Gisselquist Technology, LLC

0. Preface

Daniel E. Gisselquist, Ph.D.

Lesson Overview

Objectives

- Understand the Course philosophy
- Check the Pre-requisites
- Getting Started

Clear your desk, it's time to get started!

Verilog is a large language

- Only some verilog is necessary for design
  - Simulation verilog gets confused with synthesizable verilog
  - Programmers turn Verilog into a programming language
    It's not.
  - Verilog testbench language is inadequate for bug finding when compared with formal methods
    We'll be using `SymbiYosys` for formal verification
  - Verilog testbenches are a poor substitute for a good simulation language, such as C++
    We'll be using `verilator` and `C++` for simulation

A better solution is needed!

- Let's call it `V--`

Verilog is a large language

- Only some verilog is necessary for design
- We'll use synthesizable code only
  - No `A <= #10 B;` statements
  - No `@posedge` statements
  - No **$display**, **$monitor**, or **$final** statements, etc.
  - No `'x` values
  - Only toplevel ports can be **inouts**
  - We'll use restricted forms for multiply and memory
  - Avoid teaching loops as long as possible
  - We will use **initial** statements

    Appropriate for FPGA's

Verilog is a large language

- Only some verilog is necessary for design
- We'll use synthesizable code only
- Safe style guide
  - One clock (initially)
  - No logic generated clocks
  - We'll use **initial** statements
  - Resets values must match initial values

Verilog is a large language

- Only some verilog is necessary for design
- We'll use synthesizable code only
- Safe style guide
- Co-simulation is a *must*
  - External hardware peripheral simulations will be built in C++
  - Goal is to create a design that looks, acts, and works as though it were on the FPGA

Verilog is a large language

- Only some verilog is necessary for design
- We'll use synthesizable code only
- Safe style guide
- Co-simulation is a *must*
- Formal verification is great for bench testing
  - We'll scratch the surface here

Verilog is a large language

- Only some verilog is necessary for design
- We'll use synthesizable code only
- Safe style guide
- Co-simulation is a *must*
- Formal verification is great for bench testing
- Verilog instruction must include
  - Formal methods, and
  - Simulation

  . . . *from the beginning!*

Which board?

This course is intended to be board-agnostic

- We'll cover the basics and the mechanics
- We'll use Verilator extensively
- You don't need a board to take this course
- You may enjoy the course more with a board
- "Board bonus chapter appendices" may eventually accompany the course

Minimum Board

Course designs depend upon a minimum capability

- One button/switch, one LED
- Serial port, both transmit/receive

Clear your desk

If you have an FPGA board, then

- Find and download the schematic, . . .
- The data sheets for all of the components, . . .
- The board vendor's master constraint file, and
- The board vendor's demo code

Put these files in a project reference directory

Do this before any project with a new FPGA board!

Test your board

Your board vendor should provide you with

- A demonstration design, and
- The instructions necessary to build and load it

This design should verify that your hardware works
If you will be using hardware for this course, please verify that your hardware passes this test first

Conclusion

- Digital design can be hard, let's not make it harder
- Teach debugging tools with the language
- Are you ready to learn?