Review of Digital Design

- Why?
  - You never learned how - traffic light controllers
  - You have forgotten or are rusty
  - You have bad habits (ad hoc, asynchronous design, old technology)
  - You need a method that will work for more complex designs

- We will learn on simple designs, then add complexity
Review of Digital Design

▶ Why not jump into System Verilog (SV) now?
▶ If you can’t correctly do digital design, SV only helps you make mistakes more quickly!
▶ SV describes circuits, it doesn’t design circuits. (coding ≠ design)
▶ SV ≠ design tool. Block, timing and state machine diagrams are.
▶ Typically, coding begins once the design is finished.
▶ We use SV templates to implement our designs. Writing SV code should be a nearly mindless translation step.
▶ Clever code ≠ clever circuit design. ”Optimizing” SV code is useless in the conventional sense except for simulation. Synthesis probably ignores your clever code anyway!
▶ Good designs stem from good architecture (structure), not clever code.

A problem occurs when we think that using System Verilog is all about ”coding up” designs. Coding is not the emphasis. Instead, the focus is on describing an already defined hardware structure. The design process is not ”coding”. Design is a separate, earlier step, done with different tools.
Review of Digital Design - Top Down Methodology

- **Top Down Characteristics**
  - Aims for a detailed, complete, system-level understanding prior to implementation
  - Moves from biggest picture to smallest, general methods towards specific implementations
  - Recursively breaks down complex pieces into well understood pieces. (brain-sized)

- **Top Down Methodology**
  - Establishes an overall system-level view of how things work together without going into implementation details. Uses "blackbox" abstractions. "I know what it does but presently don’t know how.”
  - Each part in the system is refined by adding more detail.
  - Each sub-part is repeatedly refined in the same way until all the parts are defined enough to allow validation.
Review of Digital Design - Bottom Up Methodology

- Bottom Up Characteristics
  - Origins are in OOP, C++, etc.
  - Emphasizes early coding and testing

- Bottom Up Methodology
  - Individual pieces designed and tested fully
  - Individual pieces are connected together to form bigger parts
  - Keep connecting parts until you have a complete system

A bottom up methodology still requires that you clearly understand what the eventual system "looks" like. Otherwise, you create pieces of hardware that work in a vacuum but don’t support building the intended system.
Review of Digital Design - Top Down/Bottom Up

- **Top Down**
  - + Can handle really big, complex designs
  - + Simultaneous, independent efforts possible
  - + Serious architectural bugs flushed out early
  - + Reusable modules identified early
  - − Delays functional testing
  - − Some decisions can’t be made w/o some implementation
  - − Can lead to duplication of effort

- **Bottom Up**
  - + Early testing of individual parts
  - + May identify ”crux” issues more quickly
  - + Design reusability
  - − Modules created early may not fit final system (repeated redesign)
Limits to Concurrency
- 9 women can’t have a baby in 1 month
- Communication goes up quadratically with bigger group
- At some point, a bigger group gets work done more slowly
- Just one linear process will cap maximum speedup

Well Defined Interfaces
- Partitioned designs are vitally dependent on well defined interfaces
- Interfaces must be pendantsly defined
- Designs usually break at the interfaces
- Good partitioning (architecture) naturally limits interfaces