Lab 1

- Tools change as time goes on.
- Not all commands/procedures in the lab manual will work exactly as written.
- Lab printer generally won’t work. Do not depend on it.

General Course Policies

- Attendance is never taken
  - No penalty for not showing up if lectures are not worth your time
  - No credit for showing up if you don’t learn anything
- If you do come to class, arrive on time
  - Late arrivals are disruptive to those who do arrive on time
- Homework must be turned in on time for credit

Lab Attendance

- Lab attendance is NOT optional.
- Everyone must attend every lab session and be there on time.
- You must demonstrate that you yourself are doing the lab, not copying from someone else.

Co-requisites

- Lab and lecture are co-requisites.
- If you do not already have degree credit for one, you will be dropped if you are not enrolled in the other.

Turn ‘em OFF!

It seriously annoys your instructor, not a Good Thing.

Weekly Schedule

- Weekly schedule is a rough guideline. Do not expect it to be rigorously adhered to.
Midterm Exams

- Exams will always be on lab days.
- Lab will be cancelled on exam days.
- Midterm exams are tentatively scheduled for weeks 7 and 12.

EDA: Electronic Design Automation

The process of using computer-based software systems to design very large-scale integrated (VLSI) circuits.

All modern integrated circuits are designed with EDA tools.

This course uses Verilog/SystemVerilog for hardware description and Synopsys VCS for verification.

Current HDL Usage

- All current digital IC development is done with either VHDL or Verilog/SV.
- Other scripting languages (perl, make, Tk/tcl) may be used to supplement them.
- All FPGA, ASIC and 3rd party tool vendors (Synopsys, Cadence, Mentor Graphics, etc.) support both Verilog/SV and VHDL.

SystemVerilog

- ECE526 includes SV constructs useful for circuit design.
- SV is a superset of Verilog.
- It’s more for verification than design, but it has some useful features for design, too.
- It’s a huge, all-encompassing language. We will not cover all of it.

What is a Hardware Description Language (HDL)?

- Computer-based language
  - Syntactically similar to programming languages
- Model, represent, and simulate digital hardware
- Hardware concurrency
- Parallel activity flow
- Semantics for signal value and time
- Special constructs and semantics

Examples
- HILO2, HARDWARE C, AHDL (U of A 1970), ISPS (CMU, 1971)
- VERILOG
- VHDL (Very High Speed Integrated Circuit Hardware Description Language)

Why Use an HDL?

- As the complexity of systems increases, it becomes more difficult to design directly on hardware.
- Exploring different design options is easier and cheaper because you only need to change the HDL description. It is much easier to change the description than to reconfigure the prototype.
- You can try different design options quickly and easily. The time to fix a design problem is reduced and so is the cost.
Test Early and Often

Finding defects early is way cheaper than finding them after they are out in the field. Image © Texas Instruments.

Billion Dollar Error

Intel Finds $1B Design Flaw

SANTA CLARA, California (AP) -- Intel Corp. on Monday said it has found a design flaw in a recently released chip, and is working with laptop makers to replace affected computers.

Transistor Density

• First microprocessor (circa 1971) had 2,300 transistors.
• It was done by a team of four in about four months.
• That’s less than 150 transistors per engineer per month.

Density Increase

• Current state of the art in processors is three billion transistors (2010: nVidia Fermi GPU).
• At 150 transistors per engineer per month, something like 18 million man-months.
• Design at the gate level: about 10 transistors/gate.
• A team of 100 could then get a processor done in a mere 15,000 years.

Area vs. Operating Frequency

One HDL encoding can produce many designs through logic synthesis.

Popularity of Verilog HDL

• It is a general-purpose hardware description language that is easy to learn and use.
• It is similar to the C programming language.
• It allows for different levels of abstraction to be mixed in the same model.
• Most popular logic synthesis tools support Verilog HDL.
• All fabrication vendors provide Verilog HDL libraries for post logic synthesis simulation.
• The Programming Language Interface (PLI) allows the user to write custom C code to interact with the internal data structures of Verilog.
Typical Design Flow

- The behavioral description is manually converted to a Register Transfer Level (RTL) description in an HDL.
- Designer has to describe the data flow that implements the desired digital circuit.
- Logic synthesis tools convert the RTL description to a gate-level netlist (a description of the circuit in terms of gates and connections between them).
- The gate-level netlist is input to an Automatic Place and Route tool to create a layout.
- The layout is verified then fabricated on chip.

HDL Design Flow

Terms and Definitions

- **Hardware description language (HDL)**: A programming language that can describe the functionality and timing of hardware circuits.
- **Simulator**: Software which reads the HDL and emulates the hardware described by the HDL.
- **Bottom-Up design flow**: A design methodology in which you build the low-level components first and then connect them to make large systems.
- **Top-down design flow**: A design methodology in which you define the system at a very high level of abstraction and refine it at the lower levels of abstraction by partitioning the design into logical, functional, or physical units.

Key Features of HDL's

- Typically an HDL contains some high level programming language constructs along with constructs to describe the connectivity of hardware design.
- An HDL allows you to describe the design at various levels of abstraction using structural or behavioral constructs.
- An HDL allows you to describe the functionality of hardware along with its timing constraints.
- Concurrency is the ability to perform multiple tasks at the same time. Typically, programming languages are not concurrent, but in hardware, a number of operations happen at the same time. Thus, an HDL must be concurrent.
- Typically, programming languages have no concept of time. In hardware there are delays associated with going from an input to an output. An HDL allows you to model these delays because it has a concept of time.
Simulation Algorithms

• **Time Driven**
  
  — Each circuit element is evaluated at each time point, producing a new circuit state at that point.
  
  — Inefficient, because at any time only 2 to 10 percent of elements in a circuit need to be evaluated.
  
  — The change of the value of any of the variable will result in the calculation of the logical value at all points.

• **Event Driven**
  
  — Changes in circuit state are recorded. Only those elements that might cause a change in circuit state, during time, are simulated. The simulation propagates values forward, through the circuit, in response to input pin events.
  
  — Most practical and widely used simulation algorithm.
  
  — Efficient, because it “evaluates when necessary.”
  
  — Assume that at time “t=T”: A=1, B=0 and C = 1. Now if at time “t=T+1” only A changed to “0” this will require the calculation of only the output of this gate.

• **Demand driven**
  
  — Further refines “evaluates when necessary.”
  
  — Evaluates sets and gates only when their values are needed to provide simulation output.
  
  — Propagates requests for simulation values, backwards through circuit and time.
  
  — Assume that at time “t=T”: W= 1. Now, if at time “t=T+1” W does not change then the changed of any of the other variable will not require any calculation.

• **Cycle-based**
  
  Collapse all logic between registers.
  
  Only calculated final values.
  
  Intermediate timing and delay data are lost.
  
  Fastest, but not suitable for all simulations.

**Cycle Based Algorithm**

- \( S1 = Q1 \& 1 \)
- \( S2 = S1 \| 0 \)
- \( S3 = S2 \^{0} \)

\( \rightarrow S3 = Q1 \)

Only calculate final value at clock edge, not all intermediate values at the time at which they transition. Timing may be verified with a static timing analyzer.
Combined Algorithms

- Perfectly synchronous circuits may be effectively simulated with cycle-based simulation combined with STA.
- Real designs tend not to be totally synchronous (though ECE526 designs will be).
- Sophisticated simulators combine algorithms: parts that are suitable for cycle-based use cycle-based, parts that are not use event-driven.

Time Wheel in Event-Driven Simulation

- When the simulator compiles its data structures, it creates the initial queues (time=0) for the time wheel based on the HDL.
- The simulation starts when the current simulation time is 0.
- When all the events scheduled at time 0 are executed, the simulation time advances to the next time step.
- Events at each time step can append new events to event queues at a later time.

The time wheel can only go forward.
Time advances only when every event scheduled at that time is executed.

Different Levels of Abstraction

- Architectural/Algorithmic
- Register Transfer level (RTL)
- Gate
- Switch
Different Levels of Abstraction

- Architectural/Algorithmic
  - The system is described in the terms of the algorithms it performs.
  - The aim is to study the data flow of the system and potential bottlenecks.
- Register Transfer Level (RTL)
  - Describes the flow of data and control signals within and between functional blocks.
  - Schedules assignments at clock edges.

Design Methodologies

- Bottom-up design flow
  - Start with small functions
  - Build up a hierarchy, eventually creating the top level design.
- Top-down design flow
  - Start with an overall design specification.
  - Add detailed functions to make it work.

Verilog is Flexible

- There are many ways to code a Verilog project.
- Standards have developed over the years.
  Examples presented here frequently show variations on what CAN be done and do not always correspond to best industrial practices.

Different Levels of Abstraction

- Gate
  - Interconnection of logic elements (or gates) to check functionality, performance, or the timing of design.
- Switch
  - Describes logic behavior of transistor circuits.
  - Evaluates conflicts caused by bidirectional pass transistors, signal strengths of multiple elements driving a net, and so on.