Digital Design Methodology - Top Down

- **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.
Digital Design Methodology - Bottom Up

- 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.
Digital Design Methodology - 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)
Digital Design Methodology - Design Teams

- 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 pendently defined
  - Designs usually break at the interfaces
  - Good partitioning (architecture) naturally limits interfaces