ELEX 3525 : Data Communications
2020 Fall Term

Self-Synchronizing Descrambler

Introduction

While AWG's and digital 'scopes are useful for generating and capturing waveforms, they cannot operate in real time or sustain high throughputs. Programmable logic devices such as CPLDs and FPGA's are flexible and inexpensive and can be used to implement specialised communication test equipment to generate test data or measure errors. A wide range of communication protocols and signal processing IP blocks are available from FPGA manufacturers and others.

In this lab you will practice this approach by designing a self-synchronising (multiplicative) descrambler using the Altera Quartus Prime FPGA design software and testing it by simulating with a test waveform supplied by the instructor.

The schematic below shows a multiplicative *scrambler* used to generate the test waveform. As described in the lecture notes, the *descrambler* uses the same components but they are wired differently.

This scrambler computes the exclusive-OR (xor) of the current input bit and bits 5 and 9 of a shift register where the bits are numbered so that the oldest bit is bit 9 and the most recently output bit is bit 0. Note that this numbering may not match the numbering in the diagram in the lecture notes.

The extra D flip-flop on the output ensures that the output changes synchronously with the clock. You should 'register' your de-scrambler's output in the same way.

Procedure

Create a new Quartus project using all the project defaults except the project folder and name. Create a block diagram file (BDF) using the block diagram editor (File > New... > Block Diagram/Schematic File). Add the components and I/O pins required.

You can implement the shift register using either discrete D flip-flops or define a shift register using an LPM component as shown below (select Tools > IP Catalog and under Library / Basic Functions / Miscellaneous select `LPM_SHIFTREG`). You can use either VHDL or Verilog file types. Use any name for the variation file name (e.g. sr).

Select the required number of bits (see description above), right-shift (MS bit towards the LS bit), parallel data outputs, serial data input and synchronous clear. You will need to generate a `.bsf` file in addition to the Verilog or VHDL files. Add the component to the Project library.

Use the Symbol tool (gate icon) to insert an instance of the shift register component from the Project library. Use the Pin Tool to add three input and one output pins. You must use the pin names `clock`, `reset`, `datain` and `dataout` to match the names used in the supplied test waveform. The '`datain`' and '`dataout`' signals are the input and output data bits (active high, H=1). The shift register is clocked (shifted) on the rising edge of 'clock'. The `reset` signal is used to clear the shift register at the start of the simulation.

Label nodes to connect them. You can assign a name to a node or bus by right-clicking and selecting "Properties". This node or bus will then be connected to all others with the same name. You can label buses with the range of signals to be included (e.g. `sr[9..0]`). The schematic below shows some examples.

Save the project and design files and compile the design. If there are any errors, fix them and recompile the design.

Open the `.vwf` file supplied on the course web site (see the example screen capture below) and click on the "run functional simulation" icon[1]. The waveform output will show a distinctive pattern. Use a screen capture tool to capture the waveform for your report.

Report

Submit a (PDF format) report containing the identification information asked for in previous labs, a

[1] If you get an error about a `novopt` option, use Simulation > Simulation Settings and remove the `-novopt` setting from the `vsim` command options in the Modelsim Script text box.

schematic (block diagram) of your working circuit, and the waveforms showing the test input and the output of your circuit.