CSE 2600
Intro. To Digital Logic & Computer Design

Bill Siever
&
Michael Hall

Announcements

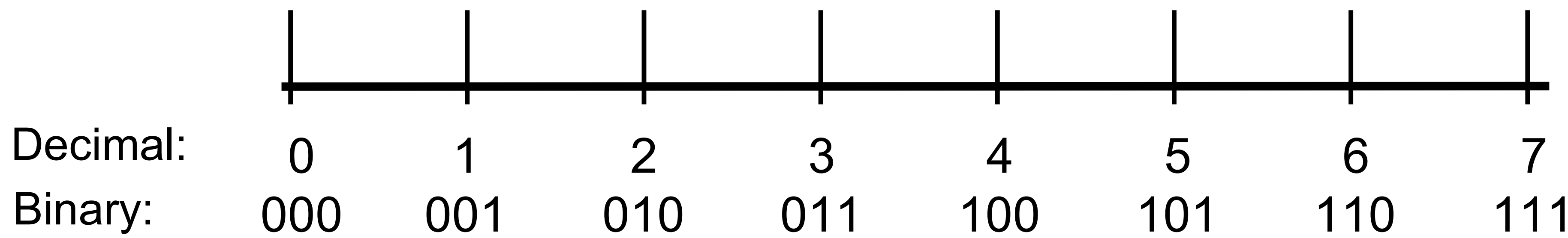
- Reminder: Prep quizzes are due Monday night with no makeup. Please plan ahead if you have a scheduled conflict; quizzes are always available.
- Homework 2B is posted / Due Sunday at 11:59pm
  - Dropboxes now available
- Office hours: Posted but there may be updates. See “help” page in Canvas or course site

Studio 2A Highlights

- Unsigned number line
- Two’s complement

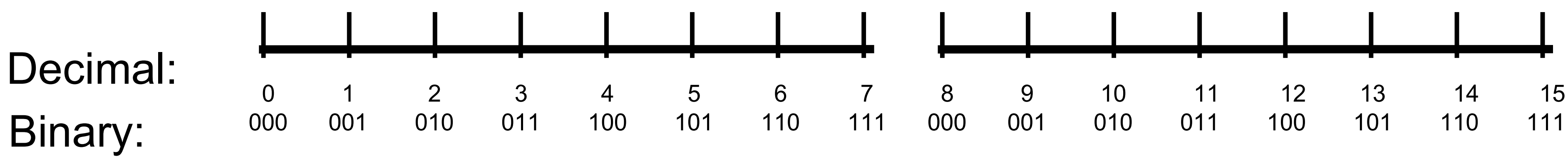
Last Time

- Studio: Binary Number Lines Extended



Last Time

- Studio: Binary Number Lines Extended



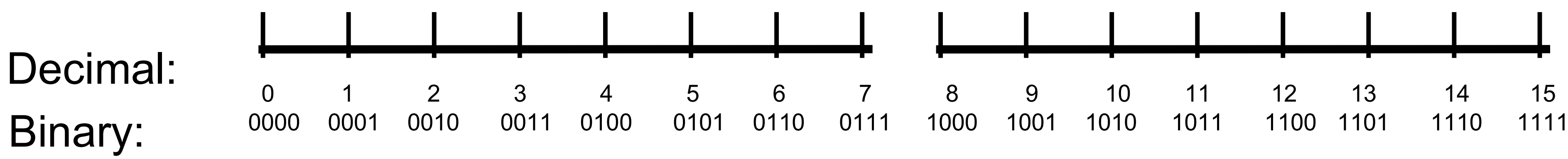
Last Time

- Studio: Binary Number Lines Extended

Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary:	**0**000	**0**001	**0**010	**0**011	**0**100	**0**101	**0**110	**0**111	**1**000	**1**001	**1**010	**1**011	**1**100	**1**101	**1**110	**1**111

Last Time

- Studio: Binary Number Lines Extended



Last Time

- Studio: Two's Complement

Decimal:	0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
Binary:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Last Time

- Studio: Two's Complement

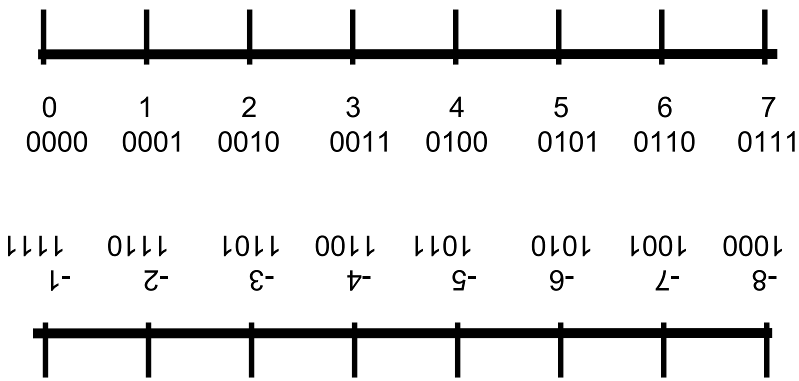
Decimal:	0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
Binary:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Last Time

- Studio: Two’s Complement - Above/Below

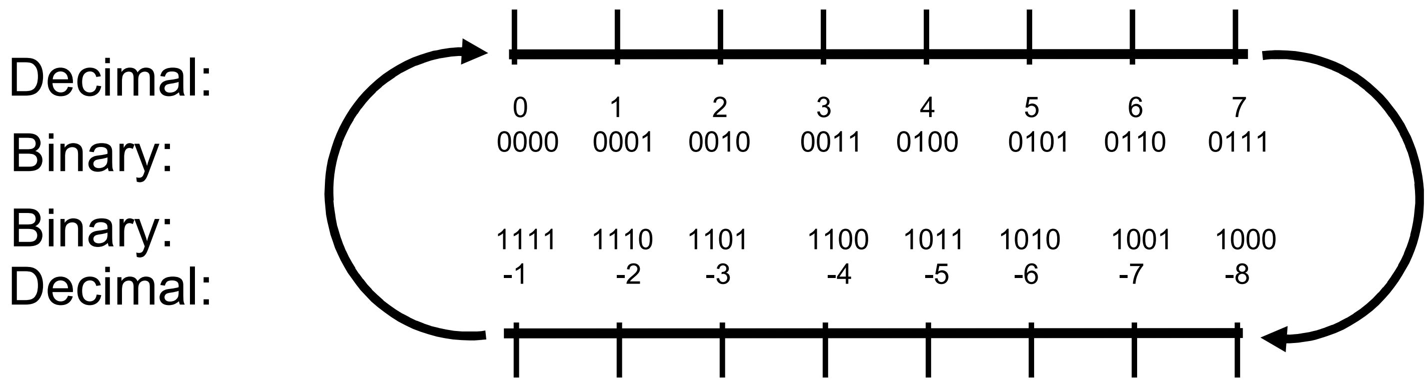
Decimal:

Binary:



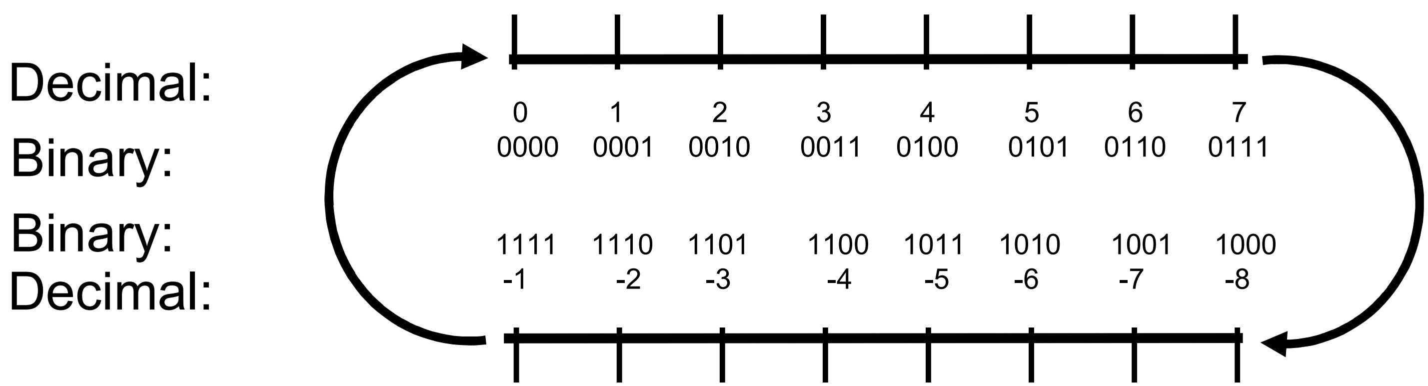
Last Time

- Studio: Two’s Complement - Above/Below & Bitwise Inversion



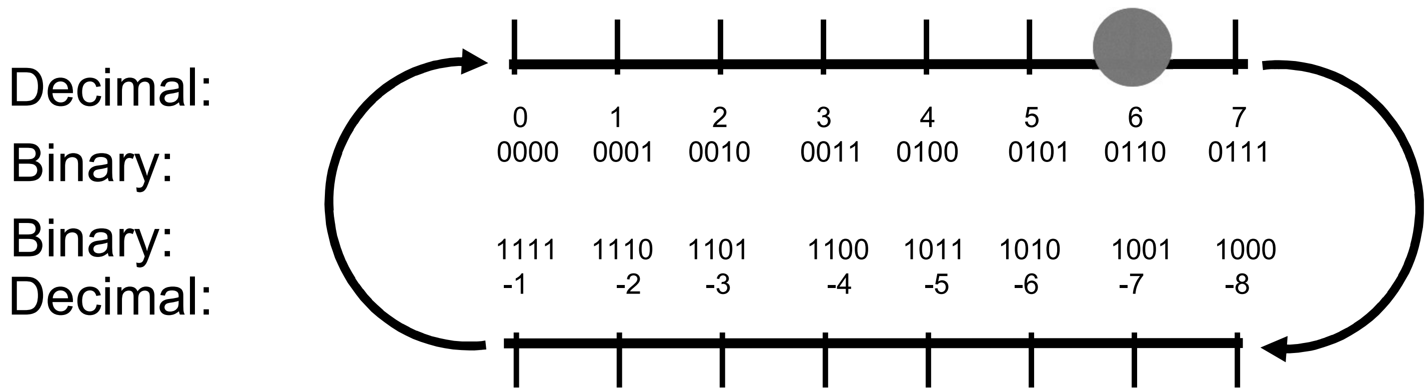
Last Time

- Studio: Two's Complement - Mathematical Negation $(-1\times)$



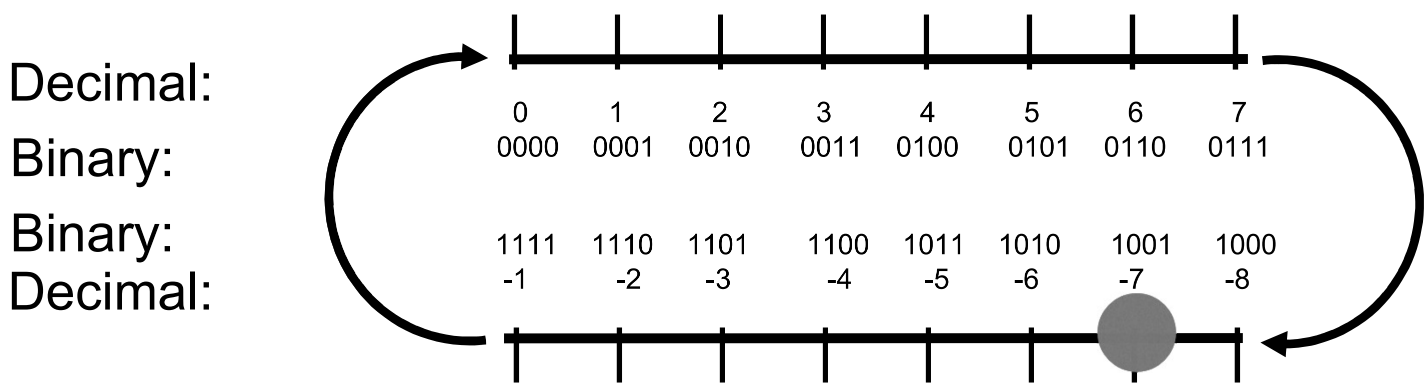
Last Time

- Studio: Two's Complement - Mathematical Negation (-1×6)



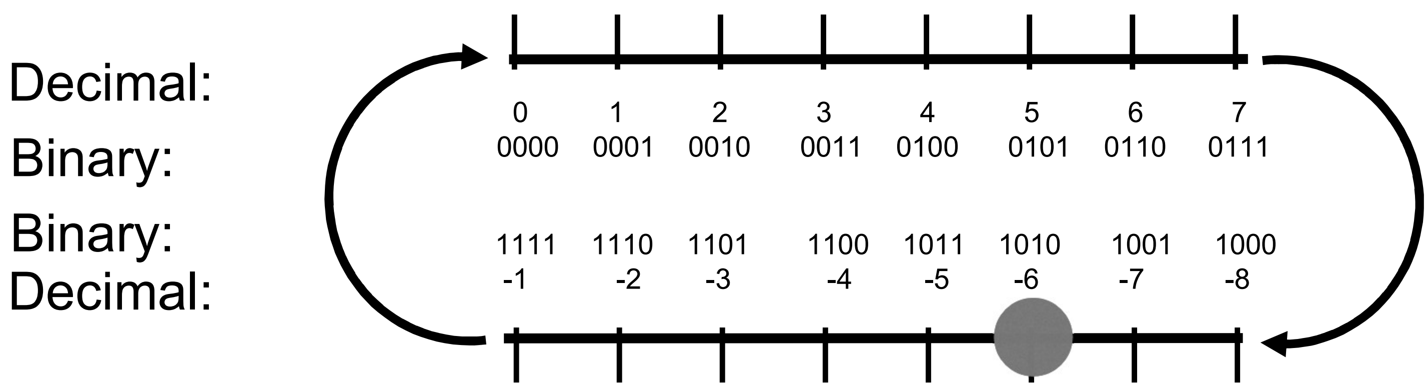
Last Time

- Studio: Two's Complement - Mathematical Negation (-1×6)



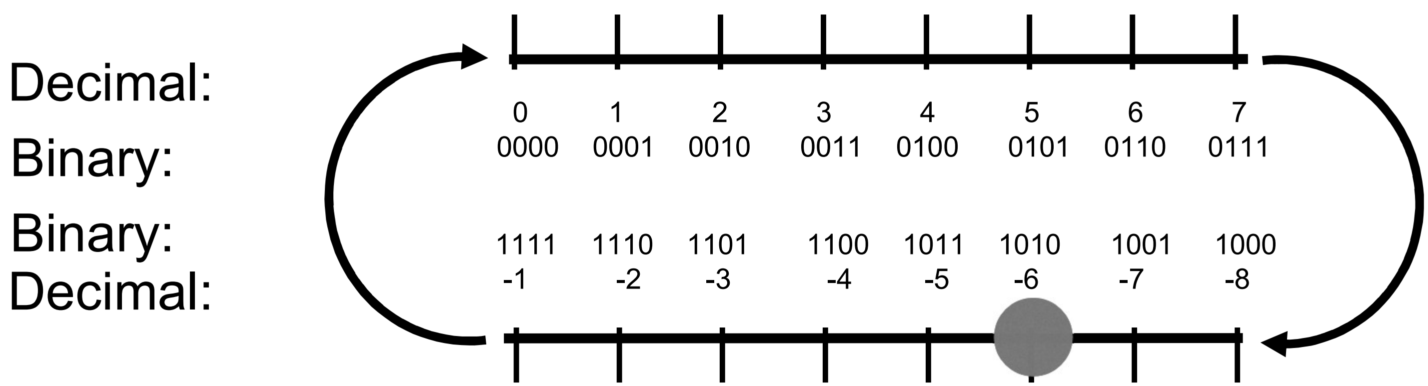
Last Time

- Studio: Two's Complement - Mathematical Negation (-1×6)



Last Time

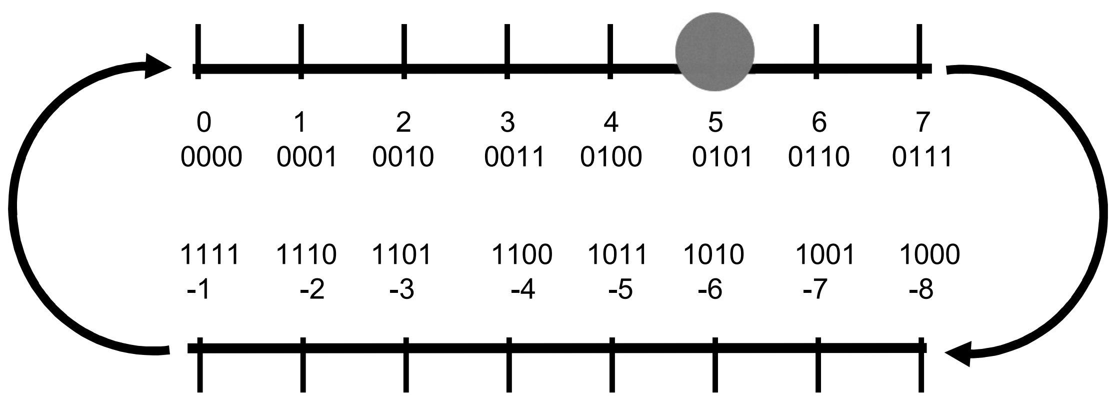
- Studio: Two's Complement - Mathematical Negation ($-1 \times$-6)



Last Time

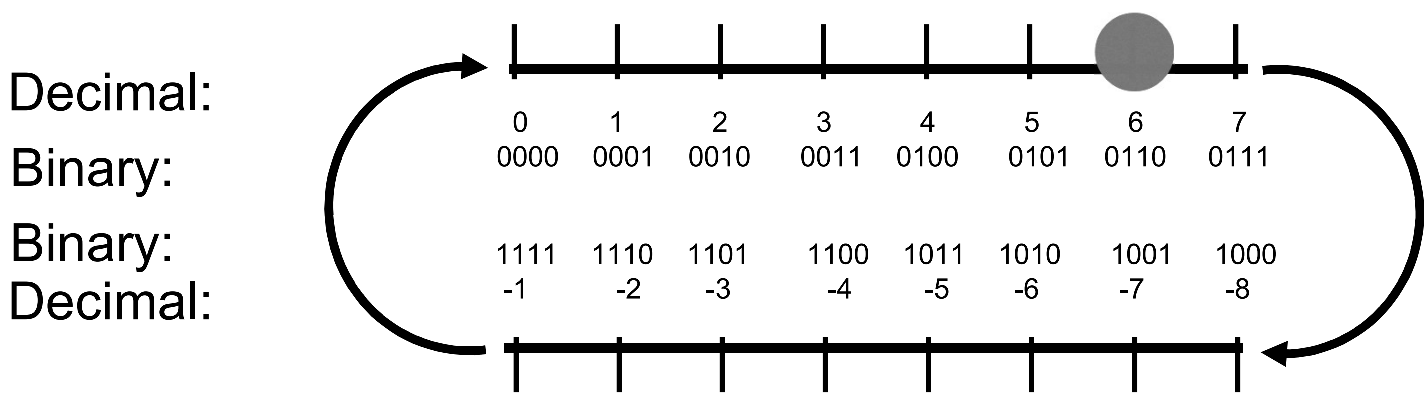
- Studio: Two's Complement - Mathematical Negation ($-1 \times$-6)

Decimal:
Binary:
Binary:
Decimal:



Last Time

- Studio: Two's Complement - Mathematical Negation (-1×-6)



Last Time

- Studio: Two's Complement - Mathematical Negation (-1 ×A)
- Logic: $\overline{A} + 1$
  - We are assumed to have "machines" for both;
    - Bitwise inversion (n invertors) and n-bit addition
      (more to come on n-bit addition)

Last Time

- Assume an n bit adder, like:

- n bit subtractor can be built:

Last Time: Tee

- $t_b + \overline{t_b}$
  - Theorem / Dual T5': $B + \overline{B} = 1$
  - Therefore "1"
- *Eventually*… JLS

Last Time

- JLS Logic Types: Unsigned & Assumes values are 0 at $t = 0$

Chapter 2

- Tables & Sum-of-products
  - A “can’t go wrong” way to build logic that behaves a specified way
- Karnaugh Maps: A form of optimization
- Timing: Delay of circuits

Background: Minterms

- Minterms: Given n variables, a product (AND) containing all n exactly once, in either their original or negated form

  - Consider: $\overline{A} \cdot B \cdot C \cdot \overline{D}$
    Identify all possible combinations of inputs which make it true:

Chapter 2: Minterms

- Consider $n = 3$ and A, B, C; Which are Minterms? Which are *not* and why?
  - ABC
  - $AB\overline{A}$
  - $CB\overline{A}$
  - $\overline{A}C$

Minterms & Truth Tables

CI	A	B		Carry Out	Sum
0+	0+	0	=	0	0
0+	0+	1	=	0	1
0+	1+	0	=	0	1
0+	1+	1	=	1	0
1+	0+	0	=	0	1
1+	0+	1	=	1	0
1+	1+	0	=	1	0
1+	1+	1	=	1	1

Minterms & Truth Tables

- Minterms are true for a *single* combination of inputs
  - This is essentially *selecting* a row of a truth table

Minterms & Truth Tables

CI	A	B			Sum
0+	0+	0	=		0
0+	0+	1	=		1
0+	1+	0	=		1
0+	1+	1	=		0
1+	0+	0	=		1
1+	0+	1	=		0
1+	1+	0	=		0
1+	1+	1	=		1

- $n = 3$: CI, A, B
- Where/when is Sum true (<u>any</u> place)?

Minterms & Truth Tables

CI	A	B			Sum
0+	0+	0	=		0
0+	0+	1	=		1
0+	1+	0	=		1
0+	1+	1	=		0
1+	0+	0	=		1
1+	0+	1	=		0
1+	1+	0	=		0
1+	1+	1	=		1

- $n = 3$: CI, A, B
- Where/when is *any* of these true?

$$Sum = \overline{CI} \cdot \overline{A} \cdot B + \overline{CI} \cdot A \cdot \overline{B} + CI \cdot \overline{A} \cdot \overline{B} + CI \cdot A \cdot B$$

Truth Table -> Sum of Minterms

Canonical Form

Important!

- Any simple function (mapping) can be represented as a truth table
  - n-bit binary numbers can be used to represent all the inputs
    - The table will need 2^n rows to represent all the possible combinations of inputs
  - m-bit binary numbers can represent the output(s)
  - *Each* of the m bits of output can be represent by a sum-of-products (sum of minterms) equation.
    - There's a minterm for each place the bit of m is a 1 (true)
    - Canonical form = The "sum" of these Minterms

Sum-of-Products

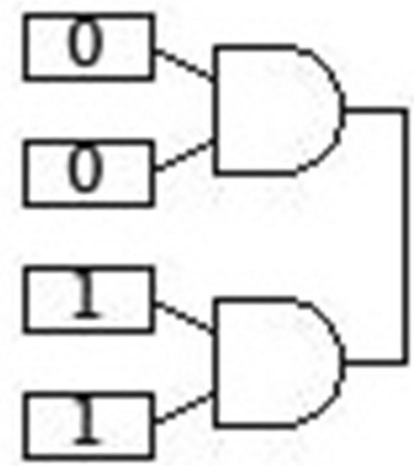
- All our combinational logic *could* be represented in a table
- All the outputs can be represented as equations
- Those equations can be realized with just the concept of AND, OR, & NOT
    - I.e., we can build computing machines for anything we can represent in a table if we have AND, OR, or NOT.
- The idea of Tables and “Look Up Tables” (LUTs) is really useful!

Product-Of-Sums

- Alternative to SOP: Uses maxterms (SUM of all input variables) in large product
  - Form: $Y = (A + B + C) \cdot (A + \overline{B} + C) \cdots$
- Can be constructed from table by focusing on:
  1) rows with zeros => each a “sum” for a zero in only that row
  and
  2) products that will combine them
- Sum-of-Prod: Smaller when fewer 1s than 0s in table;
  Otherwise Prod-of-Sums is smaller
- This class: slightly focused on Look-up Table (LUT) concept / usually favor SOP

Real Circuits: Xs and Zs

- 0s and 1s represent real-world, continuous values, like voltages
  - Ex: 0 = 0v (gnd);   1 = 5v
    - What's 2.3v?
    - What happens if a 0v wire is connected to 5v wire? ("Contention")
  - X: That's illegal / don't know



Simulator / Language Types

- Bits & Types: 10101100 can have different interpretations
  - Programming languages use data types
- Verilog (Chapter 4)'s logic type:
  - 0, 1, X (unknown), Z (high-impedance)
  - Other simulators often use X for initial value
    (Helps catch errors and misunderstandings earlier vs. building on a bad assumption!)

Real Circuits: Xs and Zs

- 0s and 1s represent real-world, continuous values, like voltages
  - Ex: 0 = 0v (gnd);   1 = 5v (relative to that ground)
    - Voltage is a *__relative__* measure
      (like water pressure: it's the difference between two points)
  - Z: "Floating" value / disconnected
    - Sometimes useful to "disconnect" something to prevent contention
      (to share wires with different things in control at different times)
    - Sometimes an error when *__nothing__* is connected
      (Behavior depends on technology and conditions; Can be random or influenced by external things — like moving a hand near a circuit!)

Circuit “Optimization”

- Time or performance?
- Number of parts?
- Total cost?
- Combination: E.g., Cheapest way to achieve a specific level of performance

Circuit “Optimization”

- Logic Minimization
  - Canonical Form is seldom the minimum number of parts
  - Can “combine” overlapping terms (implicants / product)
    - Prime Implicant: Can’t be further reduced
    - Ex: $A \cdot B \cdot C + A \cdot \overline{B} \cdot C$
      - True when $A \cdot C$. The B and $\overline{B}$ cancel

Karnaugh (K) Maps

- A ***visual*** way to do term optimization
- Rely on tables that allow easy identification of ways to combine implicants
  - Uses Gray code ordering, ***not counting order!!!***
- Only useful for up to 4 variables. I.e., small problems

Karnaugh (K) Maps

- Goal: Cover all 1s with circles
  - As few circles as possible & as large as possible
  - Span rectangles with sides of 1, 2, 4, or 8
  - Top/bottom and left/right wrap!

Give an opt. equation for…

Inputs				Output
S	D1	D0		O
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		0
1	1	0		1
1	1	1		1

		D1/D0			
		00	??	??	??
S	0				
	1				

Give an opt. equation for…

Inputs				Output
S	D1	D0		O
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		0
1	1	0		1
1	1	1		1

		D1/D0			
		00	01	11	10
S	0				
	1				

Give an opt. equation for…

Inputs				Output
S	D1	D0		O
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		0
1	1	0		1
1	1	1		1

		D1 D0			
		00	01	11	10
S	0	0	1	1	0
	1	0	0	1	1

D1 changes

D0 changes

Give an opt. equation for…

Inputs			Output
S	D1	D0	O
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

		D1 D0			
		00	01	11	10
S	0	0	1	1	0
	1	0	0	1	1

$\overline{S} \cdot D0$

$S \cdot D1$

E2: Give an opt. equation for…

Inputs				Output
A	B	C		O
0	0	0		1
0	0	1		0
0	1	0		1
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		1
1	1	1		0

		B/C			
		00	01	11	10
A	0				
	1				

E3: Give an equation for…

Inputs				Output
X	Y	Z		O
0	0	0		1
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		0

		Y/Z			
		00	01	11	10
X	0				
	1				

Don’t cares in K-maps

- Sometimes there are input combinations that can never occur
- Rather than specify a default value, we can use “X” for Don’t Care
- This often results in simpler Boolean equations

Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

		C/D			
		00	01	11	10
A/B	00				
	01				
	11				
	10				

Don’t cares in K-maps

- Sometimes there are input combinations that can never occur
- Rather than specify a default value, we can use “X” for Don’t Care
- This often results in simpler Boolean equations

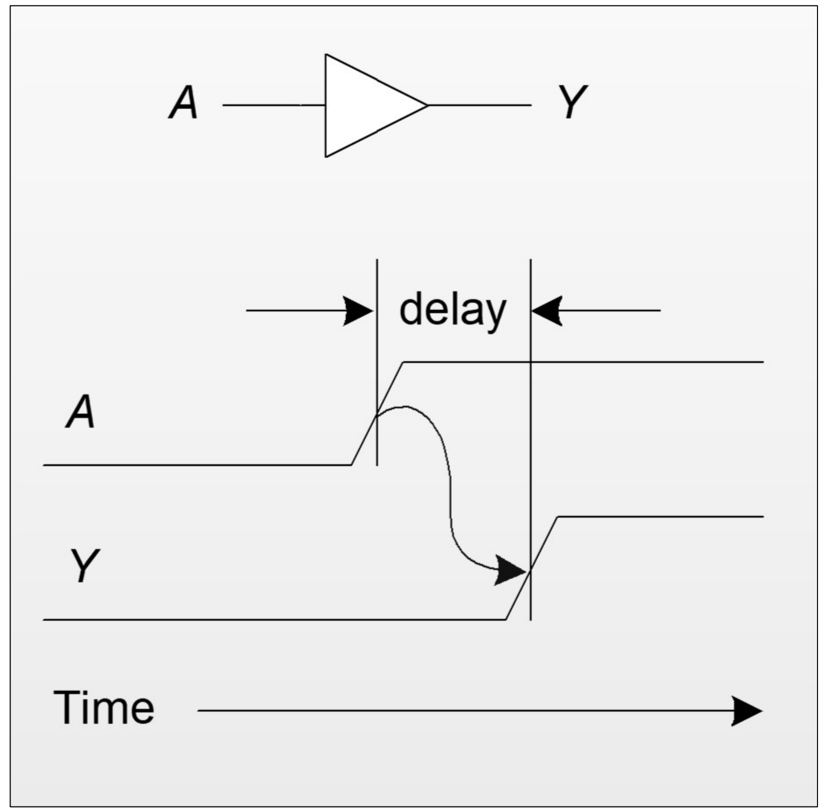
Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

		C/D			
		00	01	11	10
A/B	00	1	0	1	1
	01	0	X	1	1
	11	X	X	X	X
	10	1	1	X	X

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

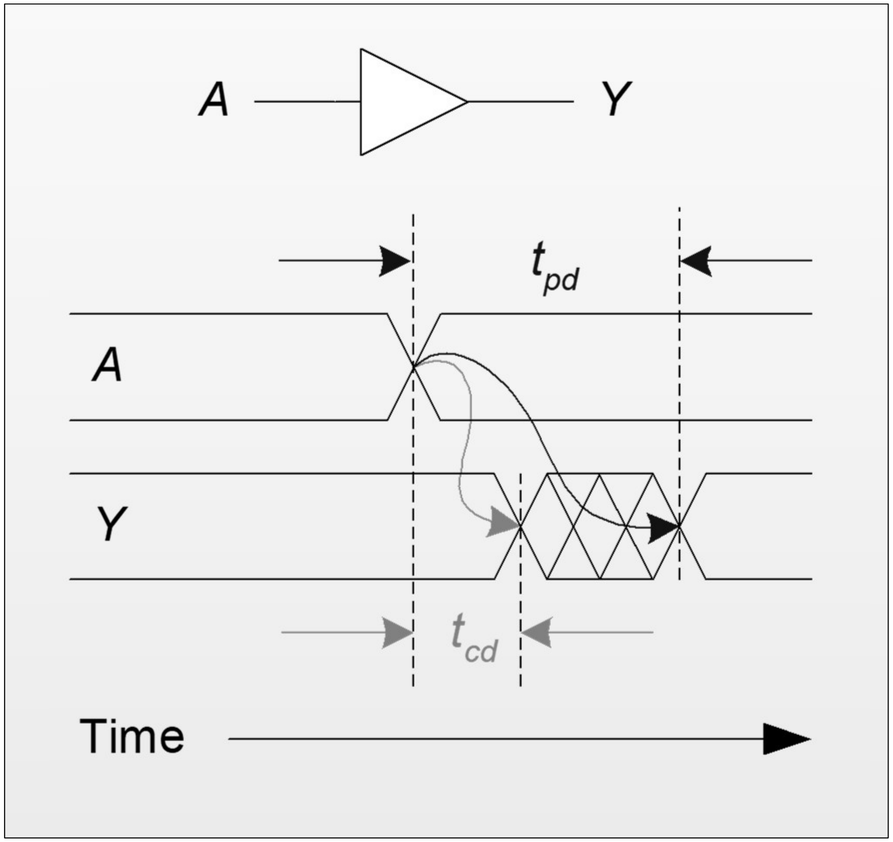
Timing

- Delay: time between input change and output changing
- Determines how fast we can build circuits
- Caused by
  - Capacitance and resistance in a circuit
  - Speed of light limitation



Propagation & Contamination Delay

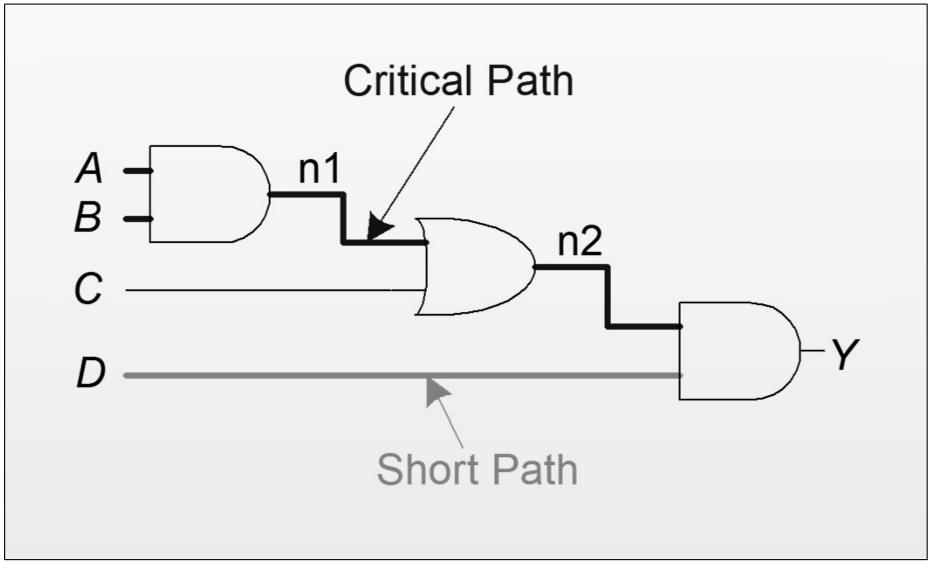
- Propagation delay: t_{pd} = **max** delay from input to output
- Contamination delay: t_{cd} = **min** delay from input to output
- Reasons why t_{pd} and t_{cd} may be different:
  - Different rising and falling delays (e.g., n- vs p-type transistors)
  - Multiple inputs and outputs, some of which are faster than others
  - Temperature of the circuit: Circuits slow down when hot and speed up when cold



Delay Calculation

Critical (Long) Path: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$
(max delay)

Short Path: $t_{cd} = t_{cd\_AND}$
(min delay)



2.8: More Parts

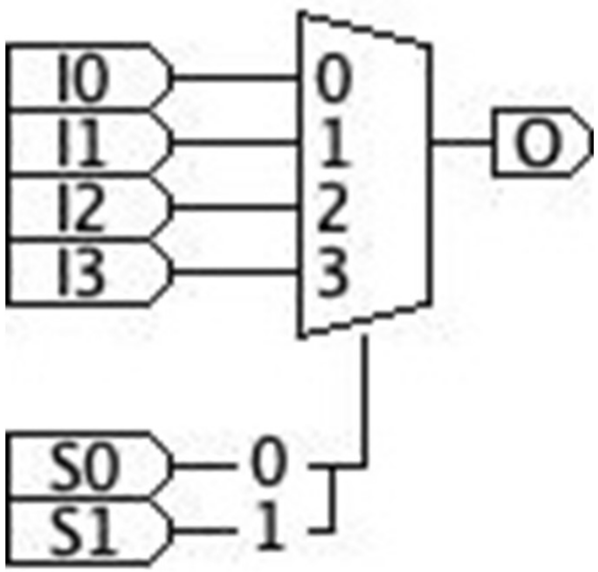
- Q: We want a 4-to-1 multiplexor. How big is the full truth table?
- Q: Our CPU may need a 32-to-1 multiplexor. How big is the truth table?
- We need a new approach
  - Hierarchical construction

Hierarchical Approach

- We've created a 2-to-1 MUX
  - Construct a 4-to-1 MUX using 2-to-1 MUXes
  - Focus on desired behavior using existing parts

4-to-1 MUX Behavior

- Behavioral Description as a Table



Inputs of Interest			Output (In terms of Inputs)
S1	S0		O
0	0		I0
0	1		I1
1	0		I2
1	1		I3

Hierarchical Construction of 4-input Mix

Questions

- I still wonder how serious hazards and glitches are in real hardware designs and how often they actually cause failures. Do designers usually eliminate all hazards, or do they sometimes rely on system timing to avoid problems?
- It is still unclear to me how the difference between contamination delay and propagation delay can induce glitches into a circuit.
- Karnaugh maps work well for up to four variables—what do we do for more than four variables, and when do K-maps become more complicated than Boolean algebra?
- I don’t fully understand how multiplexers and decoders are used to implement logic functions (like using a 4:1 MUX to build an AND or XNOR). Why is this useful instead of just using gates?
- What is the difference between X and Z, and is there ever a downside to treating don’t-care X values as 1s in Karnaugh maps?