From: Saved by Internet Explorer 11
Sent: Thu, 18 Jan 2024 14:39:03 +0300
Subject: Lab 3

Lab 3: Programmable data delay with random numbers

Part 1: PRNG

In this part you will write a four-bit pseudorandom number generator (PRNG).
Instructions:

Starting from the single-bit PRNG discussed presented in Lecture 4, construct a PRNG with a four-bit output. Each of the four output bits should have a unique sequence, so instantiate four single-bit PRNGs with different seeds their respective generic maps. Use four generics in your top-level entity declaration to define the four different seeds.

Simulate the PRNG, and examine output sequence to make sure that all possible outputs (0-F) are produced, with no obvious repeats in the pattern.

Part 2: Block memory as a programmable delay buffer

In this part you will design a programmable, synchronous delay buffer based on a dual-port block RAM, similar to the one presented in Lecture 4.

Instructions:

The RAM can be generated in Vivado, using the IP Catalog in the Project Manager. Using the IP generator example in Lecture 4 as a guide, select the Block Memory Generator and configure it as a simple dual-port RAM (write to port A and read from port B). Since the buffer input and output should be synchronous, you can select the Common Clock option in the Basic configuration tab. This will produce a memory with both ports using the clka input.

In the Port A and Port B option tabs, configure the memory to have 4-bit wide input and output data widths. Choose a data depth of 256 words, corresponding to 8 address bits. Both ports should always be enabled.

After you have generated the memory, it will appear under your design sources in the Project Manager. In the Hiearchy view of the Sources window, you can navigate down to the VHDL

description of the block memory that you have configured. If you choose the IP Source view, you can find VHDL and Verilog instantiation templates that you can copy and paste into your design.

The 8-bit address lines of ports A and B are produced by an unsigned 8-bit counter that you will also need to design (or generate as another CORE IP).

To produce a timing offset between the input and output of the buffer, you will need to add an unsigned 8-bit offset vector to the counter output. The addition can be implemented as an IP core, an 8-bit adder component of your own design, or an unsigned addition VHDL operation in your top-level architecture.

In your final design, the port B address should be connected to the counter output, while port A should be driven by the counter+offset sum. The input ports should include an 8-bit vector for setting the offset with eight user switches, and a clock input either connected to a push button or the 100 MHz oscillator (pin W5). The output ports should include four LEDs for displaying the current output of the buffer, and two PMOD port pins connected to the lowest bits of the buffer input and output, respectively.

Simulate the design with different delay offset values, and verify that the delay of the buffer changes as it should.

Part 3:Testing the programmable delay

Begin implement your design on the BASYS-3 board, with your clock input from one of the push buttons. Using an output delay setting of 1, press the clock button repeatedly and watch the changing pattern in the four output LEDs. The pattern should appear random, with no obvious repeats.

Implement the design again, this time using the on-board 100 MHz clock. With a digital oscilloscope, compare the two PMOD port outputs, and measure the timing delay for different switch settings. You may ask the instructor for help setting up and operating the oscilloscope.