What is an HDL? (Verilog/VHDL)

- A Language to describe, simulate, and create hardware
- Cannot be used like typical languages (C and Java)
- Express the dimensions of timing and concurrency
- At RTL, describes a hardware structure, not an algorithm
- At behavioral level, describes behavior with no implied structure
- "If you can’t draw it, don’t try to code it"
HDLs

Why were HDLs developed?

- DOD VHSIC program extended integration and performance of ICs
- Very large, complex, high speed ICs were successfully fabricated
- Schematics were becoming unwieldy quickly
- The number of transistors and complexity exploded
- Needed a way to describe and simulate complex designs
- VHDL considered as an alternative to IC datasheets
- **VHDL:**
  - more difficult to learn
  - strongly typed
  - widely used for FPGAs, military

- **Verilog**
  - simpler to learn
  - looks like C
  - weakly typed
  - 85% of ASIC designs use Verilog

- Once either is used, the other is learned quickly

- System Verilog is Verilog with lots of goodies added from VHDL
Logic Synthesis was a later thought (1985)

- Enabled enormous increases in productivity
- Time to market is critical to survival
- Only about 10 percent of Verilog/VHDL is synthesible
- Remainder is for testbenches, stimulus generation
What are the advantages of using HDLs

- Can express large, complex designs (>10^7 gates)
- Flexible modeling capabilities
- Description can include the very abstract to the very structural
- Can utilize top down or bottom up methodologies
- Complexity hiding by abstraction is natural
What are the advantages of using HDLs (cont.)

- **Productivity!**
  - Logic Synthesis
    - 10-20K gates/day/designer
  - Design changes are fast and easily done (text vs. drawing)
  - Leverage of SW design tools
    - vi, source control, make files, Unix text tools
  - Optimization of designs is easier
  - Exploration of alternative designs can be done quickly
  - Easy to trade off time and complexity (speed vs. area)
What are the advantages of using HDLs (cont.)

- **Reusability**
  - Packages, libraries, designs all can be reusable
  - Vendor and technology independence
    - CMOS, ECL, GaAs, NEC, TI, TSMC,... same code
- **Documentation**
  - Textual documentation is part of the code, not a separate document
- **Standards**
  - There is no such thing as a schematic standard
  - Strict standards promote clear and consistent delivery of design intent
What does Verilog HDL look like?

```verilog
module adder(
  input clk, //input clock
  input reset_n, //reset async active low
  input first_select, //on first data item use this
  input rd_fifo, //enable for ff
  input [7:0] data, //data in
  output reg [11:0] acc_out //data out of accumulator
);

always @ (posedge clk, negedge reset_n)
    if (!reset_n) acc_out <= 12’h000;
    else
        if (rd_fifo == 1’b1)
            if (first_select == 1’b1) acc_out <= data;
        else
            acc_out <= acc_out + data;
endmodule
```
What structure does that code infer?
What does that code synthesize to?
What can’t an HDL do?

- Can not make architectural tradeoffs for you, it can help however
- Does not relieve you of understanding digital design
- Guess what it is that you want something to do
- If you can’t draw the structure you are looking for, stop coding