ECE 351

Day 1

Professor Greenwood

* The “teaching approach” used by this professor is “Approach #2”
* When designing an FPGA product, your best friend will be the Applications Engineer at the Vendors facility.
  + To decrease time to market.
* Timing closure is such an important milestone that people at Intel actively celebrate it.
  + After timing closure is met, the design can be shipped to manufacturing to develop it.
* What to do when designs won’t work (meet timing constraints). Oops lost the slide…
* ASIC: Application specific integrated circuit.
  + Designed for the specific application and that specific application only.
  + The only thing that goes on silicon is exactly what you need.
    - D/A device, FIFO, etc.
  + Need to work with ASIC vendor who is going to be fabricating the work.
    - However, with an FPGA vendor the process is different.
    - The design approach with an FPGA has faster turnaround time.
* ASSP: Application specific standard product.
  + Semi-custom chip for a specific purpose which can be tailored to individual customer needs.
* 2 FPGA applications
  + Prototyping
    - You want to use an ASIC in the product you want to sell.
    - This is very time consuming and requires a lot of effort.
    - Hardware and Software design are consecutive.
      * Hardware platform must be ready for software engineers to debug.
      * This means the software guys need a working chip.
    - To allow Software development in parallel of Hardware ASIC design, FPGA “Emulation” is used to simulate the design of the Hardware.
    - This is the essence of prototyping an ASIC or ASSP design.
  + Target device
    - Your product uses an FPGA as the target device, no ASIC hardware design.
* Primer: Design Views
  + Technology Independent Hardware Description Language.
    - VHDL and Verilog are purely functional, all they describe is the logic.
  + Technology Dependent Design:
    - Based on Logic Family: Is it 3.3V CMOS, Bipolar, etc.
* Primer: Describing Designs
  + HDLs
  + Schematic Capture
    - Uses a point-and-click system for designing using a schematic using logic devices and drawing point to point connections.
* RTL: Register Transfer Level
  + Banks of Registers, Combination Logic, banks of registers, repeat.
  + Inputs & outputs defined.
  + Registers serve as memory elements.
  + Combination logic between register processes the data.
  + Circuit specified by operation and data transfers between banks of registers.
  + Defined timing: Operations occur at specific times.
* RTL: The RTL Model
  + Combinational logic implements functions
    - Reg2 = function1(Reg1)
    - Reg3 = function2(Reg2)
  + RTL model shows data movements but no details about the “Control Structure”
    - Control structure generates signals such as decoder-enables or mux-select line inputs.
  + “RTL Code”: You have a HDL where the coding style follows the RTL model.
* The Technology Independent Design Process used in class.
  + The automation of circuit design.
  + Component Library, Verilog Source Code, and User Constraints are inputs.
  + The “SYNTHESIZER” accepts these inputs.
  + The Circuit Design “Netlist” is the output of the SYNTHESIZER.
* Tradiditional Simulation Process
  + HDL Description Language is input to the Simulator
  + The Simulation Test Bench specifies the test patterns applied and how often you apply test patterns which is also input to the Simulator.
  + The Simulator outputs timing diagrams.
* Logic Synthesis Process
  + The output is a netlist Description (Verilog, VHDL, EDIF)
    - Technology Dependent.
    - EDIF is a standardize format for compatibility across software vendors (i.e. Place and route software which follows the Synthesizer).
* Netlist Schematic
  + Circuit Reference Designator
    - “U” is used for integrated circuits, “R” for resistors, “L” for inductors, etc.
    - Nets also have CRDs using “n”
    - Example.
      * Terminal Z of U87 to Terminal A of U89

net 275

(joined (portRef A (instanceRef U89)) (portRef….

* Design Flow: Generic steps from design specification to the physical part.
  + Design Specification
  + Behavioral Description
  + RTL Description (HDL)
  + Functional Verification and Testing
  + Logic Synthesis/ Timing Verification
  + The netlist (Gate-Level netlist)
    - Class stops here.
  + Logical Verification and Testing
  + Floor Planning Automatic Place and Route
  + Physical Layout
  + Layout Verification
  + Implementation.
* Programming languages
  + Alphabet
  + Syntax
  + Semantics
* Programming Languages: Alphabet
  + The symbols that can be used in the program.
  + In C
    - Lowercase letters, Uppercase letters, Numbers, and special symbols.
    - Uses the same alphabet as Verilog.
* Programming Languages: Syntax
  + The rules for combining alphabet characters into valid program statements.
  + In C
    - i++ is ok.
    - +i+ is not ok.
* Programming Languages: Semantics
  + The meaning of a valid syntactically correct program statement.
  + In C
    - i++; is the syntax
    - semantics: increment i by 1.
* Verilog Code Example: Multi-line comment
  + /\*

\*

\*

\*/

Or // for single line comments.

* Verilog Code Example: Operators
  + Unary operators.
    - A = ~B // A = Bn
  + Binary operators.
    - Y = A + B;
  + Ternary operators.
    - A = B ? C : D;
* Verilog Code Example: