Intro to Digital Design

Combinational Logic

Instructor: Justin Hsia

Teaching Assistants:
Emilio Alcantara    Eujean Lee
Naoto Uemura       Pedro Amarante
Wen Li
Introducing Your Course Staff

❖ Your Instructor: just call me Justin
  ▪ CSE Associate Teaching Professor
  ▪ From California (UC Berkeley and the Bay Area)
  ▪ Raising a toddler takes up energy and dictates my schedule

❖ TAs:
  ▪ Available in labs, office hours, and on Ed discussion
  ▪ An invaluable source of information and help

❖ Get to know us – we are here to help you succeed!
Course Motivation

- Electronics an increasing part of our lives
  - Computers & phones
  - Vehicles (cars, planes)
  - Robots
  - Portable & household electronics

- An introduction to digital logic design
  - **Lecture:** How to think about hardware, basic higher-level circuit design techniques – preparation for EE/CSE469
  - **Lab:** Hands-on FPGA programming using Verilog – preparation for EE/CSE371
Digital vs. Analog

Digital:
Discrete set of possible values

Binary (2 values):
On, 3.3 V, high, TRUE, "1"
Off, 0 V, low, FALSE, "0"

Analog:
Values vary over a continuous range
Digital vs. Analog Systems

- Digital systems are more reliable and less error-prone
  - Slight errors can cascade in Analog system
  - Digital systems reject a significant amount of error; easy to cascade

- Computers use digital circuits internally
  - CPU, memory, I/O

- Interface circuits with “real world” often analog
  - Sensors & actuators

This course is about logic design, not system design (processor architecture), and not circuit design (transistor level)
Digital Design: What’s It All About?

- Come up with an implementation using a set of primitives given a functional description and constraints

- Digital design is in some ways more art than a science
  - The creative spirit is in combining primitive elements and other components in new ways to achieve a desired function

- However, unlike art, we have objective measures of a design (i.e., constraints):
  - Performance
  - Power
  - Cost
Digital Design: What’s It All About?

❖ How do we learn how to do this?
  ▪ Learn about the primitives and how to use them
  ▪ Learn about design representations
  ▪ Learn formal methods and tools to manipulate representations
  ▪ Look at design examples
  ▪ Use trial and error – CAD tools and prototyping (practice!)
Lecture Outline

❖ Course Logistics
❖ Combinational Logic Review
❖ Combinational Logic in the Lab
Bookmarks

❖ Website: [https://courses.cs.washington.edu/courses/cse369/24sp/](https://courses.cs.washington.edu/courses/cse369/24sp/)
  - Schedule (lecture slides, lab specs), weekly calendar, other useful documents

❖ Ed Discussion: [https://edstem.org/us/courses/56771/](https://edstem.org/us/courses/56771/)
  - Announcements made here
  - Ask and answer questions – staff will monitor and contribute

❖ Gradescope: [https://www.gradescope.com/courses/746339/](https://www.gradescope.com/courses/746339/)
  - Lab submissions, Quiz grades, regrade requests

❖ Canvas: [https://canvas.uw.edu/courses/1718545/](https://canvas.uw.edu/courses/1718545/)
  - Grade book, Zoom links, lecture recordings
Grading

❖ Labs (66%)
  ▪ 6 regular labs – 1 week each
    • Labs 3-4: 60 points each, Labs 1&2, 5-7: 100 points each
  ▪ 1 “final project” – 2 weeks
    • Lab 8 Check-In: 10 points, Lab 8: 150 points

❖ 3 Quizzes (no final exam)
  ▪ Quiz 1 (10%): 20 min in class on April 23
  ▪ Quiz 2 (10%): 30 min in class on May 14
  ▪ Quiz 3 (14%): 60 min in class on May 28

❖ This class uses a straight scale (≥ 95% → 4.0)
  ▪ Extra credit points count the same as regular points
Labs

- Lab Hours: Wed & Thu 2:30-5:20 pm (CSE 003)
- Each student will get a lab kit for the quarter
  - Lab kit picked up from CSE 003 during labs/OHs this week
  - Install software on laptop (Windows or VM)
- Labs are combination of report + demo
  - Submit via Gradescope Wednesdays before 2:30 pm
  - 10-minute demos done in lab sections (sign-up process)
- Late penalties:
  - No lab report can be submitted more than two days late
  - 5 late day tokens to prevent penalties, 10%/day after that
  - No penalties on lab demos, but must be done by EOD Friday
Collaboration Policy

❖ Labs and project are to be completed *individually*
  ▪ Goal is to give every student the hands-on experience
  ▪ Violation of these rules is grounds for failing the class

❖ **OK:**
  ▪ Discussing lectures and/or readings, studying together
  ▪ *High-level* discussion of general approaches
  ▪ Help with debugging, tools peculiarities, etc.

❖ **Not OK:**
  ▪ Developing a lab together
  ▪ Giving away solutions or having someone else do your lab for you
Course Workload

- The workload (3 credits) ramps up significantly towards the end of the quarter:

![CSE 369 Lab Hours (2018) Chart]

- **Regular Labs**
- **Project (Lab 8)**
Lecture Outline

- Course Logistics
- **Combinational Logic Review**
- Combinational Logic in the Lab
Combinational vs. Sequential Logic

❖ Combinational Logic (CL)

- Network of logic gates without feedback
- Outputs are functions only of inputs

❖ Sequential Logic (SL)

- The presence of feedback introduces the notion of “state”
- Circuits that can “remember” or store information
Representations of Combinational Logic

- Text Description
- Circuit Description
  - Transistors
  - Logic Gates
- Truth Table
- Boolean Expression

*All are equivalent!*
Example: Simple Car Electronics

- Door Ajar (DriverDoorOpen, PassengerDoorOpen)
  \[ DA = DDO + PDO \]

- High Beam Indicator (LightsOn, HighBeamOn)
  \[ HBI = LO \cdot HBO \]

- Seat Belt Light (DriverBeltIn, PassengerBeltIn, Passenger)
  \[ SBL = DBI + (P \cdot PBI) \]
Truth Tables

- Table that relates the inputs to a combinational logic (CL) circuit to its output
  - Output *only* depends on current inputs
  - Use abstraction of 0/1 instead of high/low voltage
  - Shows output for *every* possible combination of inputs (“black box” approach)

- How big is the table?
  - 0 or 1 for each of *N* inputs
  - Each output is a separate function of inputs, so don’t need to add rows for additional outputs
CL General Form

If we have N inputs, how many distinct functions $F$ do we have?

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F(0,0,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F(0,0,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>F(0,0,1,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F(0,0,1,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(0,1,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(0,1,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(0,1,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(0,1,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F(1,0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F(1,0,0,1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>F(1,0,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F(1,0,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(1,1,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(1,1,0,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(1,1,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(1,1,1,1)</td>
</tr>
</tbody>
</table>
Logic Gate Names and Symbols

❖ NOT

$$\begin{array}{c|c}
A & \text{Out} \\
0 & 1 \\
1 & 0 \\
\end{array}$$

❖ AND

$$\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}$$

❖ OR

$$\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}$$

❖ XOR

$$\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}$$

❖ NAND

$$\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}$$

❖ NOR

$$\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}$$

❖ XNOR

$$\begin{array}{c|c|c}
A & B & \text{Out} \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}$$

Circle indicates NOT
More Complicated Truth Tables

3-Input Majority

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

How many rows?

1-bit Adder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Carry</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

2 separate functions

A⋅B  A⊕B
Boolean Algebra

- Represent inputs and outputs as variables
  - Each variable can only take on the value 0 or 1

- Overbar is NOT: “logical complement”
  - If \( A \) is 0, then \( \overline{A} \) is 1 and vice-versa

- Plus (+) is 2-input OR: “logical sum”

- Product (\( \cdot \)) is 2-input AND: “logical product”

- All other gates and logical expressions can be built from combinations of these
  - \( e.g., \) \( A \) XOR \( B = A \oplus B = \overline{A}B + \overline{B}A \)
Truth Table to Boolean Expression

- **Read off of table**
  - For 1, write variable name
  - For 0, write complement of variable

- **Sum of Products (SoP)**
  - Take rows with 1’s in output column, sum products of inputs
  - \[ C = \overline{AB} + \overline{BA} \]

- **Product of Sums (PoS)**
  - Take rows with 0’s in output column, product the sum of the complements of the inputs
  - \[ C = (A + B) \cdot (\overline{A} + \overline{B}) \]

We can show that these are equivalent!
Basic Boolean Identities

- $X + 0 = X$
- $X + 1 = 1$
- $X + X = X$
- $X + \overline{X} = 1$
- $\overline{X} = X$

- $X \cdot 1 = X$
- $X \cdot 0 = 0$
- $X \cdot X = X$
- $X \cdot \overline{X} = 0$
Basic Boolean Algebra Laws

❖ **Commutative Law:**
\[ X + Y = Y + X \quad X \cdot Y = Y \cdot X \]

❖ **Associative Law:**
\[ X + (Y + Z) = (X + Y) + Z \quad X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z \]

❖ **Distributive Law:**
\[ X \cdot (Y + Z) = X \cdot Y + X \cdot Z \quad X + YZ = (X + Y) \cdot (X + Z) \]
Advanced Laws (Absorption)

- $X + XY = X$
- $XY + X\overline{Y} = X$
- $X + \overline{X}Y = X + Y$
- $X(X + Y) = X$
- $(X + Y)(X + \overline{Y}) = X$
- $X(\overline{X} + Y) = XY$
Practice Problem

- Boolean Function: \( F = \overline{X}YZ + XZ \)

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>( Z )</th>
<th>( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Technology

Break
Lecture Outline

❖ Course Logistics
❖ Combinational Logic Review
❖ Combinational Logic in the Lab
Why Is This Useful?

❖ Logic minimization: reduce complexity at gate level
  ▪ Allows us to build smaller and faster hardware
  ▪ Care about both # of gates, # of literals (gate inputs), # of gate levels, and types of logic gates
Why Is This Useful?

❖ Logic minimization: reduce complexity at gate level
  ▪ Allows us to build smaller and faster hardware
  ▪ Care about both # of gates, # of literals (gate inputs), # of gate levels, and types of logic gates

❖ Faster hardware?
  ▪ Fewer inputs implies faster gates in some technologies
  ▪ Fan-ins (# of gate inputs) are limited in some technologies
  ▪ Fewer levels of gates implies reduced signal propagation delays
  ▪ # of gates (or gate packages) influences manufacturing costs
  ▪ Simpler Boolean expressions → smaller transistor networks → smaller circuit delays → faster hardware
Are Logic Gates Created Equal?

❖ No!

<table>
<thead>
<tr>
<th>2-Input Gate Type</th>
<th># of CMOS transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT</td>
<td>2</td>
</tr>
<tr>
<td>AND</td>
<td>6</td>
</tr>
<tr>
<td>OR</td>
<td>6</td>
</tr>
<tr>
<td>NAND</td>
<td>4</td>
</tr>
<tr>
<td>NOR</td>
<td>4</td>
</tr>
<tr>
<td>XOR</td>
<td>8</td>
</tr>
<tr>
<td>XNOR</td>
<td>8</td>
</tr>
</tbody>
</table>

❖ Can recreate all other gates using only NAND or only NOR gates
  ▪ Called “universal” gates
  ▪ e.g., A NAND A = \(\overline{A}\), B NOR B = \(\overline{B}\)
  ▪ DeMorgan’s Law helps us here!
DeMorgan’s Law

- \( \overline{X + Y} = \overline{X} \cdot \overline{Y} \)
- \( \overline{X \cdot Y} = \overline{X} + \overline{Y} \)

- In Boolean Algebra, converts between AND-OR and OR-AND expressions
  - \( Z = \overline{ABC} + \overline{ABC} + \overline{ABC} \)
  - \( \overline{Z} = (A + B + \overline{C}) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \)
- At gate level, can convert from AND/OR to NAND/NOR gates
  - “Flip” all input/output bubbles and “switch” gate

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>( \overline{X} )</th>
<th>( \overline{Y} )</th>
<th>( X + Y )</th>
<th>( \overline{X \cdot Y} )</th>
<th>( X \cdot Y )</th>
<th>( \overline{X + Y} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[\text{NOR}\]

\[\text{NAND}\]
DeMorgan’s Law Practice Problem

❖ Simplify the following diagram:

\[ X = A + B + A \overline{B} + \overline{A} \overline{B} + \overline{A} C + D \]

❖ Then implement with only NAND gates:
Transistor-Transistor Logic (TTL) Packages

- Diagrams like these and other useful/helpful information can be found on part data sheets
  - It’s really useful to learn how to read these

Pin numbering starts at 1, counter-clockwise from dot
Mapping truth tables to logic gates

- Given a truth table:
  1) Write the Boolean expression
  2) Minimize the Boolean expression
  3) Draw as gates
  4) Map to available gates
  5) Determine # of packages and their connections

![Diagram showing mapping from truth table to logic gates]

7 nets (wires) in this design
Breadboarding circuits
Summary

❖ Digital systems are constructed from Combinational and Sequential Logic
❖ Logic minimization to create smaller and faster hardware
❖ Gates come in TTL packages that require careful wiring