ECE 571 – Introduction to SystemVerilog
Fall 2022 Pipelined Picoblaze final project

Problem Statement:

You and your teammates (teams of 3) will test and verify a pipelined Picoblaze RTL model. The pipelined Picoblaze was written by John D. Lynch and Roy Kravitz in 2006 for a course they taught at OGI (the now defunct Oregon Graduate Institute) as the final project in a computer architecture course.

The Pipelined Picoblaze is based on the Pacoblaze model written by Pablo Bleyer Kocik. Pacoblaze is an RTL implementation of the Picoblaze **I**nstruction **S**et **A**rchitecture. The Xilinx Picoblaze architecture was originally created by Ken Chapman, a Xilinx FAE, in 2003 and has been ported to many Xilinx FPGA families including the Series 7 FPGA used in ECE 540 and ECE 544.

For this project you will define, implement, debug, and make use of a verification environment that explores verification techniques such as constrained randomization, checkers, assertions, stimulus generators, etc. that we will discuss in ECE 571. You will write and execute a Verification plan using a SystemVerilog simulator (most likely QuestaSim).

Your team will be provided with a working (thoroughly verified by an ECE 571 final project teams in Fall 2021) SystemVerilog model of the Pipelined Picoblaze. Any testbenches or additional capability you add to the module must be written in SystemVerilog.

The focus for the final project is on verification, not on design.

You will make a 20-minute technical presentation of your project during Final's week which includes:

- Your verification environment, the approach you followed, and the techniques you applied
- The results you achieved, bugs found in the "broken" model and which test(s) found them.
- Challenges you encountered
- What additional work would do if you had more time.
- Pre-existing code/ideas leveraged (besides the Pipelined Picoblaze model)
- Assessment of the code coverage of your test suite (QuestaSim can help with that)

You will describe your verification strategy, including an outline of tests, assertions, and verification environment in a Verification plan. You will submit a Verification report based on your Verification plan summarizing your results, changes/enhancements from your original Verification plan and conclusions at the end of the project. The report should list the work done by each of the team members.

Hints for your final project team:

- From experience, the most significant challenge in forming and managing a project team is to understand how much time and effort each of the team members is willing, and able, to put into the project. Some students are extremely interested in the course and in mastering SystemVerilog and are willing to put in an extraordinary effort. Others are not. This is the first, and most important, discussion to have with your team before start on the project.
- Also of great interest is the whereabouts of the members of your team; that is, one or more of your team members may be taking the course remotely. If this is the case the team needs to agree on how they will communicate and coordinate the work. The project is too big for a single team member and all team members are expected to contribute a similar amount to the results of the project.
- You need to have a good project plan. This is not the Verification plan - it is how you are going to manage the project team to ensure that the project successfully meets the goals you set out for the team. A key element of planning is to discuss schedules and the work process. Your team will be using GitHub for this project, but how will you communicate? Slack? Discord? Text message? emails? How often will your team meet, and how? Will you use Zoom (all PSU students and staff have a Zoom license) or Google meetings? Will you meet in person with proper social distancing? Having a good plan is important for all projects, but it is even more important for a project like this with a limited, but somewhat undefined, scope and a fixed amount of time to complete the work.
- You will need to have a good schedule. List milestones (I like to list milestones with 3 – 5 day activities). Check your schedule for what I call "touch points" and what project management tools call "dependencies". So often I see Gantt charts that have bars for each task and each individual but there are very few interdependencies. Most projects fail because the team underestimates the time it takes to integrate the test environment and/or because there is no time in the schedule to recover from problems. Consider allowing time for a 2nd pass through your Verification plan.
- The third thing to discuss is skill set of the team members. Look for complementary skill sets in your team. Who is good at hardware design? Who is good at coding? Who has an interest and a desire to find problems in code they didn't write? Assign tasks based on skill set and interest as much as you can. In project management-speak this, and your schedule, is sometimes called the WBR…The Work Breakdown.

I can visualize you rolling your eyes when you read through this. After all, we're talking about a 3-week project. We're talking a single verification effort, not an entire chip design. However, you are going to put a lot of effort (at least I hope you will) into this project and planning for success is more likely to end up in a success (and a better final project grade) than doing things in an ad-hoc manner.

Presentation:

You will make a 20-minute presentation to the T/A and instructor during the Final's week. Each member of the team must make part of the presentation. Your presentation should briefly describe your design (block diagram, etc.) but delve more deeply into the verification strategy your results. *Of particular interest is the assertions you have placed in your design that found bugs.*

Deliverables:

- Your Verification Plan
- Your final project presentation materials (slides, etc.)
- A completed Verification Report. Start with your Verification plan and fill in the results of your testing. Which assertions were checked? Did your verification environment identify the bugs that we placed in the design? If so, how? If not, why not? What were the bugs? Your report should summarize your test coverage. Include conclusions, lessons learned, next steps if you had more time. List the contributions of each team member.
- All your source code including any scripts, programs, etc. you may have used to produce stimulus/result data for your verification effort. Source code should be structured with meaningful signal names and comments as appropriate. Each file should start with a header including the author and a short description of what the code in the files does. The header should acknowledge the source of any work that is not your own. You do not need to submit the Pipelined Picoblaze .sv (and .svp) files unless you made changes to them.
- If you have written Makefiles, scripts, etc. please include them so that somebody could reproduce your testing and your results. For example, if you use Assembly language test programs for unit level test/CPU level tests they could/should have their own Makefile.

Grading:

- (30 pts) Your Verification Plan. This plan will most likely be incomplete since it needs to be in place to implement your verification strategy.
- (40 pts) The final project presentation. Be sure to walk through your project, describing your test strategy and verification environment. Discuss bugs your testing caught and the way by which they are caught.

  Make sure you practice your presentation before you present. 20 minutes may seem like forever, but it is a short time for a project of this magnitude presented by 3 verification engineers (that's you!). Make sure all team members present and all speak about their portion(s) of the project. Encourage questions.

- (25 pts) The quality of your Verification Report. Your report should effectively describe your verification methodology and summarize your results. Draw conclusions on what you

have accomplished and learned about your verification strategy and implementation. Start with your Verification Plan.

- (5 pts) Your source code should be organized with meaningful signal names. Make use of comments to describe aspects of your code that would be hard to understand from just reading the code. Make use of SystemVerilog constructs such as typedefs and structs, packages, assertions, and SystemVerilog verification constructs. Much of this grade will be based on the readability of your code; that is, can we get the gist of what you did by reading through, not studying the code in detail.
- (Up to 5 extra credit points) You can get extra credit by doing more than we asked for and/or for doing an excellent job. Extra credit is just that, though- Extra. You need to have a good verification strategy, a good plan, and a good presentation to be eligible for extra credit.

Important dates:

- Wed, 09-Nov: Final project assigned during class
- Tue, 15-Nov: Pipelined Picoblaze final project released to Canvas and GitHub
- Sun, 26-Nov: Verification plan submitted to Canvas by 10:00pm
- Tue, 29-Nov: Release of the “broken” Pipelined Picoblaze encrypted model(s)
- Tue, 06-Dec: Day 1 - Final project presentations (sign up for a time slot)
- Wed, 07-Dec: Day 2 - Final project presentations (sign up for a time slot)
- Thu, 08-Dec: Final project deliverables due to GitHub and D2L by 10:00pm