

Intro to Digital Design

Combinational Logic

Instructor: Justin Hsia

Teaching Assistants:

Caitlyn Rawlings

Donovan Clay

Emilio Alcantara

Joy Jung

Naoto Uemura

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:



Caitlyn



Donovan



Emilio



Joy



Naoto

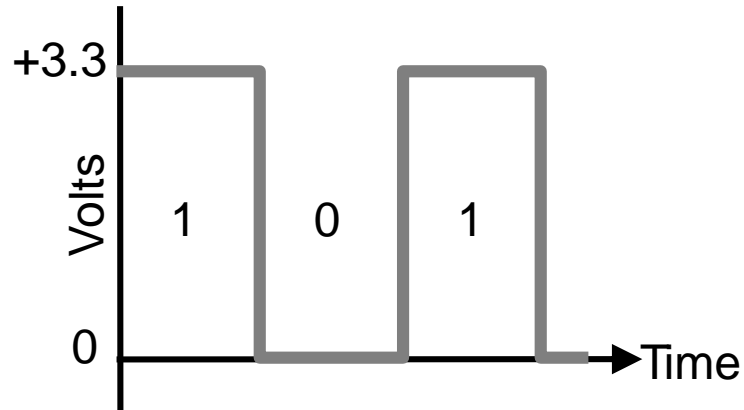
- Available in labs, support (office) hours, and on Ed
 - An invaluable source of information and help
- ❖ Get to know us – we are here to help you succeed!

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



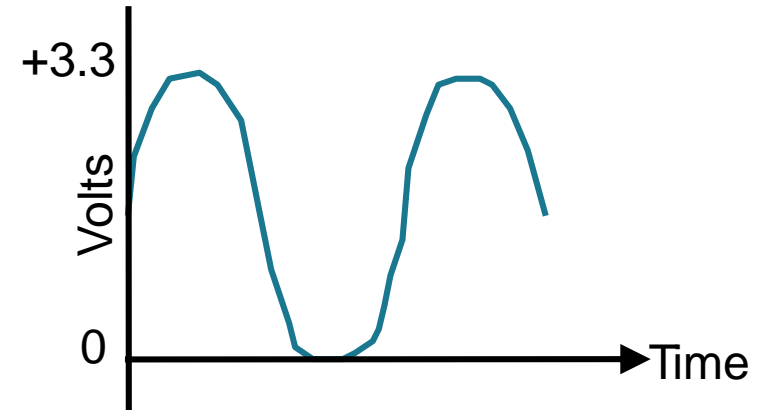
Digital:

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://cs.uw.edu/369/>
 - Schedule (lecture slides, lab specs), weekly calendar, other useful documents
- ❖ Ed Discussion: <https://edstem.org/us/courses/50615/>
 - Announcements made here
 - Ask and answer questions – staff will monitor and contribute
- ❖ Gradescope: <https://www.gradescope.com/courses/680677/>
 - Lab submissions, Quiz grades, regrade requests
- ❖ Canvas: <https://canvas.uw.edu/courses/1695952/>
 - 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 January 30
 - Quiz 2 (10%): 30 min in class on February 20
 - Quiz 3 (14%): 60 min in class on March 5
- ❖ 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
 - 4 late day tokens to prevent penalties, 10%/day after that
 - No penalties on lab demos, but must be done by EOD Friday

Collaboration Policy

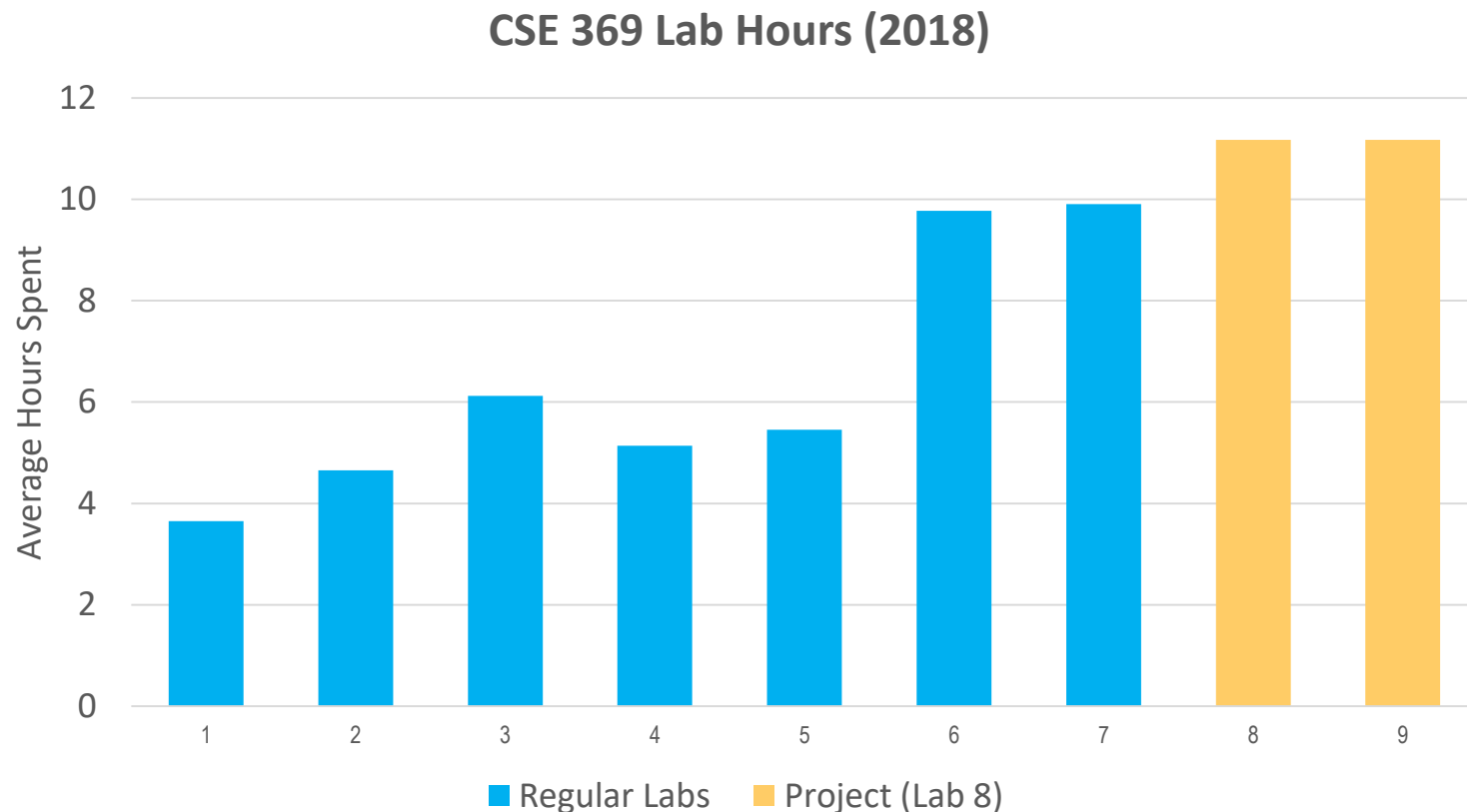
- ❖ Labs and project are to be completed *individually*
 - Goal is to give every student the hands-on experience
 - Violation of these rules is grounds for failing the class

- ❖ **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:

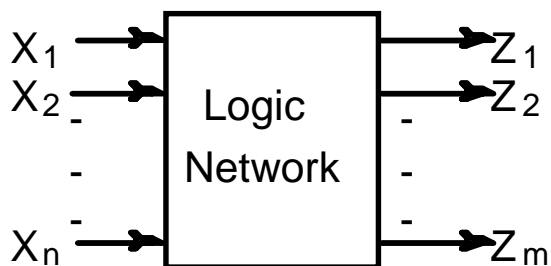


Lecture Outline

- ❖ Course Logistics
- ❖ **Combinational Logic Review**
- ❖ Combinational Logic in the Lab

Combinational vs. Sequential Logic

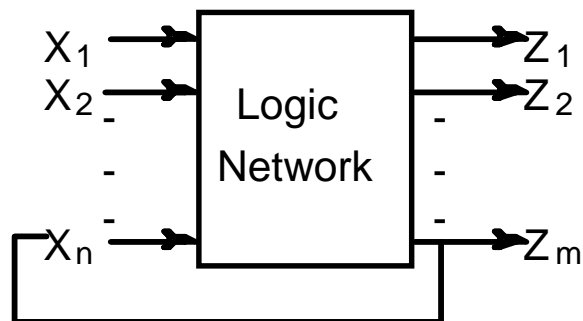
❖ Combinational Logic (CL)



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~~ Not covered in 369
 - Logic Gates
- ❖ Truth Table
- ❖ Boolean Expression

- ❖ *All are equivalent!*

Example: Simple Car Electronics

- ❖ Door Ajar (DriverDoorOpen, PassengerDoorOpen)

- $DA = DDO + PDO$



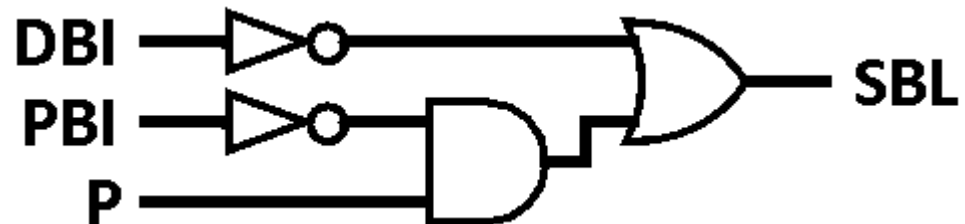
- ❖ High Beam Indicator (LightsOn, HighBeamOn)

- $HBI = LO \cdot HBO$



- ❖ Seat Belt Light (DriverBeltIn, PassengerBeltIn, Passenger)

- $SBL = \overline{DBI} + (P \cdot \overline{PBI})$

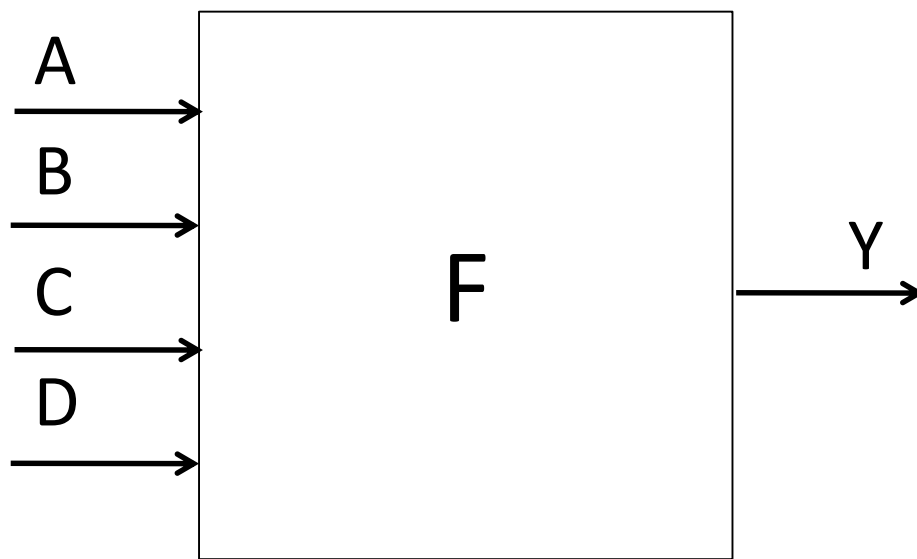


Truth Tables

- ❖ 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, so 2^N rows
 - Each output is a separate function of inputs, so don't need to add rows for additional outputs

CL General Form



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

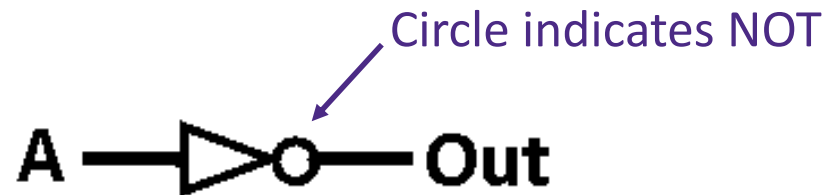
Function maps each row to 0 or 1,
so $2^{(2^N)}$ possible functions

| a | b | c | d | y |
|---|---|---|---|--------------|
| 0 | 0 | 0 | 0 | $F(0,0,0,0)$ |
| 0 | 0 | 0 | 1 | $F(0,0,0,1)$ |
| 0 | 0 | 1 | 0 | $F(0,0,1,0)$ |
| 0 | 0 | 1 | 1 | $F(0,0,1,1)$ |
| 0 | 1 | 0 | 0 | $F(0,1,0,0)$ |
| 0 | 1 | 0 | 1 | $F(0,1,0,1)$ |
| 0 | 1 | 1 | 0 | $F(0,1,1,0)$ |
| 1 | 1 | 1 | 1 | $F(0,1,1,1)$ |
| 1 | 0 | 0 | 0 | $F(1,0,0,0)$ |
| 1 | 0 | 0 | 1 | $F(1,0,0,1)$ |
| 1 | 0 | 1 | 0 | $F(1,0,1,0)$ |
| 1 | 0 | 1 | 1 | $F(1,0,1,1)$ |
| 1 | 1 | 0 | 0 | $F(1,1,0,0)$ |
| 1 | 1 | 0 | 1 | $F(1,1,0,1)$ |
| 1 | 1 | 1 | 0 | $F(1,1,1,0)$ |
| 1 | 1 | 1 | 1 | $F(1,1,1,1)$ |

Logic Gates (1/2)

- ❖ Special names and symbols:

NOT



| A | Out |
|---|-----|
| 0 | 1 |
| 1 | 0 |

AND



| A | B | Out |
|---|---|-----|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR



| A | B | Out |
|---|---|-----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Logic Gates (2/2)

- ❖ Special names and symbols:

NAND



| A | B | Out |
|---|---|-----|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR



| A | B | Out |
|---|---|-----|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

XOR



| A | B | Out |
|---|---|-----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

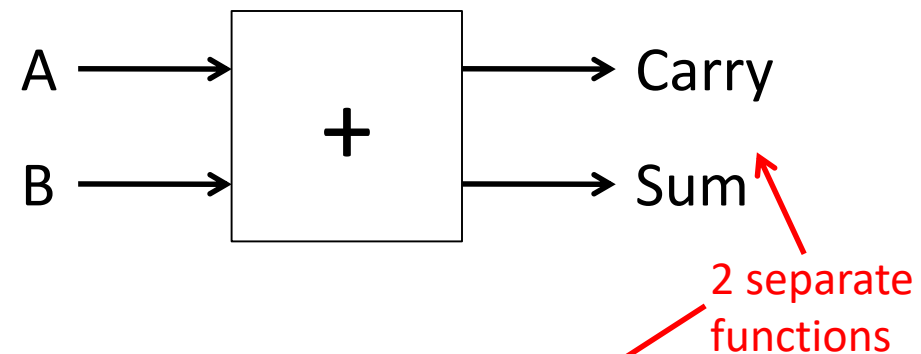
More Complicated Truth Tables

3-Input Majority

How many rows?

| A | B | C | Out |
|---|---|---|-----|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1-bit Adder



| A | B | Carry | Sum |
|---|---|-------|-----|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$A \cdot B$ $A \oplus B$

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 \bar{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.*, $A \text{ XOR } B = A \oplus B = \bar{A}B + \bar{B}A$

Truth Table to Boolean Expression

❖ Read off of table

- For 1, write variable name
- For 0, write complement of variable

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

❖ *Sum of Products (SoP)*

- Take rows with 1's in output column, sum products of inputs
- $C = \bar{A}B + \bar{B}A$

We can show that these are equivalent!

❖ *Product of Sums (PoS)*

- Take rows with 0's in output column, product the sum of the complements of the inputs
- $C = (A + B) \cdot (\bar{A} + \bar{B})$

Basic Boolean Identities

$$\diamond X + 0 = X$$

$$\diamond X \cdot 1 = X$$

$$\diamond X + 1 = 1$$

$$\diamond X \cdot 0 = 0$$

$$\diamond X + X = X$$

$$\diamond X \cdot X = X$$

$$\diamond X + \bar{X} = 1$$

$$\diamond X \cdot \bar{X} = 0$$

$$\diamond \bar{\bar{X}} = X$$

Basic Boolean Algebra Laws

❖ Commutative Law:

$$X + Y = Y + X$$

$$X \cdot Y = Y \cdot X$$

❖ Associative Law:

$$X + (Y + Z) = (X + Y) + Z$$

$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$$

❖ Distributive Law:

$$X \cdot (Y + Z) = X \cdot Y + X \cdot Z$$

$$X + YZ = (X + Y) \cdot (X + Z)$$

Advanced Laws (Absorption)

$$\diamond X + XY = X$$

$$\diamond XY + X\bar{Y} = X$$

$$\diamond X + \bar{X}Y = X + Y$$

$$\diamond X(X + Y) = X$$

$$\diamond (X + Y)(X + \bar{Y}) = X$$

$$\diamond X(\bar{X} + Y) = XY$$

Practice Problem

❖ Boolean Function: $F = \bar{X}YZ + XZ$

Truth Table:

| X | Y | Z | F |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

Simplification:

$$= \bar{X}YZ + X\bar{Y}Z + XYZ$$

$$= \bar{X}YZ + XZ$$

$$= (\bar{X}Y + X)Z$$

$$= (X + Y)Z$$

$$= XZ + YZ$$

Which of these
is “simpler”?

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!

| 2-Input Gate Type | # of CMOS transistors |
|-------------------|-----------------------|
| NOT | 2 |
| AND | 6 |
| OR | 6 |
| NAND | 4 |
| NOR | 4 |
| XOR | 8 |
| XNOR | 8 |

❖ Can recreate all other gates using only NAND or only NOR gates

- Called “universal” gates
- *e.g.*, $A \text{ NAND } A = \bar{A}$, $B \text{ NOR } B = \bar{B}$
- DeMorgan’s Law helps us here!

DeMorgan's Law

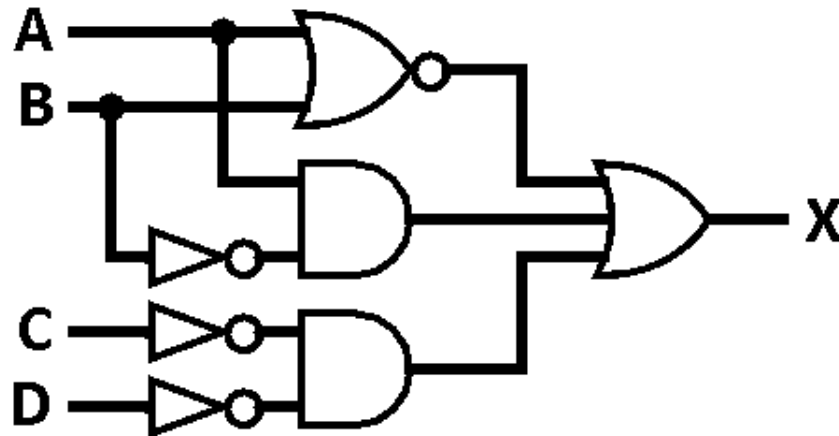
| | X | Y | \bar{X} | \bar{Y} | NOR $\overline{X + Y}$ | $\bar{X} \cdot \bar{Y}$ | NAND $\overline{X \cdot Y}$ | $\bar{X} + \bar{Y}$ |
|--|---|---|-----------|-----------|---------------------------|-------------------------|--------------------------------|---------------------|
| ❖ $\overline{X + Y} = \bar{X} \cdot \bar{Y}$ | 0 | 0 | 1 | 1 | 1 | | 1 | |
| | 0 | 1 | 1 | 0 | 0 | | 1 | |
| ❖ $\overline{X \cdot Y} = \bar{X} + \bar{Y}$ | 1 | 0 | 0 | 1 | 0 | | 1 | |
| | 1 | 1 | 0 | 0 | 0 | | 0 | |

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



DeMorgan's Law Practice Problem

❖ Simplify the following diagram:



$$X = \overline{A + B} + A\overline{B} + \overline{C}\overline{D}$$

$$X = \overline{A}\overline{B} + A\overline{B} + \overline{C}\overline{D}$$

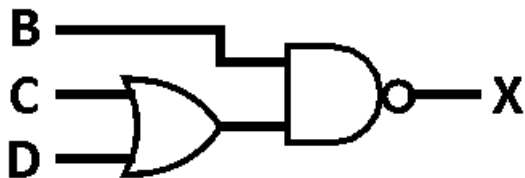
$$X = \overline{B} + \overline{C}\overline{D}$$

$$X = \overline{B} + \overline{C + D}$$

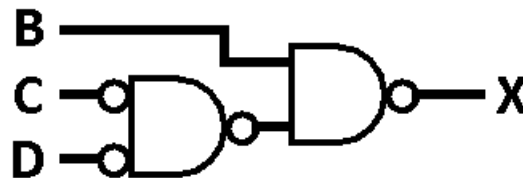
$$X = \overline{B(C + D)}$$

❖ Then implement with only NAND gates:

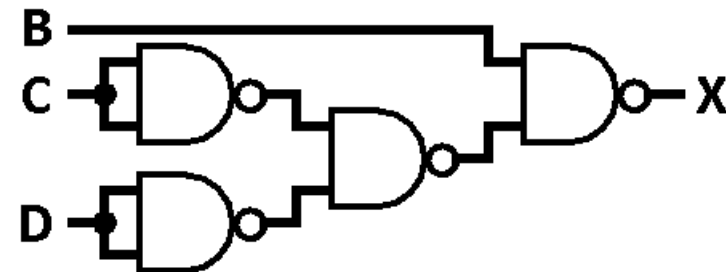
1)



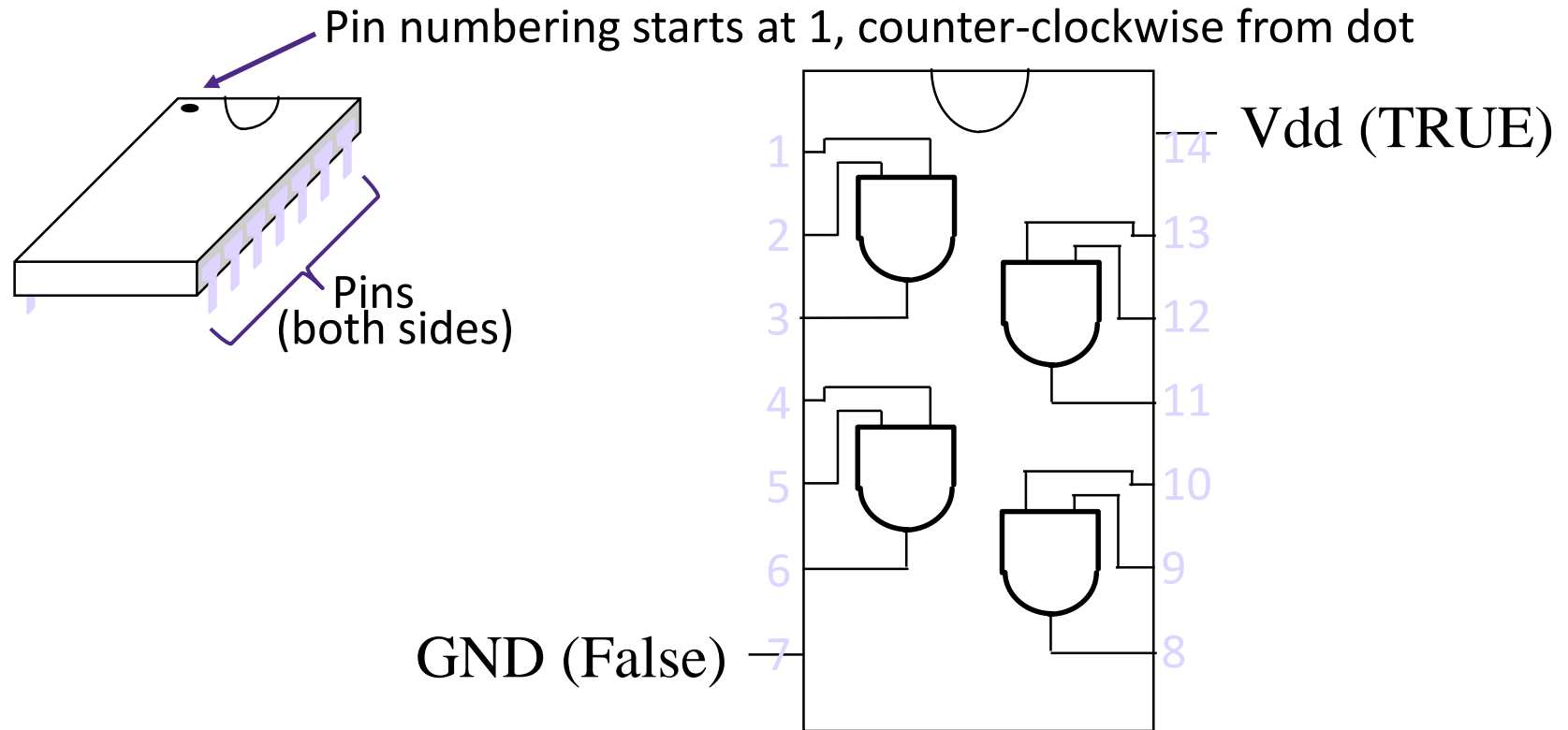
2)



3)



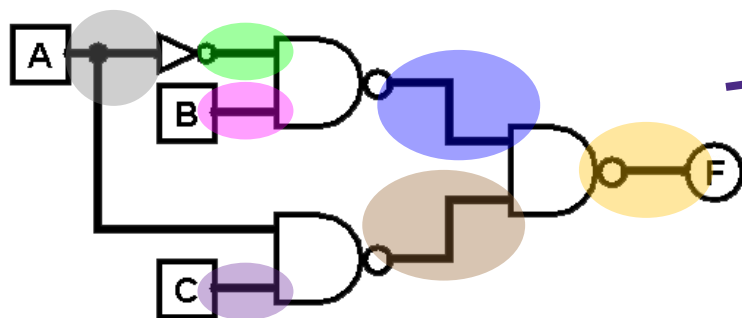
Transistor-Transistor Logic (TTL) Packages



- ❖ 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

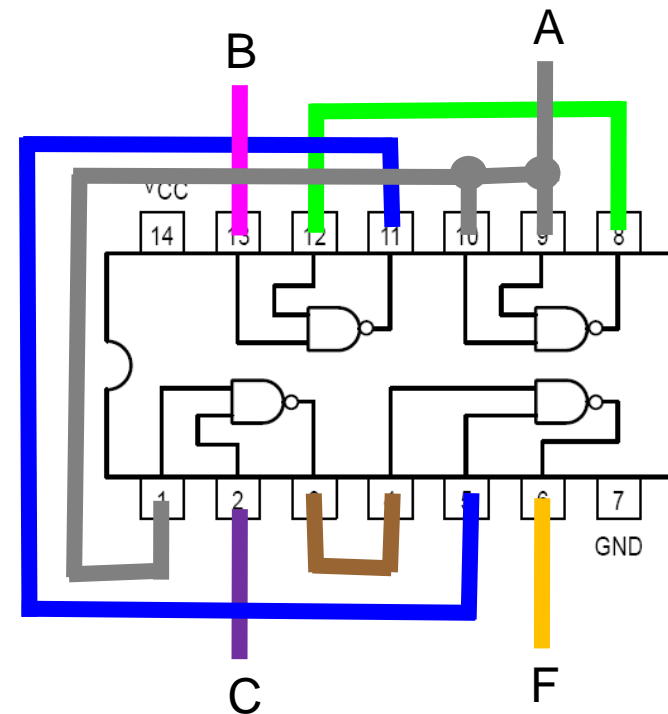
Mapping truth tables to logic gates

- ❖ 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

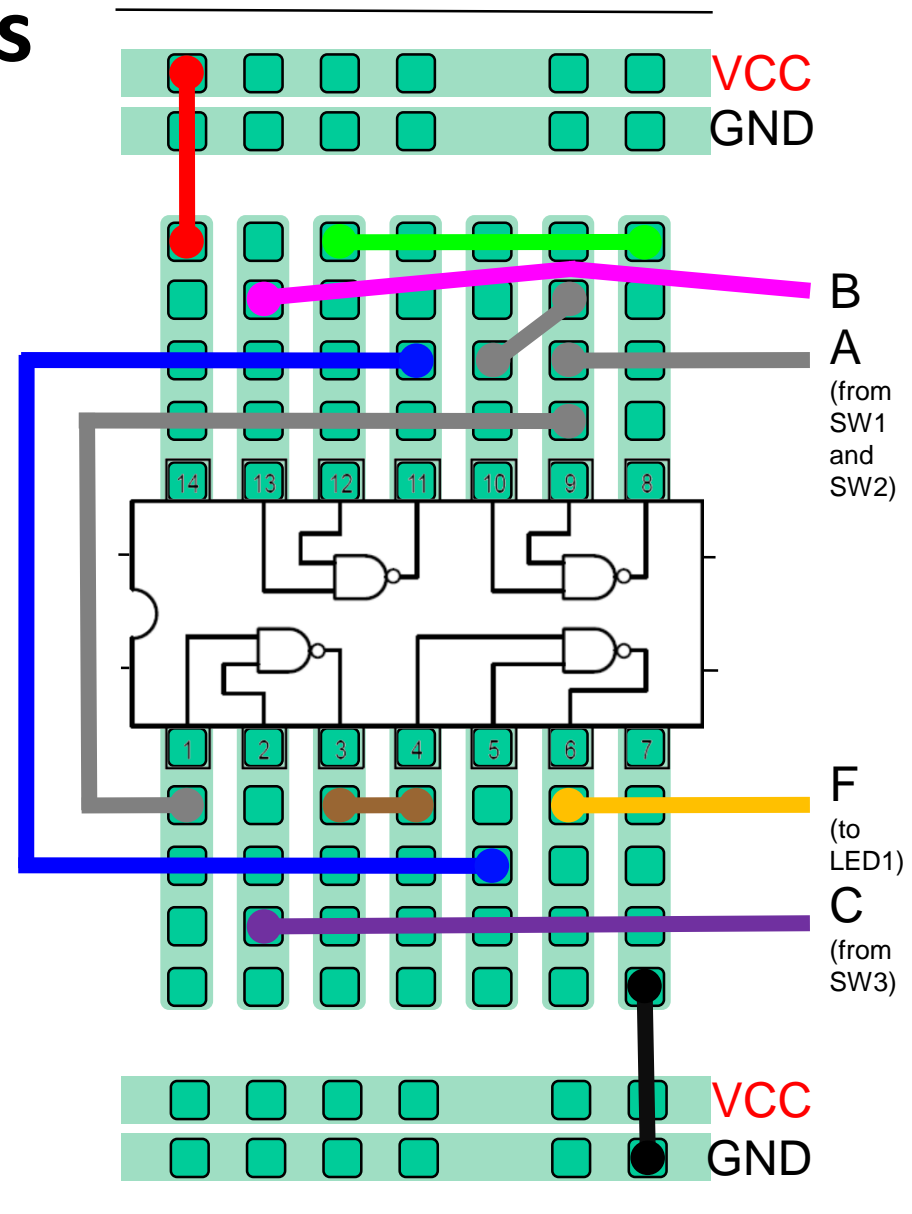
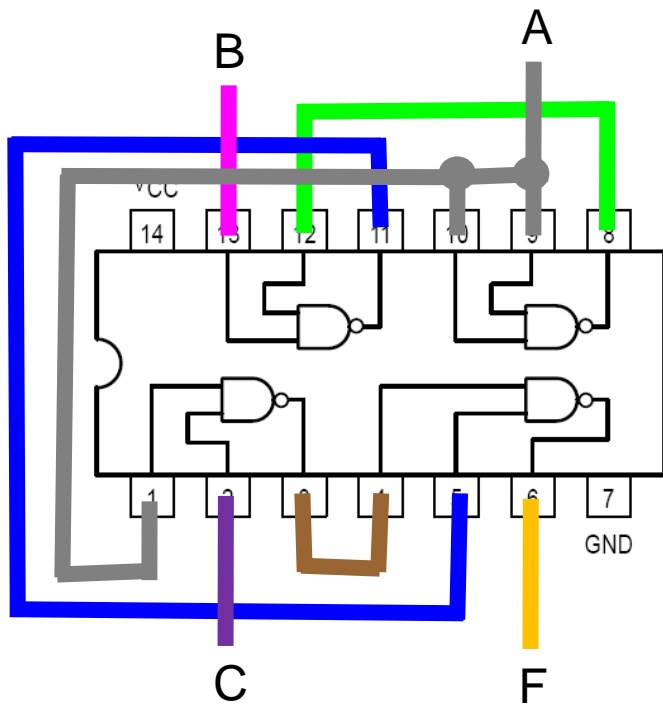


7 nets (wires) in this design

(4) →



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

