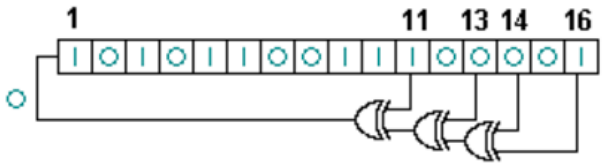
Lab 3: Random number generator and programmable data delay

Introduction

The main portion of this lab exercise is designing a synchronous pipelined data buffer with 4-bit wide inputs and outputs, designed to delay an input data stream by a programmable length. You will also design a four-bit parallel pseudo-random number generator (PRNG) to provide a data source for testing the buffer.

Pseudorandom number generator (PRNG)

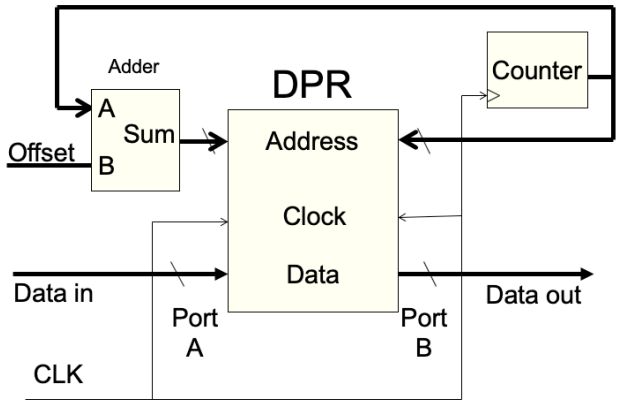


Begin by designing and simulating a single-bit PRNG with a 16-bit register, as discussed in Lecture 4. The starting value of the register (seed) should be set by a generic statement in the entity.

Next, construct the 4-bit wide PRNG by instantiating four single-bit PRNGs, one for each output bit of the generator. Make sure that each single-bit PRNG has a different seed.

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.

Programmable delay buffer



In this part you will design a programmable, synchronous delay buffer based on a dual-port block RAM with programmable length, similar to the one presented in Lecture 4. This is a commonly used element of experimental data acquisition systems, and will provide experience with instantiating and customizing vendor-provided IP cores.

The RAM can be generated in Vivado using "IP Catalog" in the Project Manager (see the IP Catalog example in Lecture 4 as a guide). Select the Block Memory Generator (look under Memories & Storage Elements/RAMs & ROMs and BRAM) and configure it as a simple dual-port RAM (write to port A and read from port B).

For this design, the input and output data should be synchronous (i.e. share the same clock), so you should select the Common Clock option in the Basic configuration tab. This will produce a memory with both ports driven by the `clka` input.

Under the "Port A" and "Port B" option tabs, configure the memory to have input and output data ports that are 4 bits wide (data width). The address port should be 8-bits wide, corresponding to a total of 256 addresses, so choose a data depth of 256 words. Both ports should be "always enabled".

After you have generated the memory, it will appear in your design sources hierarchy in the Project Manager. To use it, you can choose to the "IP Sources" view and navigate down to the "instantiation templates". There you will find VHDL (.vho) and Verilog (.veo) templates that can be copied and pasted into your design.

The buffer also requires 8-bit "counter" and "adder" to drive the 8-bit address lines of the two ports as shown in the diagram above. You may choose to implement these as processes in the top-level design, or as components (either your own VHDL modules or additional IP cores generated in Vivado IP Catalog).

Top-level design

To test the buffer, create a top level design with the following ports:

- Inputs: a clock and an 8-bit user "offset" to set the delay
- Outputs: Two 4-bit ports to read out the "input" and "output" of the buffer

The 8-bit offset input should be connected to the input of the delay buffer, and the `data_in` and `data_out` ports of the buffer should be connected to the outputs. The 4-bit PRNG should internally provide input data to the buffer (`data_in`).

In simulation, do multiple runs with different user offsets and note the relative timing delay between the buffer input and output.

You may also choose to implement the design in hardware, using eight user switches to set the offset, and the clock provided by the 100 MHz oscillator (pin W5). The output ports may be connected to LEDs and/or PMOD port pins. If you have access to an oscilloscope in the lab, you can compare the buffer delays in real time. Alternatively, the input and output data to the buffer can be sampled and monitored in Vivado by adding an integrated logic analyzer and monitoring it in the Hardware Manager.