Logistics

- T-Th 5:15-6:35, WEB 2230
- Instructor: Erik Brunvand
  - Office: MEB 3142
  - Office hours: After class and by appointment
- TAs: Daniel Khoury and Sarvani Kunapareddy
  - Office hours in the CADE lab
  - Times and days TBA

Canvas page will be course home page

Contact:
- We’ll send messages / announcements through the Canvas interface
- To send email to the instructor and TAs, use
  
  teach-6710@list.eng.utah.edu
Textbook

Principles of CMOS VLSI Design
Weste and Harris
(4th edition)

CAD Manual

Describes in detail how to use the CAD tools
Tutorial in nature

Based on v5 of the Cadence tools
Revisions for v6 available through the Canvas page
Class Goal

- To learn about modern Digital CMOS IC design
- Class project – teams will build moderate sized chip
  - Each team will develop a cell library for their project
  - We’ll form teams in a few weeks
- These chips can be fabricated through MOSIS
  - Chip fabrication service for small-volume projects
  - Educational program funded by MOSIS

Class CAD/EDA Tools

- We’ll use tools from Cadence and Synopsys
  - These only run on Linux in the CADE lab, so you’ll need a CADE account
  - I also assume you know something about UNIX/Linux
    - Lots of web tutorials if you need them…
Prerequisites

- Digital design is required! (i.e. CS/ECE 3700)
  - Boolean algebra
  - Combinational circuit design and optimization
    - K-map minimization, SOP, POS, DeMorgan, bubble-pushing, etc.
    - Arithmetic circuits, 2’s complement numbers
  - Sequential Circuit design and optimization
    - Latch/flip-flop design
    - Finite state machine design/implementation
    - Communicating FSMs
  - *Using FSMs to control datapaths*

Recommendation

- Computer Architecture experience is helpful
  - Instruction set architecture (ISA)
  - Assembly language execution model
  - Instruction encoding
  - Simple pipelining

- I assume you’ve used some sort of CAD tools for digital circuits
  - Schematic capture
  - Simulation
Assignment #1: Review

- On the Canvas page is a review assignment
  - If you can do these problems, you probably have the right background
  - If you can’t, you may struggle!!!!!

- Please take this seriously! Give this exam a try and make sure you remember what you need to know!
  - You also need to turn it in next week by Tuesday September 1st
  - Must do this independently, it will be graded

Lab #1: Schematics

- Cadence Composer tutorial
  - Chapters 1-3 in the CAD manual
  - Simple circuit design with simulation
    - Learn basic Verilog for testbench

- Available on the Canvas page

- Due on Tuesday, September 8th, 5:00pm
  - on-line submission on Canvas

START NOW!!!!!
Grading

- Labs (cell designs) & Homework (40%)
- Design review (5%)
- Mid-term exam (15%)
- Final Project (40%)

- See the syllabus (Canvas page) for more details about grading breakdown

Cheating Policy

- In a word: Don’t!
- School of Computing academic misconduct policy is in effect for this class
  - Read the department policy! (linked to the Canvas page)
  - If you haven’t done so, fill out the form
  - Short version: Don’t turn in other people’s work, or allow others to turn in your work as their own
  - Default sanction for any academic misconduct is **FAILING GRADE IN THE COURSE**
Transistor History

1958: First integrated circuit
- Flip-flop using two transistors
- Built by Jack Kilby at Texas Instruments

2008: Intel Core2 Duo – 291,000,000 transistors
- 53% compound annual growth rate over 50 years
  - No other technology has grown so fast so long
- Driven by miniaturization of transistors
  - Smaller is cheaper, faster, lower in power!
  - Revolutionary effects on society

2012: NVIDIA GK110 (Kepler) ~7,000,000,000 transistors
Back to Intel Core2 Duo...

- Even 291 million is a LOT of transistors
- Where are they used?
  - Mostly for memory!
  - Intel Core2 Duo: 4MB shared L2 cache, 32K Icache 32K Dcache on each core
  - $4 \times 2^{28} + 2(64 \times 2^{28}) = 34,603,008$ bits
  - Around 6 transistors per bit of memory
  - $\sim 35,000,000 \text{ bits} \times 6 = \sim 210,000,000$ transistors

Intel Core2 Duo (2008)

- 65nm process, 75W, 144 mm$^2$ die
Historical Comparison

Core2 Duo
- 65nm devices (released in 2008)
- 144mm² die
- 291,000,000 transistors
- over 4MB (32Mbit) of on-chip storage
- 2200MHz

6502 (Apple II, Nintendo NES etc.)
- 600nm devices (6 micron) (released in 1975)
- 22mm² die
- 3510 transistors (nmos only)
- 56 total bits of state
- 1MHz

Transistor Revolution

- Vacuum tubes ruled in first half of 20th century: large, expensive, power-hungry, unreliable
- 1947: first point contact transistor
  - William Shockley, John Bardeen, and Walter Brattain at Bell Labs
  - Read Crystal Fire
    - by Riordan, Hoddeson
First Integrated Circuit

The first working integrated circuit was created by Jack Kilby in 1958.

1.3MHz oscillator on Germanium

First Monolithic IC

Robert Noyce, Flip Flop, on Silicon 1961
Transistor Types

- Bipolar transistors
  - npn or pnp silicon structure
  - Small current into very thin base layer controls large currents between emitter and collector
  - Base currents limit integration density

- Metal Oxide Semiconductor Field Effect Transistors (MOSFET)
  - nMOS and pMOS FETs
  - Voltage applied to insulated gate controls current between source and drain
  - Low power allows very high integration
MOS Integrated Circuits

- 1970’s processes usually had only nMOS transistors
  - Inexpensive, but idle current consumes power

Hand-Design...
Moore's Law

1965: Gordon Moore plotted transistors per chip

- Fit straight line on semilog scale
- Transistor counts have doubled every 26 months
Other Exponentials

- e.g. clock speed, performance, etc...

The Big Picture
Cell Design Tools

- NC_Verilog
- Behavioral Verilog
- Your Library

Spectre
- Cadence Virtuoso Layout
- DRC
- Encounter Library Characterizer
- Abstract

LVS
- Cadence Composer Schematic

NC_Verilog

Cell Design Tools

- NC_Verilog
- Behavioral Verilog
- Your Library

Spectre
- Cadence Virtuoso Layout
- DRC
- Encounter Library Characterizer
- Abstract

LVS
- Cadence Composer Schematic

NC_Verilog

CAD1
Cell Design Tools

Gates are made from Transistors
Cadence Composer Symbol

Cadence Virtuoso Layout
module moore (clk, clr, insig, outsig);
    input clk, clr, insig;
    output outsig;

    // define state encodings as parameters
    parameter [1:0] s0 = 2’b00, s1 = 2’b01, s2 = 2’b10, s3 = 2’b11;

    // define reg vars for state register
    // and next_state logic
    reg [1:0] state, next_state;

    // define state register (with
    // synchronous active-high clear)
    always @(posedge clk)
        begin
            if (clr) state = s0;
                else state = next_state;
        end

    // assign outsig as continuous assign
    assign outsig = ((state == s1) || (state == s3));

    // define combinational logic for
    // next_state
    always @(insig or state)
        begin
            case (state)
                s0: if (insig) next_state = s1;
                    else next_state = s0;
                s1: if (insig) next_state = s2;
                    else next_state = s1;
                s2: if (insig) next_state = s3;
                    else next_state = s2;
                s3: if (insig) next_state = s1;
                    else next_state = s0;
            endcase
        end

endmodule
HDL Synthesis

- Convert the Behavioral HDL into a set of logic gates
  - This process is called “synthesis”
  - Synthesis will target the cells (gates) in your library
  - We’ll use Design Compiler from Synopsys

- Output from synthesis is a Structural HDL description

```
module moore ( clk, clr, insig, outsig );
  input clk, clr, insig;
  output outsig;

  wire   n4, n5, n6, n7, n8;
  wire   [1:1] state;
  wire   [1:0] next_state;

  DFF_QB state_reg_0 ( .D(next_state[0]), .G(clk), .CLR(clr), .Q(outsig), .QB(n4) );
  DFF state_reg_1_ ( .D(next_state[1]), .G(clk), .CLR(clr), .Q(state[1]) );
  MUX2_INV U7 ( .A(n6), .B(n7), .S(n5), .Y(next_state[1]) );
  INVX1 U8 ( .A(state[1]), .Y(n5) );
  NAND2 U9 ( .A(outsig), .B(insig), .Y(n7) );
  INVX1 U10 ( .A(n4), .Y(n6) );
  XOR2 U11 ( .A(insig), .B(n8), .Y(next_state[0]) );
  NOR2 U12 ( .A(state[1]), .B(n4), .Y(n8) );
endmodule
```
Assemble Gates into Circuit

- Process is called Place and Route
  - We’ll use the Encounter Digital Implementation (EDI) system from Cadence

Standard Cell Procedure
Place Cells and Fillers

Connect Rows to Power
autoRouted View

autoRouted Layout View
Corners

Routing
Slightly Larger Example

Assemble into Chip
Example from 2001

16-bit Processor, approx 27,000 transistors

Same Chip (no M2, M3)

1.5mm x 3.0mm, 72 I/O pads
Another 2001 Project

3.0mm x 3.0mm
84 I/O Pads

Standard Cell Portion
Example from 2003

Basketball Scoreboard Display

Example from 2005

Bomb game
With VGA output
Example from 2013

- Game of Life processor
- Michael Ballantyne
- Meng Jia
- VGA controller
- “Life” automata
- Memory interface

<table>
<thead>
<tr>
<th>Chip</th>
<th>Pin Function</th>
<th># of Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>address</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>data</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>write enable</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>20</td>
</tr>
<tr>
<td>ROM</td>
<td>address</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>data</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>18</td>
</tr>
<tr>
<td>VGA</td>
<td>total</td>
<td>3</td>
</tr>
<tr>
<td>Main Chip</td>
<td>reset</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>clock</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>43</td>
</tr>
</tbody>
</table>
Fabricate and Test Chip

- We can fabricate the chips through MOSIS
  - Educational program sponsored by MOSIS’ commercial activities
  - Chips are fabricated, packaged, and shipped back to us

- Then we get to test them to see what they do, or don’t do…
  - CS/ECE 6712 in spring semester
  - Test machine is Tektronix LV500 (or maybe not…)

What is “Design?”

- What is a good design process?
- What makes a good design?
- What are the skills required?

- This is part of what makes this fun!
  - We’ll discuss throughout the class
First Taste of Digital VLSI

- This class is “soup to nuts”
  - Entire process from start to finish
    - Design and characterize a cell library
    - Use that cell library to build a chip
  - But, there’s lots more to learn!
    - More modern issues
    - Industry best practice
  - 6770 Advanced VLSI takes over where 5710/6710 leaves off!

VLSI at Utah

- VLSI is a means to an end, not an end in itself…
  - How to build ultra small and efficient systems
  - Learn how, why, when, and where a VLSI implementation makes sense
- Research at Utah has ties to VLSI
  - SoC: Brunvand, Balasubramonian
  - ECE: Stevens, Kalla, Walling, Walker, Gaillardon
VLSI in Industry

- Varied skills needed
  - Architecture, CAD, design, validation
  - Software skills as critical as circuit skills!

- If you’re a CS student, don’t be intimidated!
  - Varied employment opportunities
  - Large companies to small startups

- Grad degrees highly valued here
  - Especially for design side employment

IC Technology Curve

[Graph showing the IC Technology Curve with feature sizes and years from 1970 to 2015]
Our Technology

- We’ll use the ON Semi 0.5u (500nm) 3-level-metal CMOS process (very very old stuff!!)
  - We have technology files that define the process
  - MOSIS Scalable CMOS Rev. 8 (SCMOS)

- Tech files from NCSU CDK
  - NCSU toolkit is designed for custom VLSI layout
  - Design Rule Check (DRC) rules
  - Layout vs. Schematic (LVS) rules

Possible Technology

- Global Foundries 180nm process
  - Formerly IBM 180nm 7RF process

- Only very old, not very very old!
  - But, quite a bit more complex!

- We’ll see if the TAs and I have the stamina to put together a cohesive design kit that we trust…
Course Overview

- Start with transistors as switches
  - Boolean gates
- Study logical & electrical transistor behavior
- Mask layout for the gates
  - Design and characterize a set of gates (library)
- Use that library to build a whole-chip project
- Fabricate the chip and test in Spring 2016
  - This is optional
  - Rewarded with a fun 1-hour testing class (6712)

Project Details

- Standard Cell Library
  - Each group will design a small, but useful, standard cell library
- Use HDL synthesis with this library as a target
  - Use Cadence EDI for place and route
- Custom Datapath
  - Use ICC router to connect HDL-Synthesized control to custom-designed datapath
- It will be VERY helpful to have a mix of knowledge on your team
Tool Details

- Multiple design views for your library cells:
  - Start with Schematic, Verilog, Symbol, Layout views of each cell
  - Complete design in Composer schematics, simulated with NC_Verilog
  - Complete design layout in Virtuoso, detailed simulation using Spectre
  - Validate they are the same with Diva LVS/DRC
  - Characterize with Encounter Library Characterizer (ELC)

- Synthesized controller using Synopsys Design Compiler or Cadence RTL Compiler

- Place-Route with Encounter Digital Implementation System (EDI)

- Final assembly back in Virtuoso and CCAR

Timetable

- The project will be a race to the finish!
  - There is no slack in this schedule!!!
Timetable

- The project will be a race to the finish!
  - There is no slack in this schedule!!!
- VLSI design always takes longer than you think
  - Even if you take that rule into account!

- After you have 90% finished, there’s only 90% left…
  - All team members will have to contribute!
  - Team peer evaluations twice a semester
Summary

• Learn about VLSI design
  • Develop tool & layout skills independently
  • Form a team – develop a cell library
  • Decide on a project architecture
  • Then use your team’s library to make a chip

• Verilog / synthesis / place & route / chip-fab