EE 4755—Digital Design Using Hardware Description Languages

Basic Information

URL: http://www.ece.lsu.edu/v

Offered by:

David M. Koppelman, Room 345 ERAD Building

578-5482.

koppel@ece.lsu.edu, http://www.ece.lsu.edu/koppel/koppel.html

Tentative office hours: M-F 15:30-16:30.

Formal Prerequisite

EE 3755 (Computer Organization).

Informal Prerequisites (What you really need to know.)

C (The computer language.)

Digital hardware design: Should be able to answer these:

Is a ripple adder something to get excited about?

What does it mean to clock a register?

Your part of the design carries the critical path: your fault or your forte?

Graded Material

Midterm Exam, 35%

Fifty minutes, open notes.

Final Exam, 35%

Two hours, open notes.

Yes, it's cumulative.

Homework/Projects, 30%

Written and computer assignments.

Lowest grade or unsubmitted assignment dropped.

Electronic Design Automation (EDA)

Electronic Design Automation (EDA) (Short Definition)

The use of software to automate electronic (digital and analog) design.

Electronic Design Automation (EDA) (Longer Definition)

Electronic design in which

the design is written in a *hardware description language*

possibly consisting of components from a vendor's *IP library*

the functionality of the design is verified by *simulation*

the correctness, testability, and compliance of a design is evaluated by software

and the design is converted to a manufactureable form using *synthesis tools*.

Hardware Description Languages

Hardware Description Language:

A language used for describing the structure of hardware and how the hardware should behave. Examples include SystemVerilog and VHDL.

The Most Popular Hardware Description Languages

Verilog and *SystemVerilog*

(Verilog was replaced by SystemVerilog in 2009.)

C-Like and C++-Like Syntax

Originated in industry in 1984, now IEEE standard.

VHDL

Ada-Like Syntax

Originated as a DoD project, now an IEEE standard.

Verilog vs. VHDL

Good News: Many tools support both.

SystemVerilog covered in this course.

History of Verilog and SystemVerilog

Verilog developed by Gateway Design Automation in 1984, later bought by Cadence.

Became an IEEE standard 1995: *IEEE 1364-1995*.

Major update in 2001: *IEEE 1364-2001*.

Minor update in 2005.

Also in 2005, related language *SystemVerilog* was standardized as *IEEE 1800-2005*.

In 2009 Verilog and SystemVerilog merged: *IEEE 1800-2009*.

Minor update in 2012: *IEEE 1800-2012*.

VHSIC Hardware Description Language (VHDL)

Ada-like syntax. (Ada is a DoD-developed language for large embedded systems.)

Developed as part of U.S. Department of Defense (DoD) VHSIC program in 1983

Became IEEE standard 1076 in 1987.

Some believe that VHDL is harder to learn than Verilog.

At most one or two lectures on VHDL.

Design Flow

Design Flow:

The steps used to produce a design, from initial design entry to the generation of the final manufacureable form. Describes which programs will be used, when they will be used, and how they will be used.

EDA tool vendors usually provide design flows that show how their products can be used.

Companies develop design flows that are used to produce their designs.

A simple design flow is described below.

Informal Design Flow

♢ Enter design. (Use your favorite text editor and HDL.)

♢ Enter *testbench* for design.

The testbench checks correctness.

This is very important—you never want to say "I thought it worked."

♢ Run simulation to verify correctness.

♢ Use waveform viewer and other tools to find bugs. (If any.)

♢ Run synthesis program.

Synthesis reports indicate area and timing.

Not satisfied? Go to Step 1.

Otherwise, *tape out*, download to FPGA, etc.

Demonstration

Demonstration: look at different possible integer multiplier designs.

Multiplier Module

Inputs: multiplier, multiplicand.

Input: clock (sequential versions).

Output: product.

Parameter: width (number of bits each in multiplier and multiplicand).

What makes the multiplier interesting.

It's used in many applications.

There are many cost and performance tradeoffs.

Multiplier Versions

`mult_behavioral`

Rely on EDA software to make all the right decisions.

Simple Design Flow

Simple Design Flow (Simple Flow, for short)

Three easy steps (not counting step zero).

Used to describe the major steps in a typical design flow.

List of Steps in Simple Design Flow

Simple Flow Step 0: Goal Determination.

Simple Flow Step 1: Design Capture

Simple Flow Step 2: Behavioral Verification

Simple Flow Step 3: Synthesis and Timing Verification

Simple Flow Step 0:

Start with: an idea for a new chip.

Goal: a box full of the new chips.

Simple Flow Step 1: Design Capture

Using the back of an envelope or some other suitable medium ...
... develop a rough draft of the design.

Using a text editor ...
... write a *Verilog description* of the design.

Using a text editor ...
... write a *Verilog description* of a *testbench* used to test the design.

The testbench generates inputs for the design and verifies the design's outputs.

Simple Flow Step 2: Behavioral Verification

Using a *simulator* and *waveform viewer* . . .
. . . check if design passes testbench tests . . .
. . . and if not, debug.

Waveform viewer is sort of a virtual logic analyzer, can view signals on any part of design.

Simulator output includes messages generated by behavioral code . . .
. . . including "pass" or "fail" message produced by testbench.

Using text editor . . .
. . . fix bugs, and tune performance.

Simple Flow Step 3: Synthesis and Timing Verification

Using synthesis programs . . .
. . . generate *design database*.

> Design database has information needed to fabricate the chip . . .
> . . . and to perform simulations with accurate timing.

Simulate using design database to verify that timing is acceptable . . .
. . . if timing is not acceptable edit the Verilog structural description and repeat steps above.

Using the Internet, E-mail design database and credit card number to fab.

After a few weeks, get parts back in mail.

Topics Covered in This Course

- Coding in SystemVerilog.
- Writing structural and synthesizable descriptions.
- Writing testbench code.
- Using simulation, waveform viewers and similar tools.
- Synthesis.
- Using synthesis tools.

EDA (Electronic Design Automation) Tools

Programs to support design automation.

These include SystemVerilog simulators, synthesis programs, design rule checkers, etc., etc., etc.

Major EDA Vendors

- Synopsis
- **Cadence Design Systems**
- Mentor Graphics

Course Will Use Products of Cadence Design Systems

ECE is a member of Cadence's University Software Program.

Software would normally cost well over 100 k$.

For Simulation: *Cadence Incisive Enterprise Simulator*

A collection of programs including:

- *ncvlog* – Verilog simulator.
- *simvision* – Waveform viewer (to view sim results).
- *irun* – Convenient front end to other programs.

For Synthesis: *Encounter RTL Compiler*

- *rc* – Front end for synthesis tools.

Design Capture

Design Capture:
Entering a design in electronic form.

Start: Idea in engineer's head, scribbles on back of envelope.

Finish: Design in electronic form readable by some EDA tools.

Design Capture Methods

- Schematic Capture

  Enter design using GUI (graphical user interface) schematic editor.

  Fun and easy for beginners but tedious for all but small designs.

- Finite-State Machine Editors

  Programs meant for designing FSM, to be part of larger design.

- Hardware Description Languages

  Like any programming language ...
  ... design entered using standard or specialized text editor.

HDL Descriptive Styles

Descriptive style refers to a set of rules that a description adheres to.

HDL's are used to write a hardware *description* or *model* (don't call it a program).

Descriptive Styles

Structural: how parts are connected together, like a schematic.

Behavioral: what hardware is supposed to do.

Register Transfer Language (RTL): a form in which registers are explicit. Covered later in semester.

Intent of structural code is the description of hardware.

Intent of behavioral code can be

description of hardware, for use by a synthesis program

testbench, used to verify correctness of other descriptions

simulate a part not yet designed in detail

Synthesis Design Target

Design Target:

The type of device to be manufactured or programmed. Synthesis programs generate output for a particular design target.

Design Targets

- *Programmable Logic Array (PLA)*

  Chip that can be programmed (once) to implement a logic function.

  Usually programmed at the factory.

  PLAs might be used in prototypes or when only a few parts are needed.

- *Application-Specific Integrated Circuit (ASIC)*

  A fully custom chip.

  Usually the fastest design target, can have the most components.

- *Gate Array*

  A chip full of gates manufactured in two steps:

  First generic layers containing gates are fabricated . . .
  . . . but gates are not connected to each other.

  Later, metal layers connecting gates are added.

  Designer using gate arrays specifies only metal layer.

  Since gates fabricated in advance time is saved.

- *Field-Programmable Gate Array (FPGA)*

  A chip full of logic whose connection and function can be programmed and later re-programmed.

  Many FPGA vendors provide EDA tools for their products.

Typical Synthesis Steps

Start With:

Choice of Design Target

Type of target: FPGA, ASIC, etc.

Manufacturer and family.

A synthesis program or programs.

Behavioral or Structural Description

Functionality has been verified by simulation.

Behavioral description (if used) follows synthesizebility rules specified for synthesis program.

Major Synthesis Steps (Summary)

Synthesis of technology-independent gate-level description.

Map gates and modules to technology-specific versions.

Place and route.

Major Synthesis Steps (Details)

Synthesis of *technology-independent* gate-level description.

Synthesis program infers registers and minimizes logic.

Registers aren't explicitly declared (even though it will appear otherwise) ...
... so synthesis program must determine (infer) where they are needed.

Because (most) synthesis programs minimize combinational logic ...
... descriptions should be written for clarity.

Output of this step is purely structural code ...
... consisting of gates and standard modules (*e.g.*, for arithmetic), and library modules.

Based on output, designer might tweak design or give hints to synthesis program.

Place and Route

Placement is the determination of the physical location of a part.

Routing is the determination of paths for wires interconnecting parts.

Output of place-and-route step:

Timing information (since technology and wire lengths are known) which may be ...
... *backannotated* (written into) the original behavioral description.

Behavioral descriptions re-simulated to see if they meet timing criteria.

For FPGAs, code to program the devices.

For ASICS and gate arrays, ...
... a design database to *tape-out* and send to a fab.

Fabrication facilities apply additional steps, not covered here.