

CECS 360 Fall 2016 Project 5

Summary of Integrated Circuit Design Software

By

Shinesunny Som

12/8/2016

Instructor: John Tramel

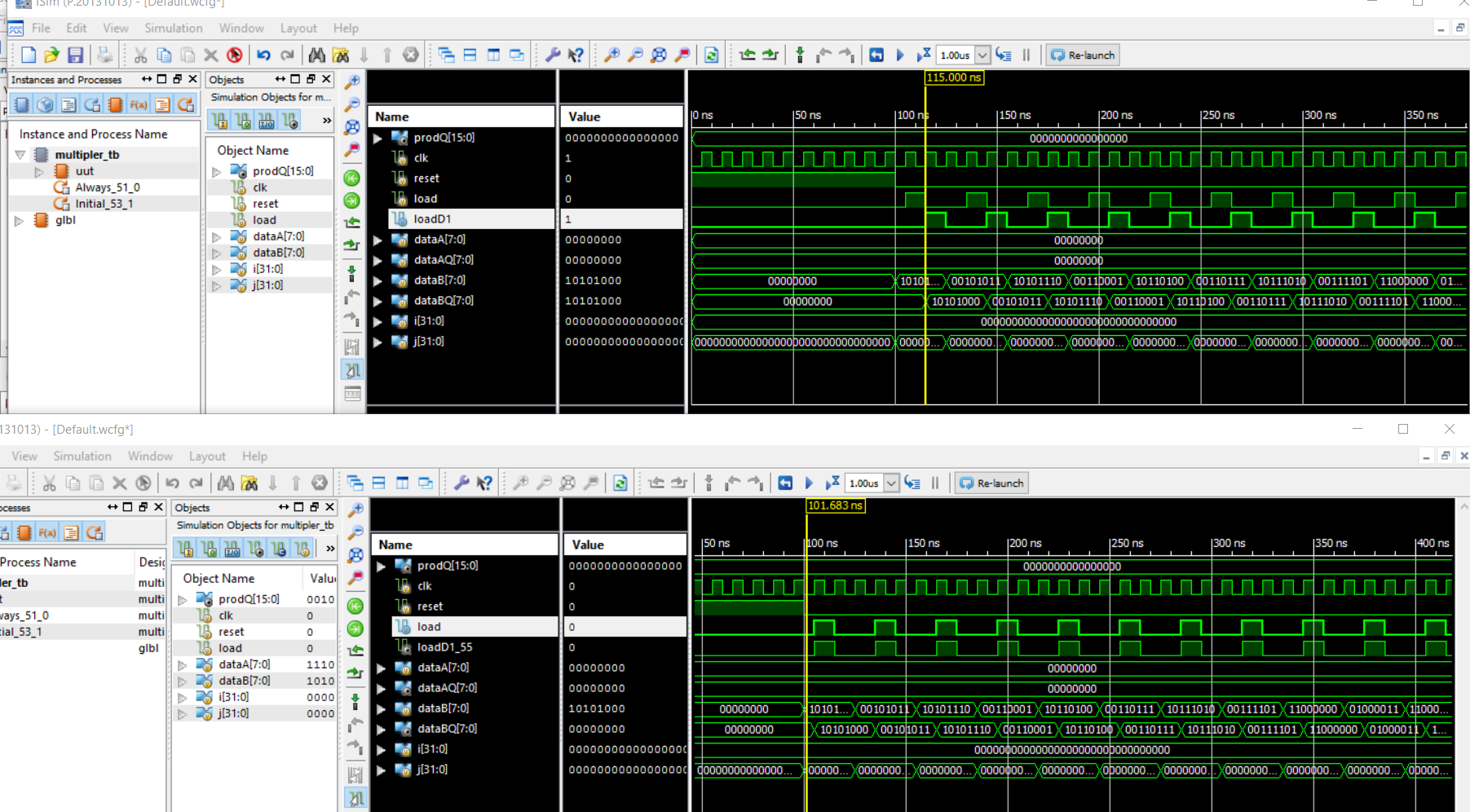
**Lab Description:** Technically not considered a lab but rather a summary of what I have learned the entire course from CSULB CECS 360. For the rest of the report, I will outline the topics and include my personal thoughts of the course and, explain throughly what I have learned. I will also include the last topic covered in class at the end of the report.

**Project 1 - AISO:** Some students claimed the first lab was ***weak.*** On the other hand, I thought it was a good lab to start the semester knowing the fact that I felt like many students came out of CECS 301 with different levels and professors taught it (Not everybody had Allison). I have taken Allison for 301 before but, even I felt extremely rusty, and not to mention the newer improved, Pong Chu’s debounce module is arguably better than the one Allison provided. The crux of learning the first project understand what AISO (Asynchronus in, synchronus out) is and give everybody a warm up and having everybody in the same page in terms of learning 201+301 all in a few weeks.

**Project 2 – Designing a Testbench/Project 3 – Verification:** If I had to narrow down of the many things I learned in this course and I thought it was ***the most valuable*** it would be Project 2, and 3. In my previous courses with verilog (CECS 201, 301), we have learn to use testbenches, and presume with “not knowing why and how it function that it is correct and it works.” I find this portion of the course to be the most significant because I could picture visually this in the working field. Generally, you are self-checking, verifying designs of other projects, rather than generating it from scratch. Not saying that creating project isn’t important, in fact, it is equally as important as testing the modules and entirety of the design. Project 2 help me understand how to develop testbenches better and understand what to look for when creating a stimulus. Project 3 gave a deepen of understanding of self-checking designs, and use vectors to test the design. It also included the use of HLL which give an actual proper use of applying something applicable outside of this course realm to complete the project. It takes a bit of creativity as well to provide multiple solutions for simulating testbenches and verifying design as most of us engineer will be at minimum required to learn and to perform.

**Project 4 – Simulating the Netlist:**

Just as equally as important as all of the other projects, this particular project is my favorite one (Not because it was easy.) It given me a more emphasis of being more keen in verification checks.



“Behavioral simulation only give you an ideal perfect simulation test while using the post-translation simulation will simulate a more of closer gate, register inspection of a design.” Understanding propagation delays is the most important of verifying design.

**Static Timing Analysis:**

The final lectures of instruction (including the last 2 labs) gave emphasis on static timing analysis. In any module design, the UCF file will provide two key functions: Identify the system clock, and provide the desired mapping of the I/O design. You can also further investigate and make changes specifically to clock periods, and editing the timing constraints. You can also edit the output contraints and view the layout of the FPGA which we are programming the file into. With the desired clocking contraints, we can also analyze the post-map static timings by simulating the Post-Map Simulation found in the iSim. The Post-Place & Route Timing analysis can be a useful tool to meet timing and to ensure that all of the paths meet timing.