VERILOG 1:

AN OVERVIEW

Verilog in This Course

- On one hand...
  - The important content of the course is core digital systems design principles
  - Verilog is simply the language by which you communicate your design to the simulator and synthesis tool
  - The core principles apply regardless of the HDL language you use
- However, at the same time...
  - Almost all of your submitted work in this class will require you to write good quality verilog
  - You must have a strong working knowledge of the *basic features* of the language
  - Every hardware description method/language will have many of verilog's features

Verilog is a *Hardware* Description Language (HDL)

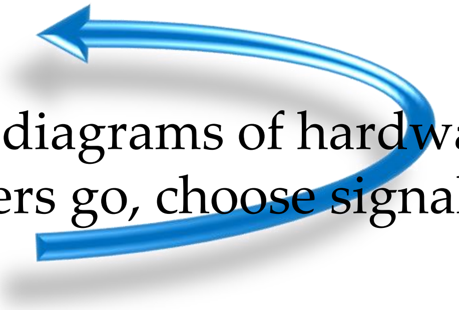
- You'll design far better hardware if you think of it differently than a standard programming language
- A "standard programming language" such as C, C++, python, java, etc.:
  - Is a way to code an *algorithm* or *is a way to calculate a result*
  - Is written by a *software writer*
  - Often results in a more elegant solution when the programmer uses finer features of the language
- On the other hand, a hardware description language:
  - Is a way to describe *hardware*
  - Is written by a *hardware designer*
  - Results in a far better solution when the designer uses only the most basic features of the language

Write Clean ”Synthesizable” Verilog

- Why are hardware designs made with simpler language features better?
  - Synthesis tools will have fewer opportunities to interpret (destroy) the circuit you really want
  - Less-commonly-used CAD tools (such as place & route, design rule checking (DRC), layout versus schematic (LVS), formal verification, automatic test pattern generation (ATPG), etc.) often do not work properly with uncommon language constructs
- In this class:
  - To engrain good practices, use only the simplest constructs shown in handouts, lectures, and labs to receive full credit
  - Corollary: do not copy code off the web or other outside sources!
- No prize for dense cryptic code except a better chance for non-functional time-consuming designs

The Design Process

- Design process
  - Think
  - Draw diagrams of hardware, figure out where logic and registers go, choose signal names carefully
  - Think



  - Then…
    - Write verilog
    - Test it
    - Optimize it

"A man who carries a cat by the tail learns something he can learn in no other way" -Samuel Clemens

Writing Efficient Code

- Think about what kind of hardware will result from some particular code even if the code looks simple

```
if (x<y) begin
   a = 4’b0001;
end
```

- An inequality requires a slow carry-propagate subtractor. It is simplified since the *sum* bits do not need to be computed, but the slow *carry-out* of the entire adder is needed
- Think about the hardware you want and what you’ll get with the verilog you write
  - Example: A 4× upsampler could be built with a 4-input mux; why not use a much-simpler AND?

Verilog Simulator Tools

- Mentor Graphics
  - Modelsim
- Cadence
  - NC Verilog – verilog simulator
  - Simvision – waveform viewer
  - Verilog XL – perhaps no longer supported? slower run time, but faster start up time as it spends less time compiling before running
    - Often gives different (helpful!) and slightly more descriptive error messages than NC Verilog
- Synopsys
  - VCS – similar to NC Verilog
  - Virsim – waveform viewer and environment
- Many others…

Verilog vs. VHDL

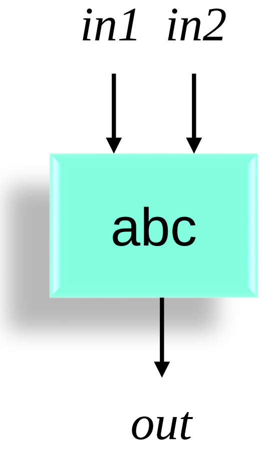
- Verilog
  - Invented in 1983 at Automated Integrated Design Systems (later Gateway Design Automation) which was purchased by Cadence in 1990. It was transferred into the public domain in 1990 and it became IEEE Std. 1364-1995, or *Verilog-95.*
  - Later versions include
    - *Verilog-2001* aka *IEEE 1364-2001*
      - Added **`signed`** (2's complement) arithmetic support
      - Added support for combinational **`always @(*)`**
    - *Verilog-2005* aka *IEEE 1364-2005*
  - Strong similarities to C
  - Seems to be more commonly used in high-tech companies

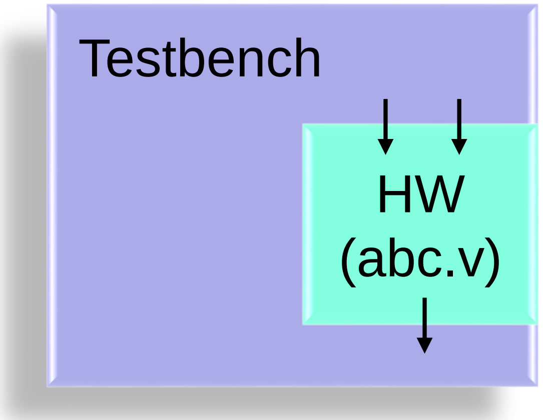
Verilog vs. VHDL

- VHDL (VHSIC Hardware Description Language)
  - Published in 1987 with Dept. of Defense support as IEEE Std. 1076-1987. Updated in 1993 as IEEE Std. 1076-1993, which is still the most widely-used version.
  - Later versions in 2000, 2002, and 2008.
  - Strong similarities to Ada
  - The only(?) HDL language used in government and defense organizations, and seems to be more often used in east-coast companies. Widely taught in universities ↔ used in textbooks—who started it?!

Hardware Verilog vs. Testing Verilog

- We make a strong distinction between "Hardware" verilog blocks vs. "Testing" verilog blocks
  - Hardware verilog
    - Only the simplest, cleanest code
  - Testing verilog (referred to as a "testbench")
    - Any code style is fine
    - Having said that, clear code is always best


in1
in2
abc
out


Testbench
HW
(abc.v)