and FIFO.vhd can be configured by changing the values within the parameterization and generic parameters respectively within each module. The output word width can be scaled as can the FIFO address space. The watermark flags can be set to trigger at various depths by adjusting the ALMST_F and ALMST_E parameter values.

The design has a data input port, a clock, active low reset, read enable, write enable, data output port, data count port, empty, full, almost empty, almost full, and error signals. The VHDL generated black box model with inputs and outputs can be seen below in figure 1.

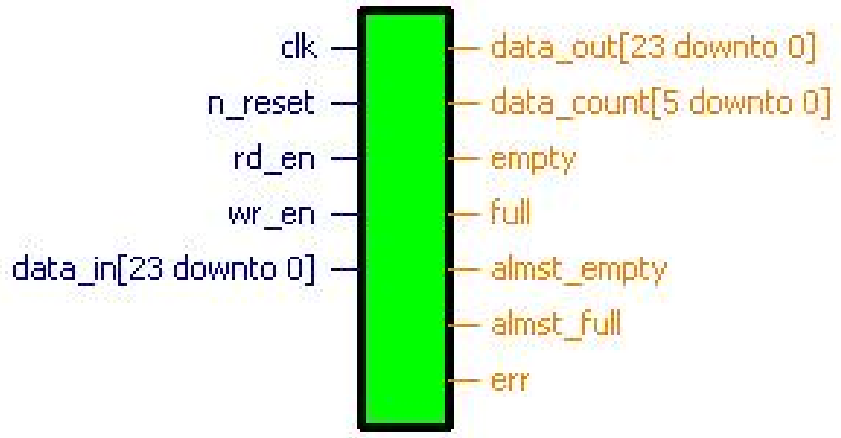

Figure 1 – Black Box Diagram for FIFO Buffer

The FIFO Verilog implementation HDL block diagram from the Lattice Diamond “Generate Hierarchy” function can be seen in figure 2 below.

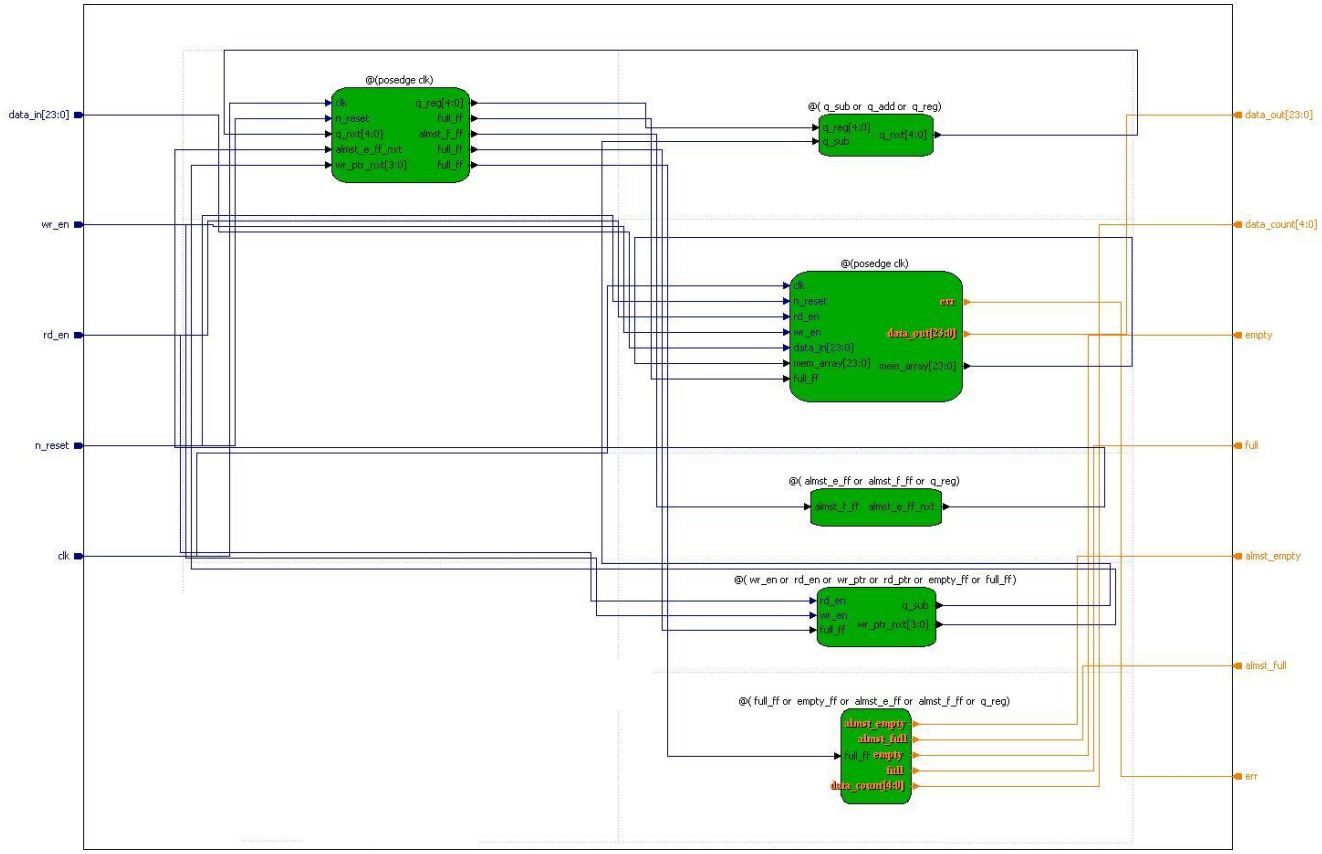

Figure 2 – FIFO Block Diagram

The basic procedural blocks, or process blocks in the case of the VHDL implementation, are used to infer combinational logic to control the pointers and flags to make a FIFO out of an inferred RAM block. The general blocks used in both VHDL and Verilog designs are as follows:

- Flip flop Update, rising edge clock sensitive
- Almost Full/Empty Flag Control, combinational
- Read and Write Pointer Control, combinational
- Memory Array Read/Write Control, rising edge clock sensitive
- Counter with Control Flags, combinational control
- Output Register Connections

The RTL diagram for a 24-bit data, 4-bit address space, with 4 word deep “almost full” and “almost empty” flags can be found in Figure 3 below.

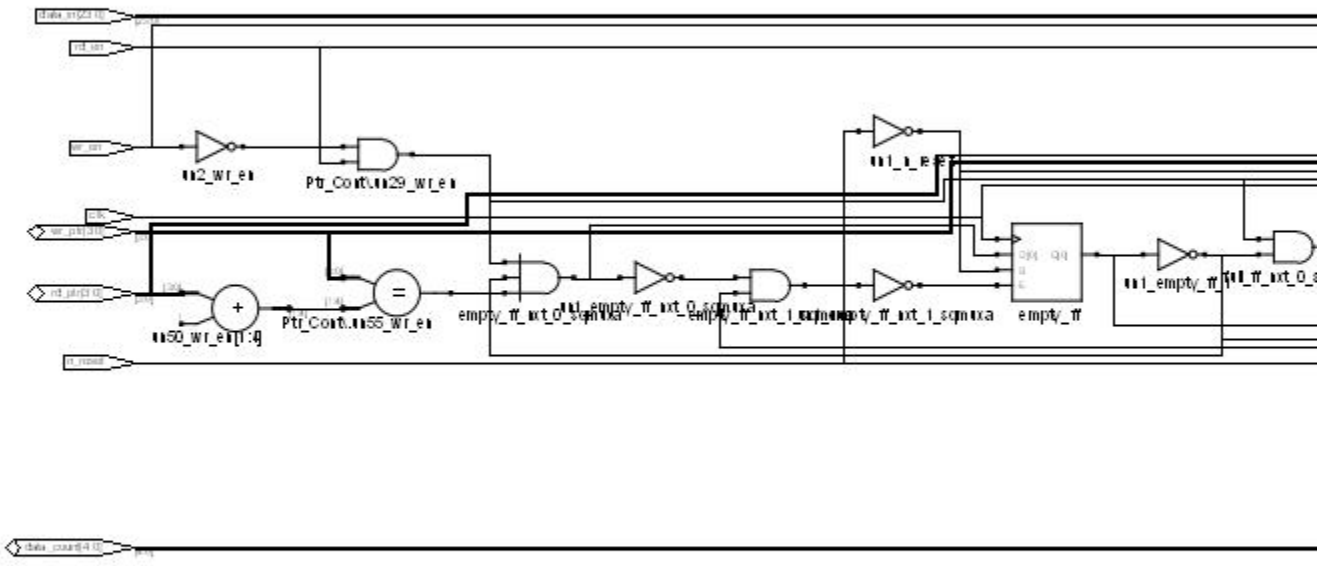

Figure 3 – RTL Diagram for FIFO Buffer

The Aldec Active HDL Testbench output with test vectors can be seen below in figure 4. When the FIFO approaches either an empty or full state, the adjustable watermark flags will go high followed by the respective empty or full flag. An error flag goes high if there is an attempted FIFO write when full or a FIFO read when empty.

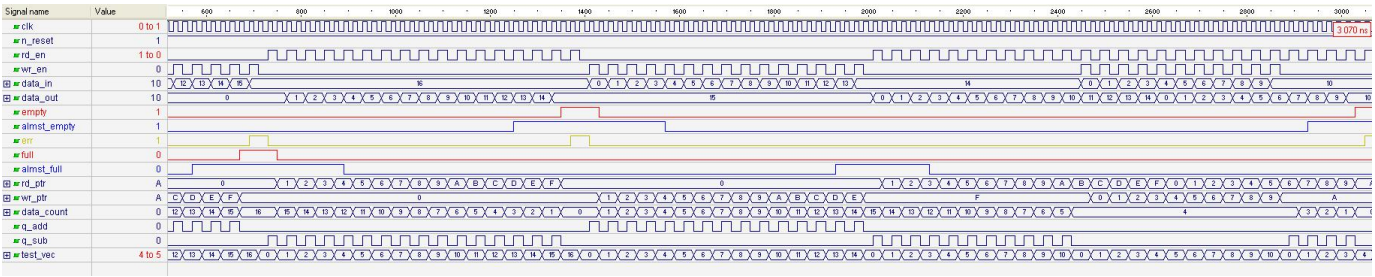

Figure 4 – FIFO Aldec Active HDL Testbench Output

The included test bench was created from the “generate test bench template” command in the “HDL Diagram” window. The testbench was then modified to include various test cases to exercise the condition flags and ensure proper read and write pointer control. The test bench provided should cover most test cases.

Design Software

Lattice Diamond version 2.0.1 was used to develop the “FIFO.vhd” and ”FIFO_v.v” with supporting software from Synopsis (Synplify Pro for Lattice) and Aldec (Active-HDL Lattice Edition). Diamond can be used as a stand alone development environment with alternative synthesis and simulation software.

Conclusion

This design demonstrates the use of a configurable FIFO module in both Verilog and VHDL using the Lattice Diamond Design Software. The Verilog and VHDL testbenches illustrate the use of “for-loops” to generate test vectors and signal delays.

Additional Information

Further design support, product tutorials, application notes, users guides and other documentation can be found on the Lattice website at http://www.latticesemi.com/support/index.cfm?source=topnav.

For questions relating to this reference design, the author can be contacted at:

Appendix

Verilog Files:

FIFO_v.v

FIFO_v_tf.v

VHDL Files:

FIFO.vhd

fifo_tb.vhd

Feedback for Our Sponsor

Please take a few seconds to help us justify the continued development and expansion of the eewiki.

Click on one of our Digi-Key links on your way to search for or purchase electronic components.

Is the eewiki helpful? Comments, feedback, and questions can be sent to eewiki@digikey.com.