CSE 260M / ESE 260
Intro. To Digital Logic & Computer Design

Bill Siever
&
Jim Feher

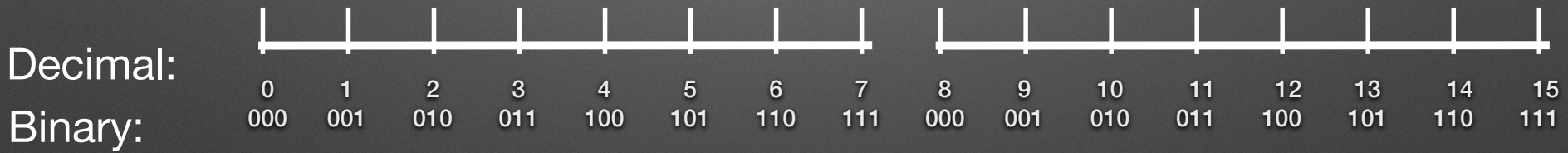
Last Time

- Studio: Binary Number Lines Extended

Decimal:	0	1	2	3	4	5	6	7
Binary:	000	001	010	011	100	101	110	111

Last Time

- Studio: Binary Number Lines Extended

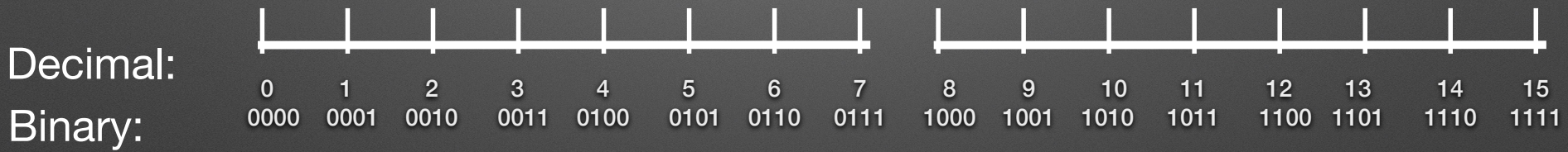


Last Time

- Studio: Binary Number Lines Extended

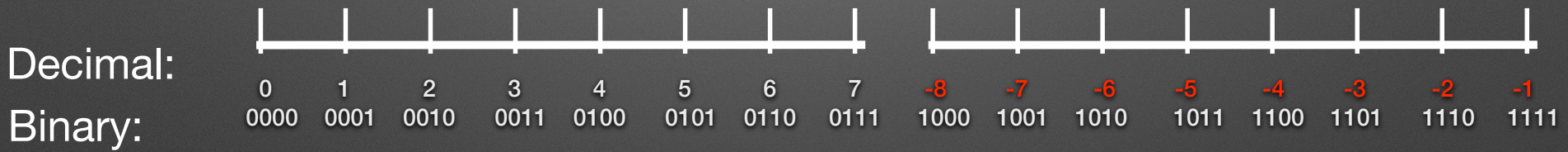
Last Time

- Studio: Binary Number Lines Extended



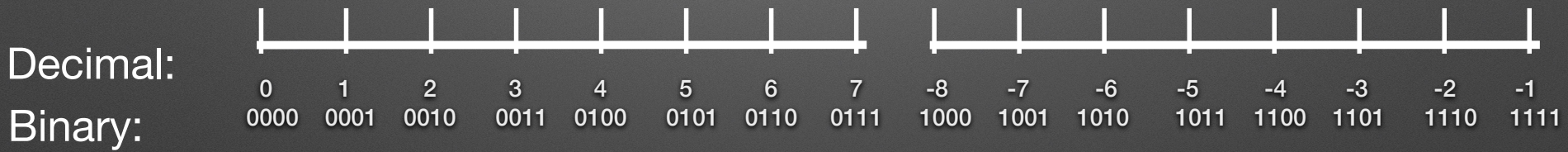
Last Time

- Studio: Two's Complement



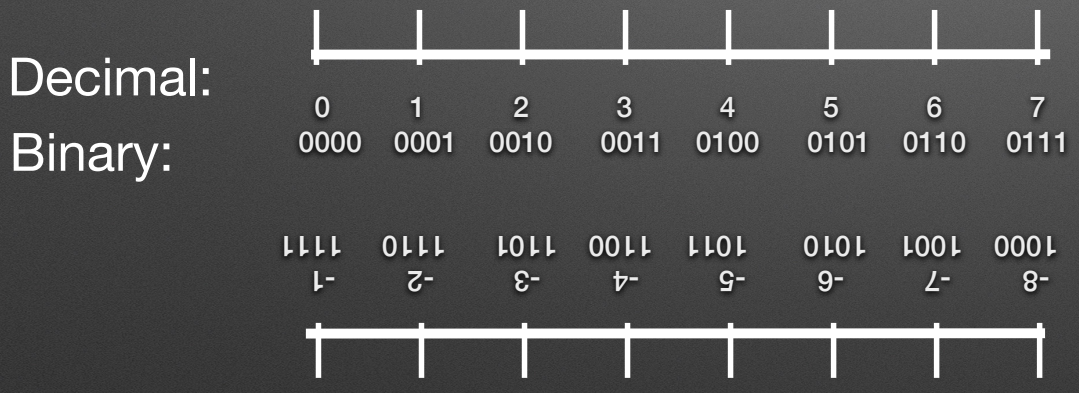
Last Time

- Studio: Two's Complement



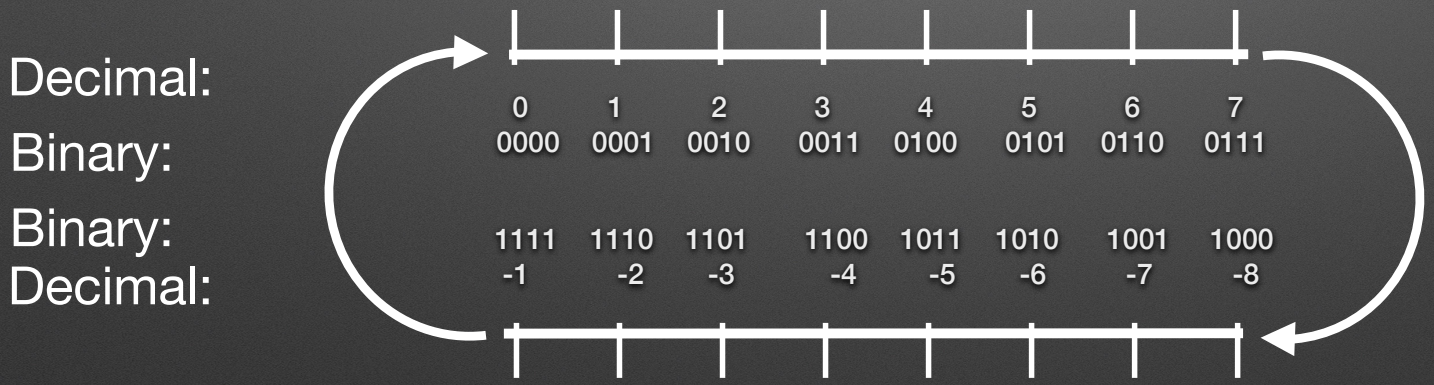
Last Time

- Studio: Two's Complement - Above/Below



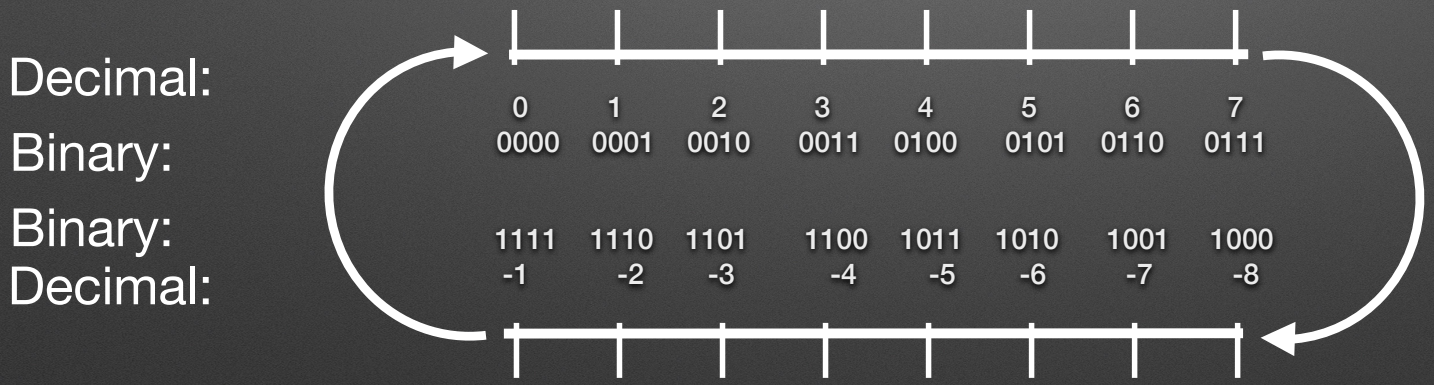
Last Time

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



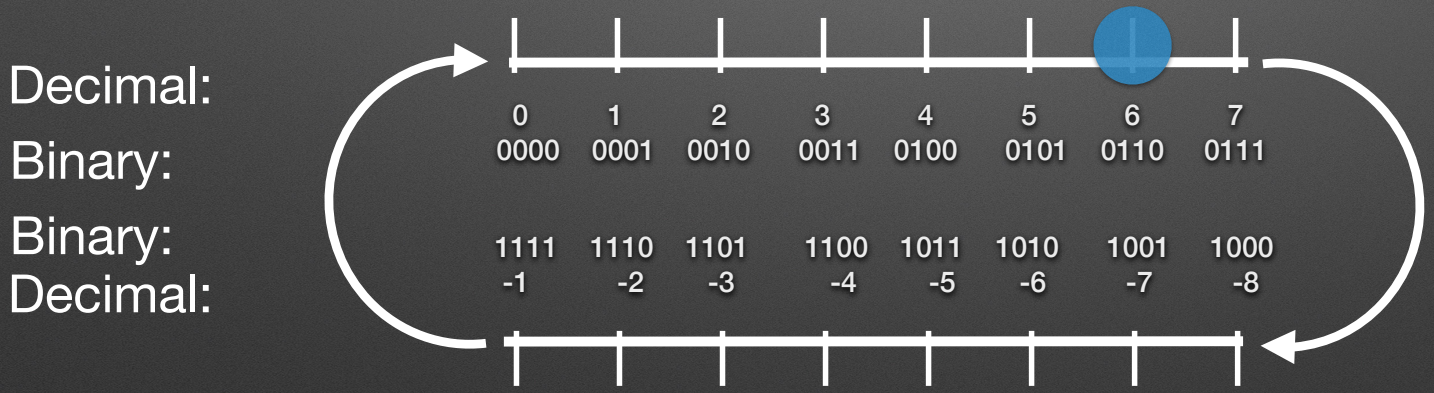
Last Time

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



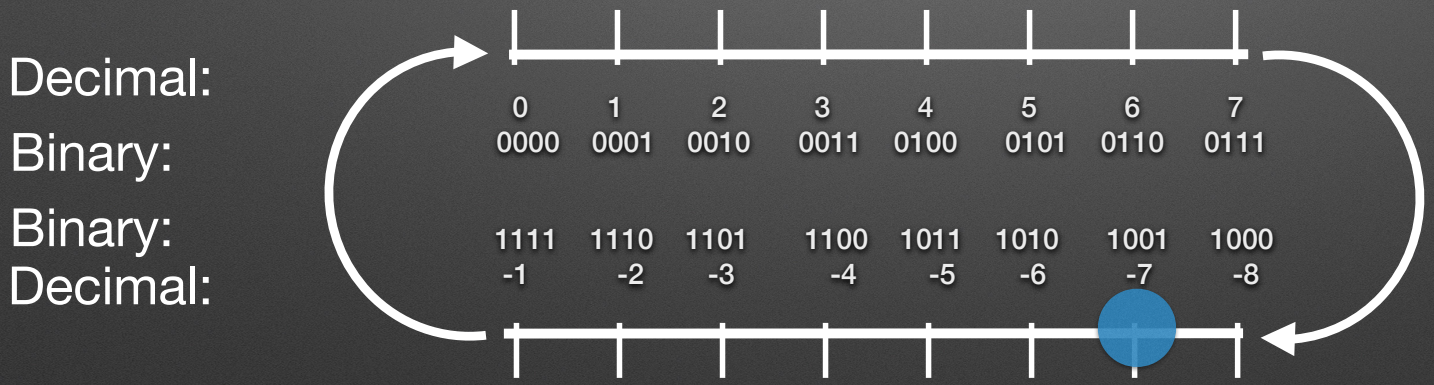
Last Time

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



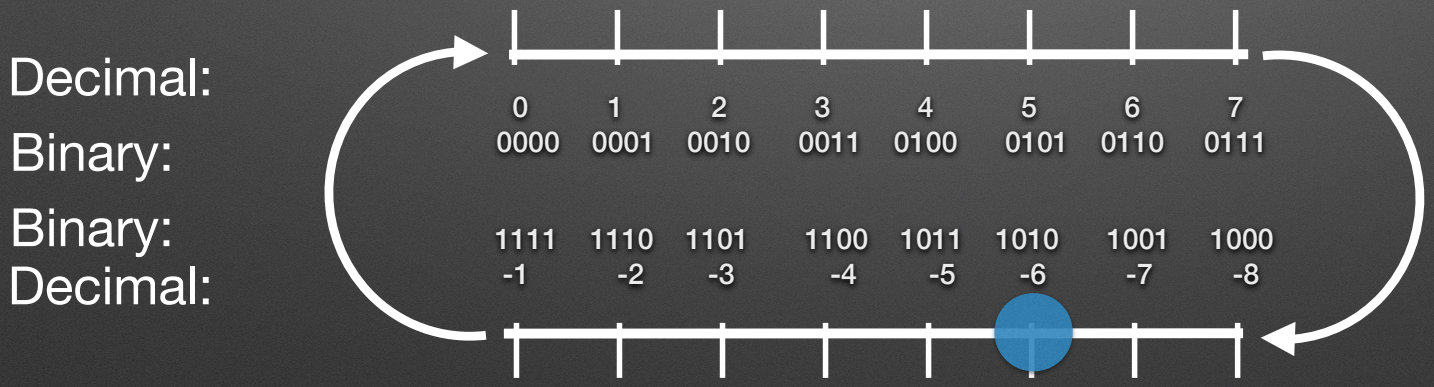
Last Time

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



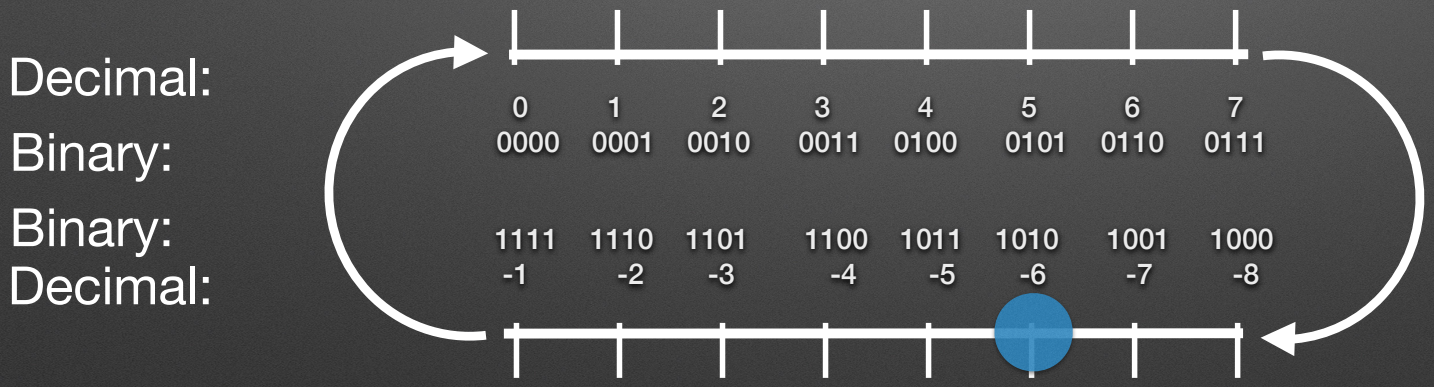
Last Time

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



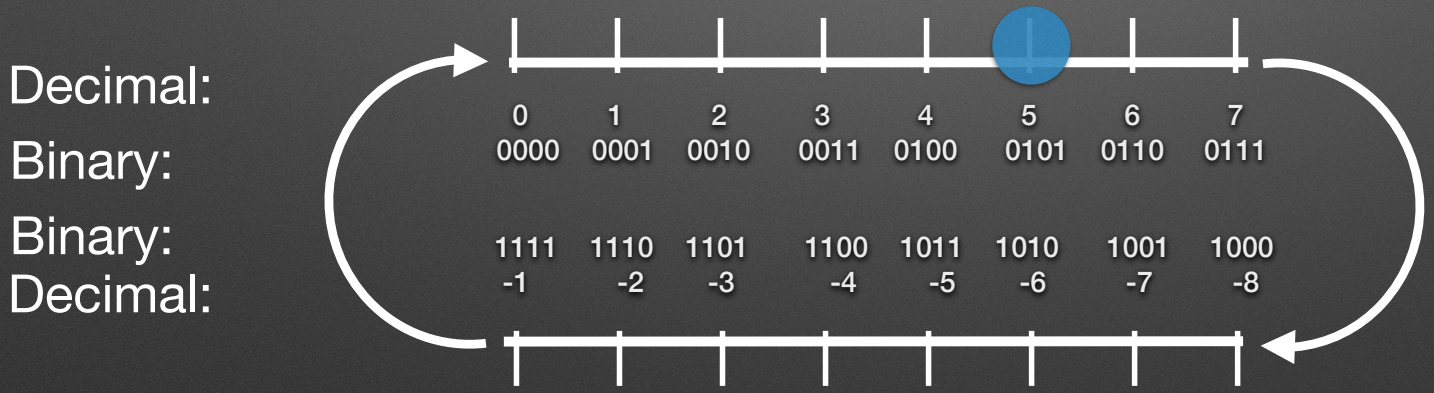
Last Time

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



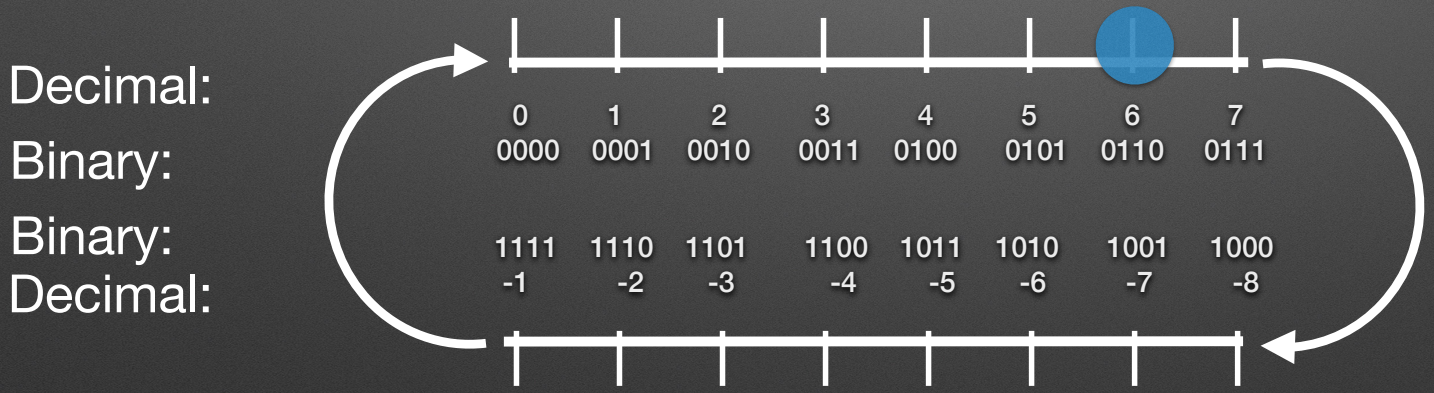
Last Time

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



Last Time

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



Last Time

- Studio: Two's Complement - Mathematical Negation ($-1 \times$ A)
- Logic: $\bar{A} + 1$

Chapter 2: Minterms

- Minterms
  - Given n variables, a product (AND) containing all n exactly once, in either their original or negated form
  - It's the minimum number of variables needed to exactly "select" a row in a truth table

Chapter 2: Minterms

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

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

Cl	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

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 each of these true?
  - $\bar{C}I \cdot \bar{A} \cdot B$
  - $\bar{C}I \cdot A \cdot \bar{B}$
  - $CI \cdot \bar{A} \cdot \bar{B}$
  - $CI \cdot A \cdot B$

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?

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?

$$Sum = \bar{C}I \cdot \bar{A} \cdot B + \bar{C}I \cdot A \cdot \bar{B} + CI \cdot \bar{A} \cdot \bar{B} + CI \cdot A \cdot B$$

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 combination 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 and OR
  - I.e., we can build computing machines for anything we can represent in a table if we have AND and OR.
- The idea of Tables and “Look Up Tables” (LUTs) is really useful!

Give an equation for…

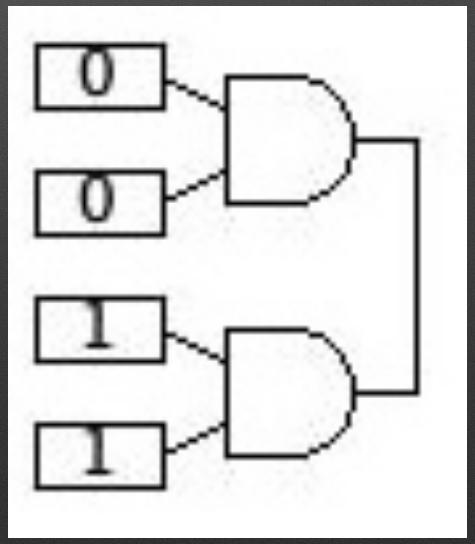
Inputs			Output
A	B	C	O
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Give an 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

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



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)
    - Prime Implicant: Can’t be further reduced
    - Ex: $A \cdot B \cdot C + A \cdot \bar{B} \cdot C$
      - True when $A \cdot C$. The B and $\bar{B}$ cancel

Karnaugh (K) Maps

- A way to do term optimization
- Rely on tables that allow easy identification of ways to combine implicants
  - Uses Gray code ordering, *not counting order!!!*
  - Terms are simplified by commingling in horizontal or vertical groups
    - Groupings must have a power of 2 items: 1, 2, 4, or 8.
  - Only useful for up to 4 variables. I.e., small problems

Give an 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

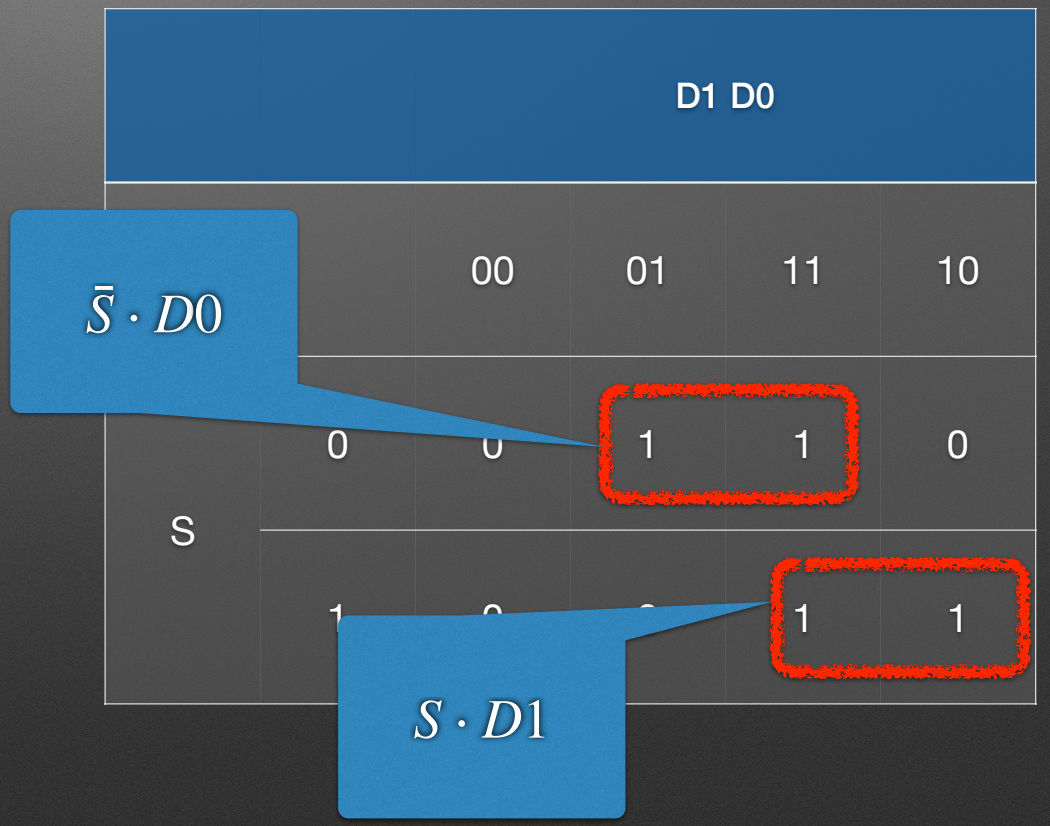
		D1/D0			
		00	01	11	10
S	0				
	1				

Give an 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

Give an 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



2.8: More Parts

- Multiplexor (MUX)
  - 2-input Mux Behavior
- 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

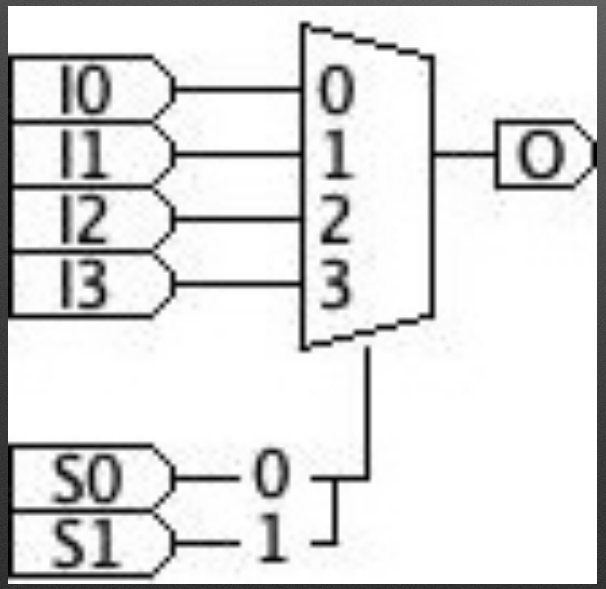
Hierarchical Approach

- 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

Hierarchical Construction of 4-input Mix

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

More Parts

Questions

- [What is the meaning of life? / What is success]
- [What about optimizing larger things? / Limit of K-Maps]
  - Quine–McCluskey algorithm
- [What were people doing between Babbage's time Turing's time?]
  - Industrial Revolution; Advance of Science; Digital Computing took convergence of many things…
- "What's your favorite gate?" — Stargate Defender (old computer game)?
- [What's computer architecture?] (Wait until chapters 6-7)

Next Time

- Studio
- Homework 1 due Thursday night!