ECE 6770 – Lab Assignment 4

Due 2 March 2017 via Canvas

1 Introduction

In this lab assignment you will extend the process of designing your special arithmetic function chip. This will accomplish four main tasks:

1. Fix the design and simulation flaws from your Lab 2 project to create a correct design with robust flows.
2. Investigate a design-for-test (DFT) flow.
3. perform hierarchical chip-level design.

You will use the Makefile based project management system started in Lab 2. All of the synthesis, physical design, simulation, and validation steps will be integrated into that design system so you can perform all steps sequentially, or any of them independently.

You will add two more blocks (a pseudo-random test pattern generator and a signature analyzer) that constitute a built-in logic block observation (BILBO) system to your design. This allows a design to be automatically tested for manufacturing flaws. The test will cover stuck-at faults and delay faults that cause the circuit to fail timing. The pseudo-random test pattern generator will be used to feed inputs to the multiplier and the signature analyzer will verify the correct functionality of the multiplier. Behavioral code for the pattern generator, which is a linear feedback shift register (LFSR), and for the signature analyzer are provided. You will need to synthesize these blocks to generate their structural verilog netlists.

2 Multiplier Chip Design

The top-level design consists of a four-stage pipelined multiplier, a 16-degree LFSR for generating pseudo-random inputs to the multiplier, and a signature analyzer block for examining the output of the multiplier.

The LFSR provided implements an n-degree polynomial going through $2^n - 1$ states in an unconventional sequence before repeating the states. The only state not covered is the all-zero state.

The signature analyzer is a variant of the LFSR used to provide data to the multiplier which compresses the output of the pipelined multiplier.

The core of the design has been provided. You will need to modify the design to add a `test_mode` pin that will switch between performing operations in BIST mode and taking inputs from the regular data stream. Determine the hierarchy of your design such that there are three blocks, the LFSR, the core MULT block, and the signature analyzer. The mux and demux functions can be in any of the blocks, but the blocks must stand independently for Lab 5 (layout and, DRC and LVS).

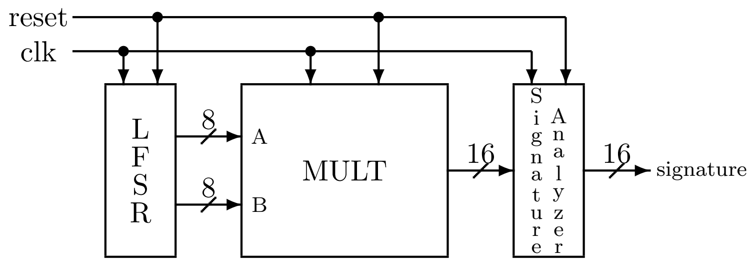

Figure 1: Sample Multiplier Chip Floorplan

3 Floorplan and Partitioning

Figure **??** gives a sample floorplan for your multiplier chip without the test mode muxing. The three blocks correspond to the three components instantiated at the chip top-level. A rough guide for pin assignments can be taken from the floorplan. Pictures of the floorplan can be used as a guide when creating the place and route scripts.

4 Deliverables

The first aspect of this lab will be to debug and fix any problems that you had in Lab 2 with generating a correct design and simulation of a multiplier. You will create a design that is a four pipeline stage design where your simulation scripts accurately model the timing you used when you synthesized your design. No negative slacks are allowed in the synthesis. In addition, *you rewrite the test bench for the MULT core in Verilog.*

Create a C++ or other formal specification that proves the correct operation of the design under BIST operation.

You will also create a test bench for the design when it operates in BIST mode. In this mode you will run the simulation for a full iteration cycle of the 16-bit LFSR: $2^{16} - 1$ iterations. At the end you will read out the signature and verify that it correctly matches the golden value proving correct operation.

For normal operation you will use the same test bench from Lab 2.

Each of the three blocks will be synthesized: the multiplier from lab 2, and the lfsr and signature analyzer provided on the web page from this lab, along with any adjustments to the blocks to enable test mode and normal operation. Negative slacks are not allowed. The frequency of each block must be identical for all the blocks. Timing of the interfaces between each block must be clearly identified in the .tcl scripts with delays at the interface fully specified such that a full clock cycle is covered.

The simulations for the normal operation and test mode will operate on the top level design in a possibly modified chip.v file. This module will contain three components: the structural synthesized mult, LFSR, and signature analyzer blocks.

After working through this lab, submit as a tar file to canvas the following items in an organized directory tree. **Do not include any temporary or additional files in your tar ball!** Your submitted directory should have **only** the following files:

- README.txt: A text file containing a brief description of any problems that you encountered and how you solved them. Describe how you interconnected the signals and power and ground in the top level hierarchical cell, chip.v. Note what you changed in the chip.tcl and chip.conf file compared to the lower level block scripts.

Evaluate the BIST operation and how effective you feel this will be in determining manufacturing faults. Comment on whether you need to execute all $2^{16} - 1$ input patters to cover all stuck-at faults in the design and all timing failures that come from manufacturing faults or variations.

Create a table clearly lists the timing of the interfaces: when inputs and outputs are valid from each block, and that interfaces have correct timing and margins that cover no more than a clock cycle with sufficient margin (setup and hold). Also list the cycle time that you have.

- The new Verilog test bench for the multiplier function core that matches timing from the test bench. A test bench you develop that tests the design in BIST mode. It will only need to check the result at the end of the complete exhaustive set of iterations.
- The C++ or other golden model for the design under BIST mode. The result will be a single 16 bit signature that will identify correct manufacturing of the design.
- The `<DESIGN>.dcopt.constraint` files for the three modules and the `<DESIGN>.dcopt.out` files.
- The Makefile and all .tcl scripts used to synthesize all three modules.
- Snippets of the simulation waveforms that show the timing in the simulation matches the timing identified in the README file. These need to show the relationship between primary inputs and outputs to your design and the clock, and when signals change and stabilize between the three hierarchical modules. Take the snapshot of the simulation waveforms with `xv`.
- Behavioral and structural verilog for your mult, lfsr, and signature modules, and the behavioral verilog for chip.v.