Intro to Digital Design
Combinational Logic

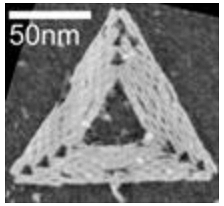
Instructor: Chris Thachuk

Teaching Assistants:

Eujean Lee	Nandini Talukdar
Stephanie Osorio-Tristan	Wen Li

Introducing Your Course Staff

- Your Instructor: just call me Chris
  - From Canada
  - CSE Assistant Professor
    (research focus: Molecular Programming + DNA computing)
  - I like: research, teaching, hiking, cycling, sci-fi



- TAs:

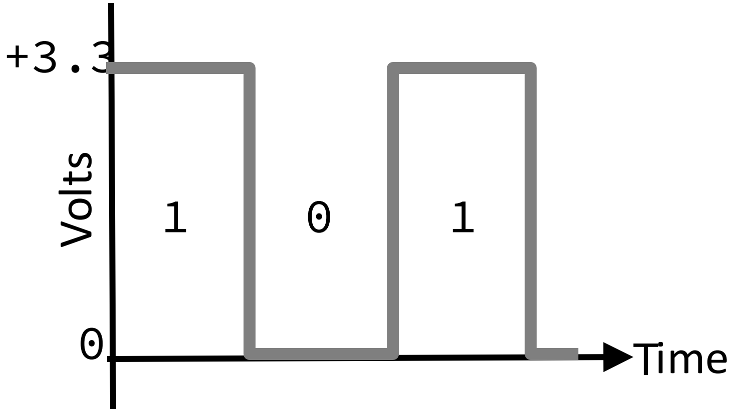
  - Available in labs, office hours, and on Ed discussion
  - An invaluable source of information and help
- Get to know us – we are here to help you succeed!

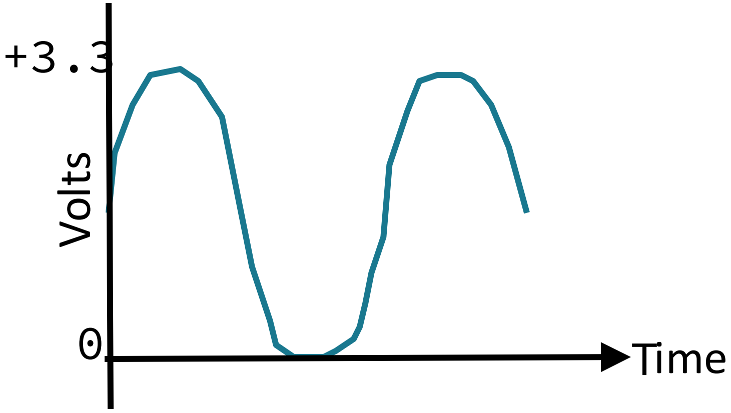
Course Motivation

- 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

apply

Lab

Lecture Outline

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

Bookmarks

- Website: https://courses.cs.washington.edu/courses/cse369/24au/
  - Schedule (lecture slides, lab specs), weekly calendar, other useful documents
- Ed Discussion: https://edstem.org/us/courses/67458/
  - Announcements made here
  - Ask and answer questions – staff will monitor and contribute
- Gradescope: https://www.gradescope.com/courses/885999/
  - Lab submissions, Quiz grades, regrade requests
- Canvas: https://canvas.uw.edu/courses/1746563/
  - 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
  - Quiz 2 (10%): 30 min in class
  - Quiz 3 (14%): 60 min in class
- This class uses a straight scale (≥ 95% → 4.0)
  - Extra credit points count the same as regular points

Labs

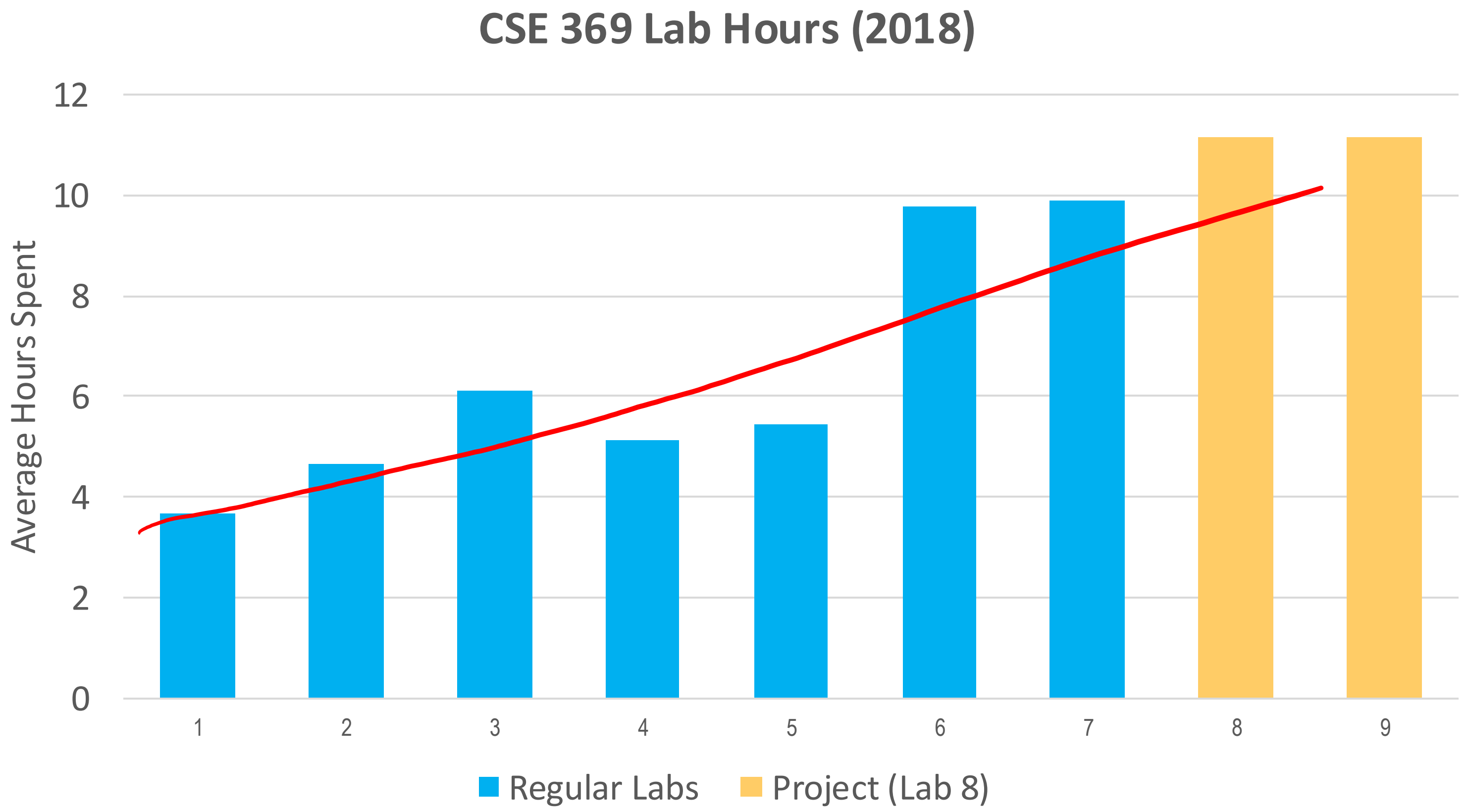
- Lab Hours: Wed & Thu 2:30-5:20 pm (CSE 003)
- Each student will get a lab kit for the quarter
  - Lab kit picked up from CSE 003 during labs/OHs this week
  - Install software on laptop (Windows or VM)
- Labs are combination of report + demo
  - Submit via Gradescope Wednesdays before 2:30 pm
  - 10-minute 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:

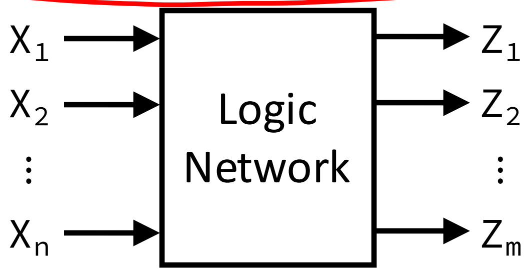


Lecture Outline

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

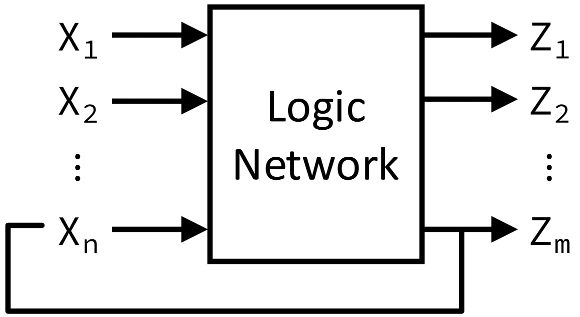
Combinational vs. Sequential Logic

- **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

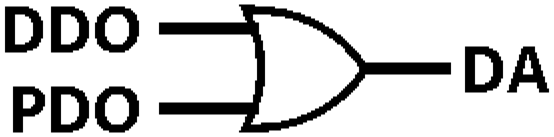
Representations of Combinational Logic

- Text Description
- Circuit Description
  - ~~Transistors~~ Not covered in 369
  - Logic Gates
- Truth Table
- Boolean Expression

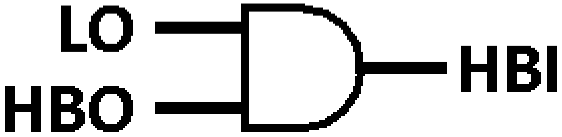
- *All are equivalent!*

Example: Simple Car Electronics

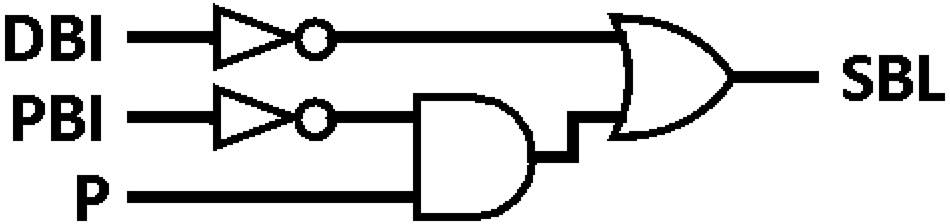
- Door Ajar (DriverDoorOpen, PassengerDoorOpen)
  - $\mathrm{DA} = \mathrm{DDO} + \mathrm{PDO}$



- High Beam Indicator (LightsOn, HighBeamOn)
  - $\mathrm{HBI} = \mathrm{LO} \cdot \mathrm{HBO}$

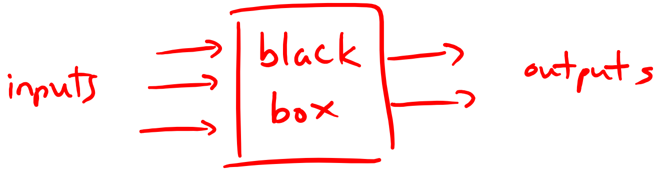


- Seat Belt Light (DriverBeltIn, PassengerBeltIn, Passenger)
  - $\mathrm{SBL} = \overline{\mathrm{DBI}} + (\mathrm{P} \cdot \overline{\mathrm{PBI}})$



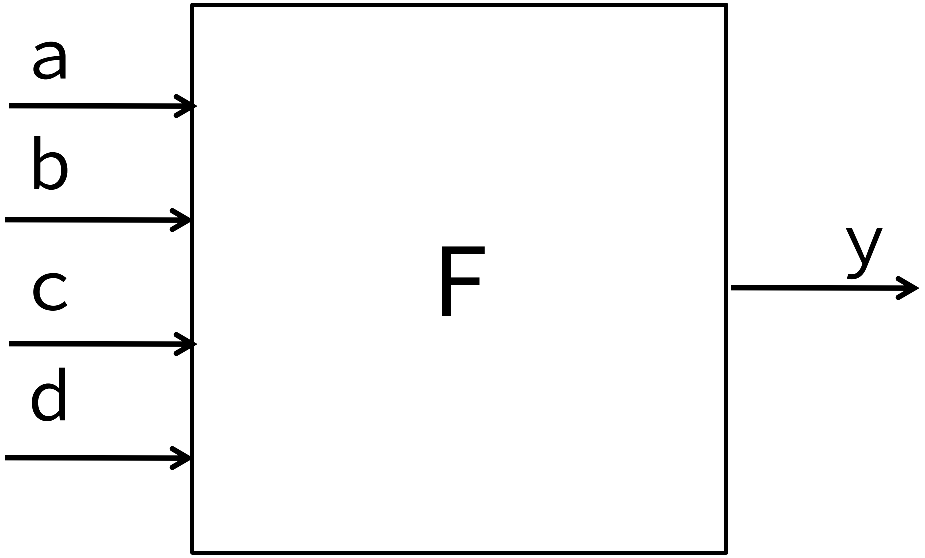
Truth Tables

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

CL General Form



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)

0/1

2^N rows (16)

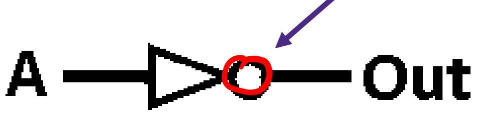
If we have N inputs, how many distinct functions F do we have?

2^N output "positions", each being 0/1

so $\boxed{2\text{^}(2\text{^}N) \text{ possible functions}}$

Logic Gate Names and Symbols

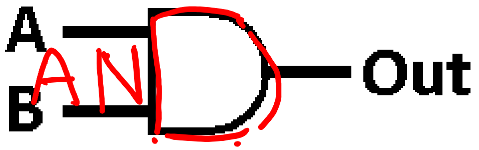
- NOT



Circle indicates NOT

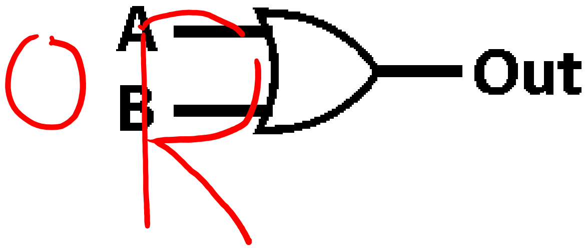
A	Out
0	1
1	0

- AND



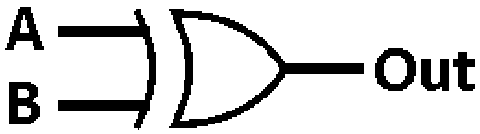
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR

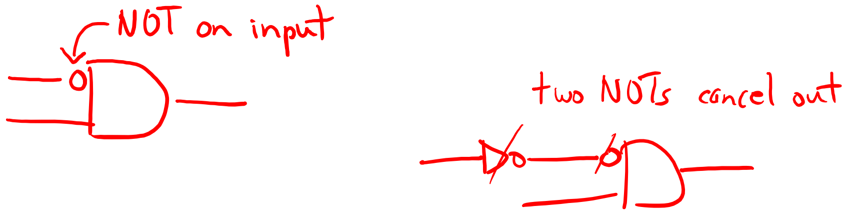


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

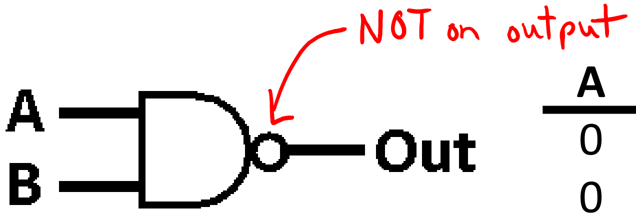
- XOR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

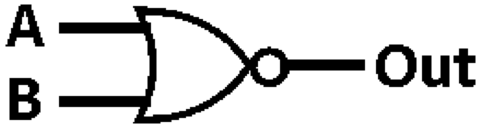


- NAND



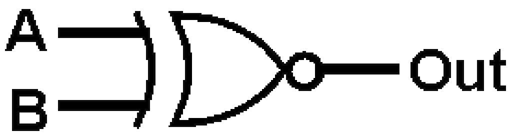
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

- NOR



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

- XNOR



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

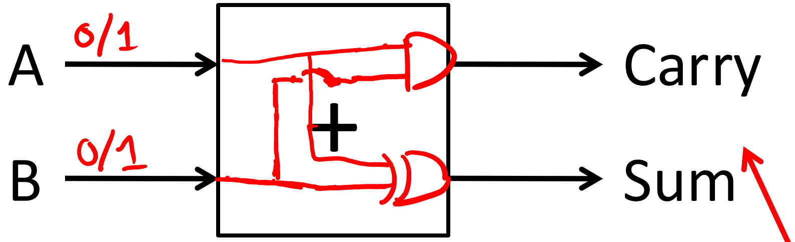
More Complicated Truth Tables

3-Input Majority

How many rows? $2^3 = 8$ 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



2 separate functions (columns)

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 $\overline{\mathrm{A}}$ 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.*, $\mathrm{A\ XOR\ B} = \mathrm{A} \oplus \mathrm{B} = \overline{\mathrm{A}}\mathrm{B} + \overline{\mathrm{B}}\mathrm{A}$

Truth Table to Boolean Expression

a	b	c	row
0	0	0	1
0	1	1	2
1	0	1	3
1	1	0	4

- 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

    sets to 1 when input combination matches
  - $C = \underset{\text{row 2}}{\overline{A}B} + \underset{\text{row3}}{\overline{B}A}$

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*

    sets to 0 when input combination matched
  - $C = \underset{\text{row 1}}{(A + B)} \cdot \underset{\text{row4}}{(\overline{A} + \overline{B})}$

Basic Boolean Identities

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

- $\mathrm{X} \cdot 1 = \mathrm{X}$
- $\mathrm{X} \cdot 0 = 0$
- $\mathrm{X} \cdot \mathrm{X} = \mathrm{X}$
- $\mathrm{X} \cdot \overline{\mathrm{X}} = 0$

Basic Boolean Algebra Laws

- **Commutative Law:**

  $X + Y = Y + X$ $\qquad$ $X \cdot Y = Y \cdot X$

- **Associative Law:**

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

- **Distributive Law:**

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

Advanced Laws (Absorption)

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

$$X + \overline{X}Y = X \cdot 1 + \overline{X}Y = X \cdot (1 + Y) + \overline{X}Y$$
$$= X + XY + \overline{X}Y$$
$$= X + Y \ ☑$$

Practice Problem

- Boolean Function: $F = \underline{\overline{X}YZ} + \underline{XZ}$

Truth Table:

X	Y	Z	F
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	0
1	1	1	1

Simplification:

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

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

distribution

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

absorption

$$= (X + Y)Z$$

distribute

$$= XZ + YZ$$

2 gates (1 OR, 1 AND)

3 gates (2 AND, 1 OR)

Which of these is "simpler"?

Technology Break

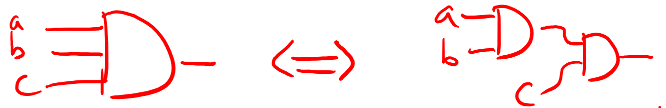
Lecture Outline

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

Why Is This Useful?

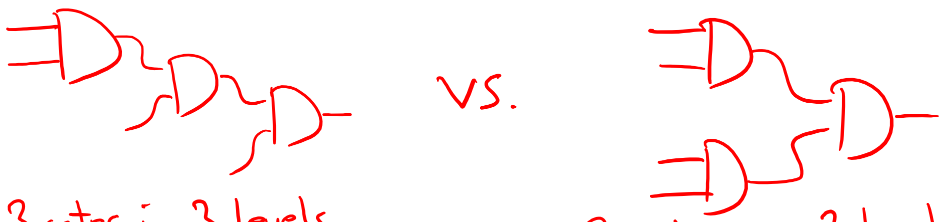
- Logic minimization: reduce complexity at gate level
  - Allows us to build smaller and faster hardware
  - Care about both # of gates, # of literals (gate inputs), # of gate levels, and types of logic gates

literals:



1, 3-input gate ⟺ 2, 2-input gates

levels:



3 gates in 3 levels vs. 3 gates in 2 levels (faster)

types: speed of NOT gate vs. AND gate?

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?

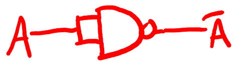
- No!

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

← simplest, but not too useful

useful, and simpler than alternatives

- 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!



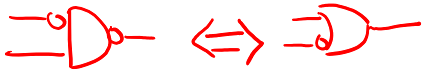
	x	y	NAND	
→	0	0	1	0→1
	0	1	1	
	1	0	1	
→	1	1	0	1→0

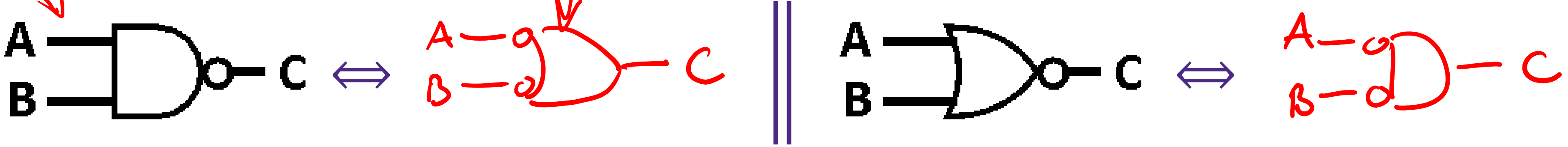
DeMorgan's Law

- $\overline{X+Y} = \overline{X}\cdot\overline{Y}$
- $\overline{X\cdot Y} = \overline{X}+\overline{Y}$

				NOR		NAND	
X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X+Y}$	$\overline{X}\cdot\overline{Y}$	$\overline{X\cdot Y}$	$\overline{X}+\overline{Y}$
0	0	1	1	1	1	1	1
0	1	1	0	0	0	1	1
1	0	0	1	0	0	1	1
1	1	0	0	0	0	0	0

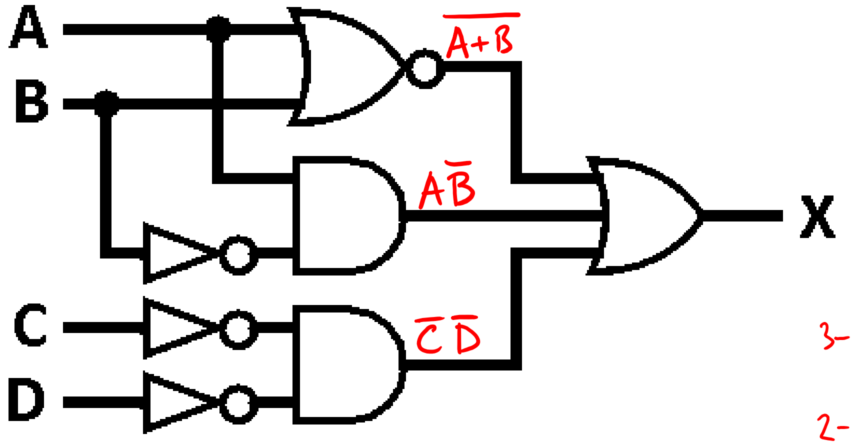
- In Boolean Algebra, converts between AND-OR and OR-AND expressions
  - $Z = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C$
  - $\overline{Z} = (A+B+\overline{C})\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+B+\overline{C})$
- At gate level, can convert from AND/OR to NAND/NOR gates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





DeMorgan’s Law Practice Problem

- Simplify the following diagram:



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

DeMorgan's

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

5 gates $X = \overline{B} + \overline{C}\,\overline{D}$

DeMorgan's

3-4 gates $X = \overline{B} + \overline{C+D}$

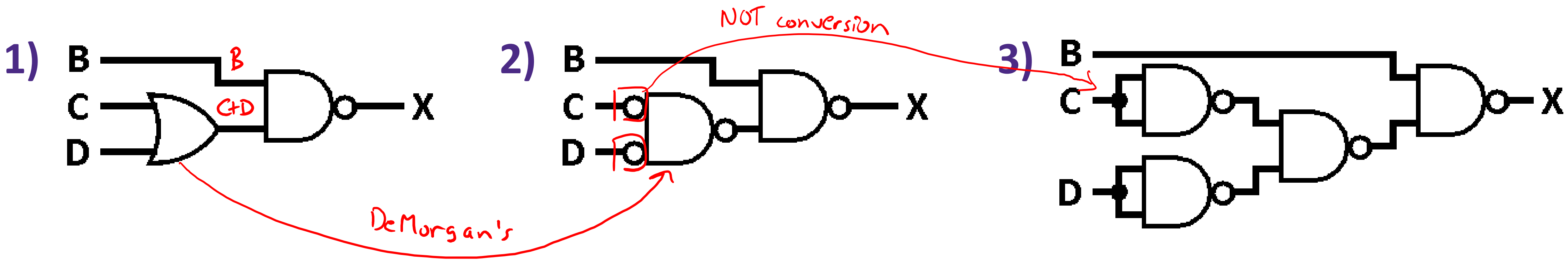
DeMorgan's

2-3 gates $X = \overline{B(C+D)}$

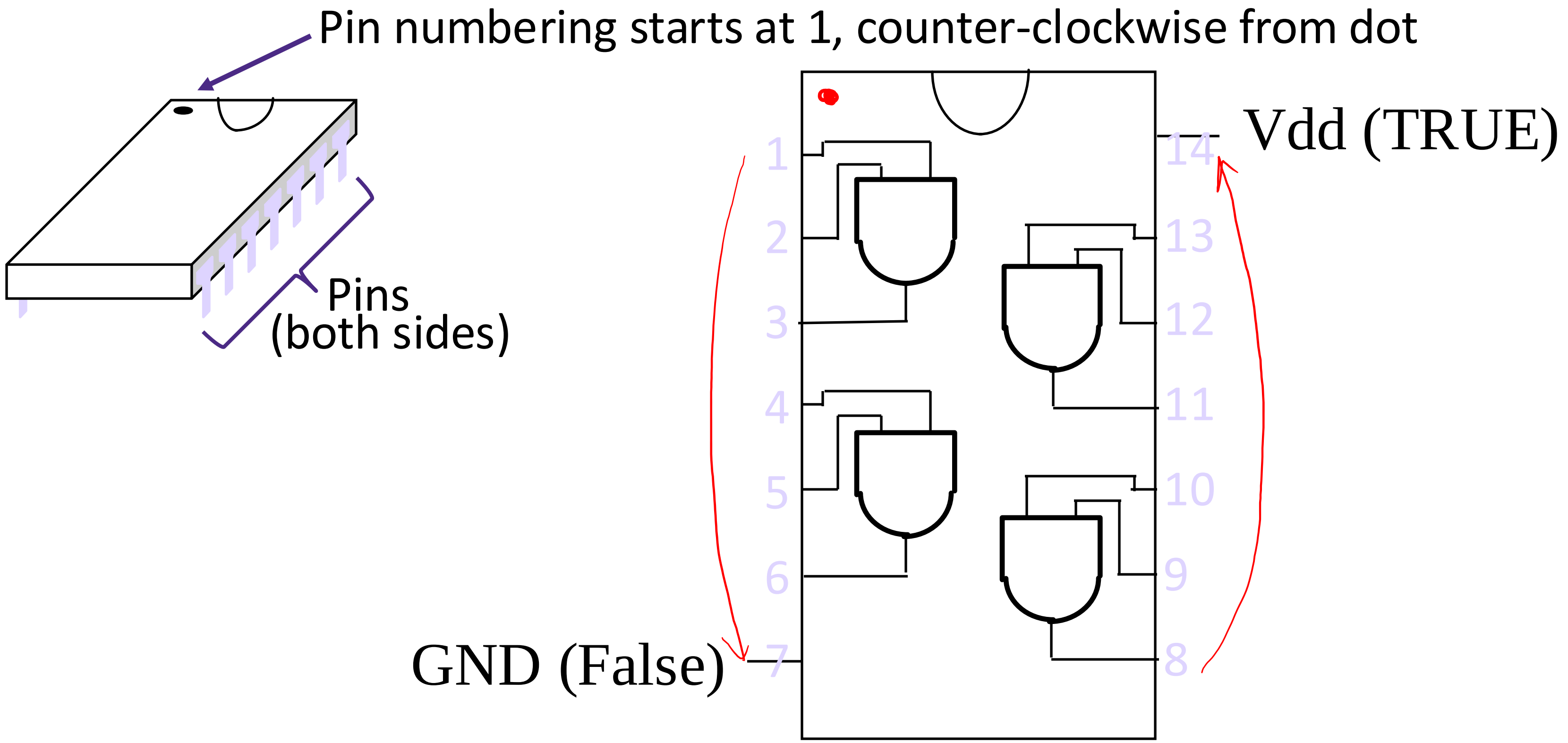
let $E = C+D$, so $X = \overline{B} + \overline{E}$

$X = \overline{BE}$

- Then implement with only NAND gates:



Transistor-Transistor Logic (TTL) Packages

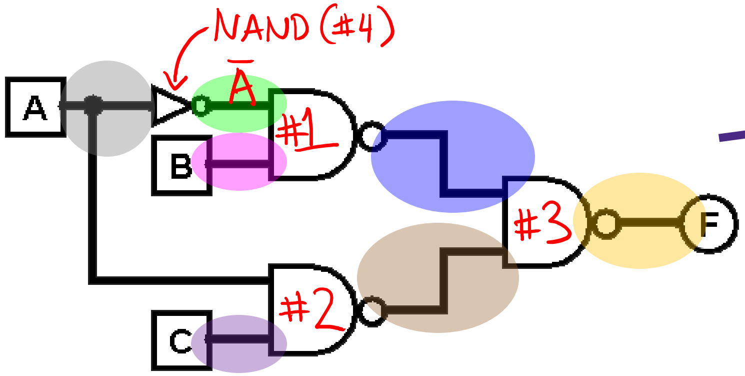

Pin numbering starts at 1, counter-clockwise from dot

Pins (both sides)

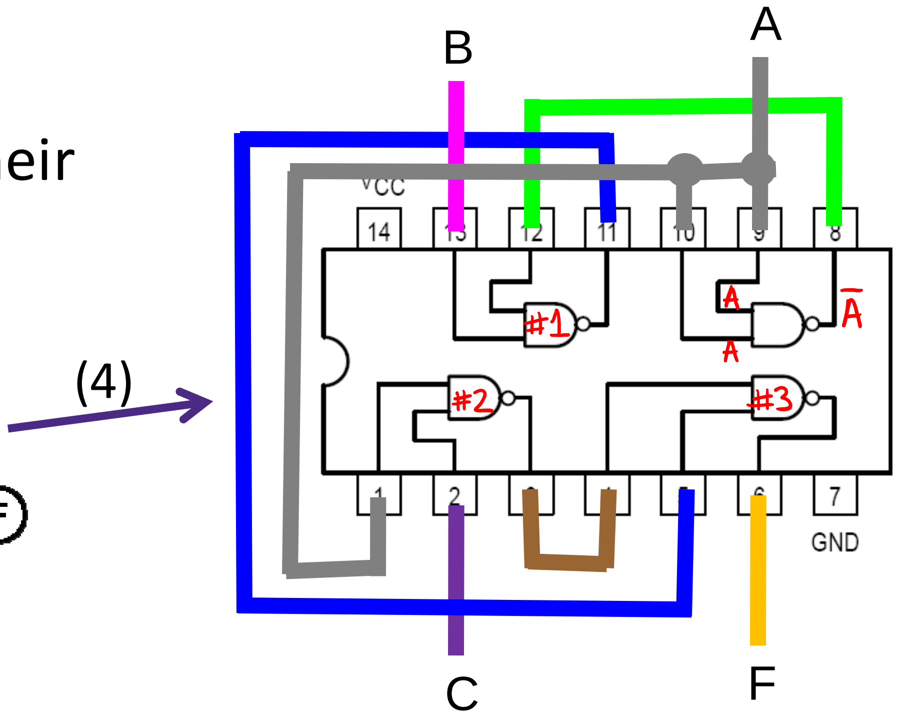
- Diagrams like these and other useful/helpful information can be found on part data sheets
  - It’s really useful to learn how to read these

Mapping truth tables to logic gates

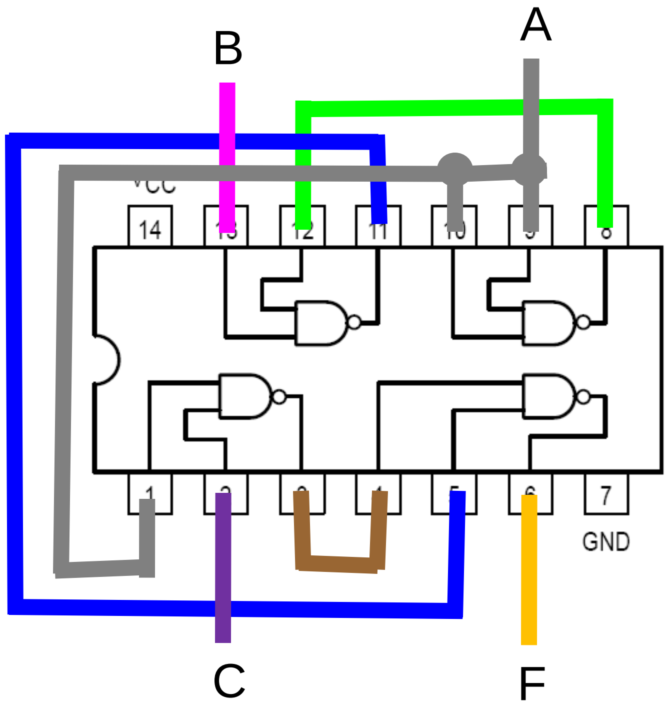
- Given a truth table:
  1) Write the Boolean expression
  2) Minimize the Boolean expression
  3) Draw as gates
  4) Map to available gates
  5) Determine # of packages and their connections

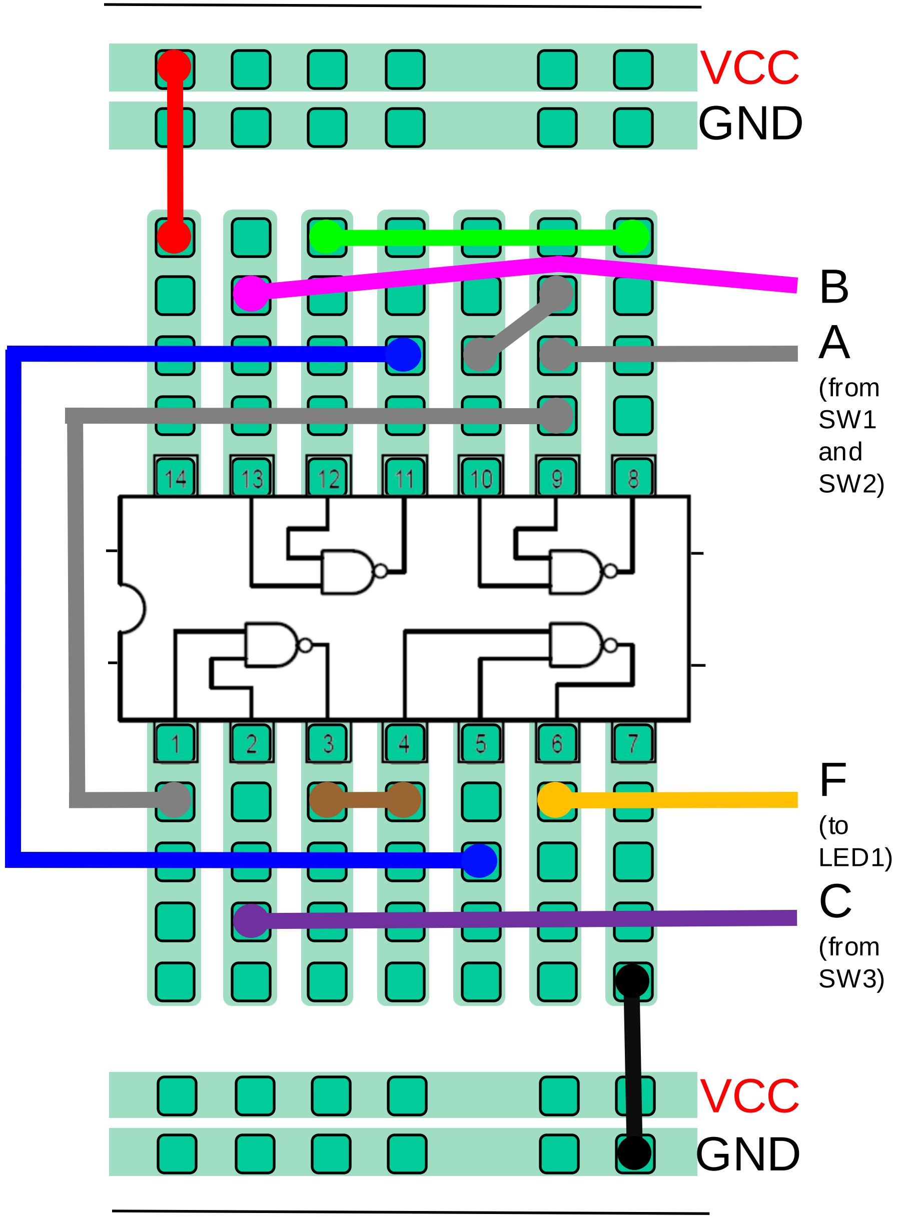


7 nets (wires) in this design



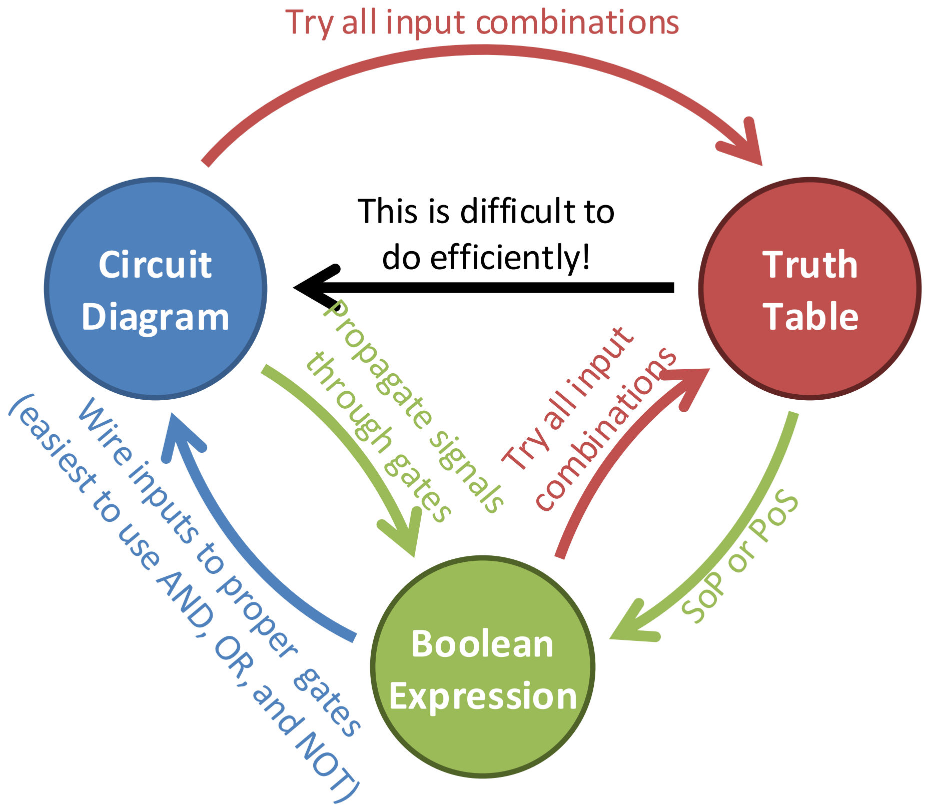
Breadboarding circuits





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


Try all input combinations
Circuit Diagram
This is difficult to do efficiently!
Truth Table
Propagate signals through gates
Try all input combinations
Wire inputs to proper gates (easiest to use AND, OR, and NOT)
SoP or PoS
Boolean Expression