

CNS 188a Overview

- Boolean algebra as an axiomatic system
- Boolean functions and their representations using Boolean formulas and spectral methods
- Implementing Boolean functions with **relay** circuits, circuits of **AON** (AND, OR, NOT) gates and **LT** (Linear Threshold) gates
- Analyzing the complexity (size and depth) of circuits
- Relations (as opposed to functions) and their implementation in circuits
- Feedback and convergence in LT circuits

Basic Axioms and Properties

- **A1. Identities:**

$$a + 0 = a \quad \text{and} \quad a \cdot 1 = a$$

- **A2. Complements:**

$$a + \bar{a} = 1 \quad \text{and} \quad a \cdot \bar{a} = 0$$

- **A3. Commutativity:**

$$a + b = b + a \quad \text{and} \quad a \cdot b = b \cdot a$$

- **A4. Distributivity:**

$$a + b \cdot c = (a + b) \cdot (a + c) \quad \text{and} \quad a \cdot (b + c) = a \cdot b + a \cdot c$$

- **L1. Self Absorption:**

$$a + a = a \quad \text{and} \quad a \cdot a = a$$

- **L2: Simple Absorption:**

$$a + 1 = 1 \quad \text{and} \quad a \cdot 0 = 0$$

- **T3. Associativity:**

$$(a + b) + c = a + (b + c)$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

- **T4. DeMorgan Laws:**

$$\overline{(a + b)} = \bar{a} \cdot \bar{b}$$

$$\overline{(a \cdot b)} = \bar{a} + \bar{b}$$

- **T0. Duality:**

Correctness is maintained when interchange $+$ and $\cdot$, as well as 0 and 1 .

- **T1. Distinct Complement:**

Every element has another element that is its unique complement.

- **T2. Absorption:**

$$a + ab = a \quad \text{and} \quad a \cdot (a + b) = a$$

Last Time

Representation Theorem (Stone 1936):

Every *finite* Boolean algebra is isomorphic to a Boolean algebra of subsets of some *finite* set S .

0-1 Theorem:

An identity is true for **any** finite Boolean algebra if and only if it is true for a **two-valued (0-1)** Boolean algebra.

Stone's Amazing Theorem

Representation Theorem (Stone 1936):

Every *finite* Boolean algebra is isomorphic to a Boolean algebra of subsets of some *finite* set S .

Two Boolean Algebras with m elements are isomorphic

Every Boolean algebra has 2^k elements

Intuition beyond the axioms:

We can 'naturally' reason about results in Boolean algebra

The 0-1 Theorem

0-1 Theorem:

An identity is true for **any** finite Boolean algebra if and only if it is true for a **two-valued (0-1)** Boolean algebra.

Proof: The easy direction

Assume an identity true for any finite Boolean algebra

 True for 0-1 Boolean algebra

The 0-1 Theorem

0-1 Theorem:

An identity is true for **any** finite Boolean algebra if and only if it is true for a **two valued (0-1)** Boolean algebra.

Proof: The non-obvious direction

Assume an identity true for a 0-1 Boolean algebra

Need to prove true for any finite Boolean algebra

The key: Stone's representation theorem

The 0-1 Theorem

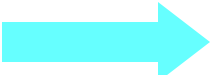
Proof (cont.): Assume an identity true for 0-1 Boolean algebra
Need to prove true for any finite Boolean algebra
The key: Stone's representation theorem

By contradiction:

Assume that there exists an identity that is **true** for 0-1 Boolean algebra but is **not true** for some other Boolean algebra.



By Stone: The identity is **not true** for some assignment of **binary vectors** with bitwise OR / AND



There exists a **0-1 assignment** to the identity that violates the equality, **CONTRADICTION!!**

Absorption Theorem

Theorem 2: $a + a \cdot b = a$

Proof (for 0-1 algebra):

0	+	0	·	0	=	0
0	+	0	·	1	=	0
1	+	1	·	0	=	1
1	+	1	·	1	=	1

Claim: True for any length binary vectors with bitwise OR/AND

Why? True bitwise!

Absorption Theorem

Theorem 2: $a + a \cdot b = a$

Proof (for 0-1 algebra):

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

$$\boxed{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \begin{pmatrix} a_1 \cdot b_1 \\ a_2 \cdot b_2 \\ a_3 \cdot b_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}}$$

The 0-1 Theorem: Why Only Identities?

Claim: Assume 0-1 Boolean algebra

$$\text{If } a + b = 1$$

$$\text{Then } a = 1 \text{ or } b = 1$$

Proof:

xy	OR(x,y)
00	0
01	1
10	1
11	1

The 0-1 Theorem: Why Only Identities?

Claim: Arbitrary Boolean algebra

$$\text{If } a + b = 1$$

$$\text{Then } a = 1 \text{ or } b = 1$$

Proof: Claim is NOT TRUE!

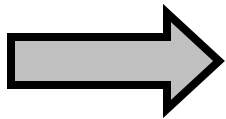
$\cup$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\emptyset$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\{a\}$	$\{a\}$	$\{a\}$	$\{a, b\}$	$\{a, b\}$
$\{b\}$	$\{b\}$	$\{a, b\}$	$\{b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$

We Just Solved HW#1 - 5b...

Equality Properties

- (a) Let $a + b = 0$, can you conclude that both $a = 0$ and $b = 0$?
- (b) Let $a + b = 1$, can you conclude that either $a = 1$ or $b = 1$?
- (c) Prove that $a = b$ if and only if $a \cdot \bar{b} + \bar{a} \cdot b = 0$.

(c) is useful for proving that two formulas **a** and **b** are **equal**



Identity checking by checking
if a formula is **identically zero**

Q: How to check if a formula is **identically zero**?

Satisfiability (the SAT problem)

Q: How to check if a formula is identically zero?

A1: use the 0-1 Theorem and check all 0-1 assignments

A2: transform the formula to a canonical form (**representation**)

A3: efficient **SAT** algorithms, P vs NP, applications in **CAD**, ...

SAT = is there a **satisfiable** assignment?

CAD = **Computer Aided Design**

Boolean Functions

Boolean formula with n variables

Assign all possible elements to the formula

A Boolean function is a mapping defined by a Boolean formula

$$a + b$$

ab	OR(a,b)
00	0
01	1
10	1
11	1

$$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$$

ab	XOR(a,b)
00	0
01	1
10	1
11	0

Boolean Functions

$$x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

$\oplus$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\emptyset$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\{a\}$	$\{a\}$	$\emptyset$	$\{a, b\}$	$\{b\}$
$\{b\}$	$\{b\}$	$\{a, b\}$	$\emptyset$	$\{a\}$
$\{a, b\}$	$\{a, b\}$	$\{b\}$	$\{a\}$	$\emptyset$

Representation

Disjunctive (Additive) Normal Form (DNF):

Sum of terms, each term is Normal

Normal = contains all the variables or their complements

$$\bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c}$$

DNF Theorem

DNF Theorem:

Every Boolean function can be expressed in DNF.

$$\begin{aligned}\overline{(a \cdot \bar{b} + a \cdot c)} + \bar{a} &= \overline{(a \cdot \bar{b})} \cdot \overline{(a \cdot c)} + \bar{a} \\ &= (\bar{a} + b)(\bar{a} + \bar{c}) + \bar{a} \\ &= \bar{a} + b \cdot \bar{c} + \bar{a} \\ &= \bar{a} + b \cdot \bar{c}\end{aligned}$$

Absorption:

- A2. Complements:
 $a + \bar{a} = 1$ and $a \cdot \bar{a} = 0$

- T4. DeMorgan Laws:

$$\begin{aligned}\overline{(a + b)} &= \bar{a} \cdot \bar{b} \\ \overline{(a \cdot b)} &= \bar{a} + \bar{b}\end{aligned}$$

- A4. Distributivity:

$$a + b \cdot c = (a + b) \cdot (a + c) \quad \text{and} \quad a \cdot (b + c) = a \cdot b + a \cdot c$$

$$\begin{aligned}&= \bar{a} \cdot (b + \bar{b}) \cdot (c + \bar{c}) + b \cdot \bar{c} \cdot (a + \bar{a}) \\ &= \bar{a} \cdot b \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c} + \bar{a} \cdot b \cdot \bar{c} \\ &= \bar{a} \cdot b \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c}\end{aligned}$$

DNF Theorem

DNF Theorem:

Every Boolean function can be expressed in DNF.

Proof: By the algorithm

Apply DeMorgan Theorem (T4) until each negation is applied to a single variable

Apply distributive axiom (A4) to get a sum of terms

Augment a missing variable a to a term using (A2) multiplying by $a + \bar{a}$

Use self absorption (L1) to eliminate duplicate terms

DNF Representation Theorem

DNF Representation Theorem:

DNF is a representation: two Boolean functions are equal if and only if their DNFs are identical.

Proof:

Easy direction: DNFs identical implies functions are equal

Other direction: different DNFs implies different functions

Idea: consider the value of the functions for the 0-1 assignments

• L2: Simple Absorption:

$$a + 1 = 1 \quad \text{and} \quad a \cdot 0 = 0$$

Other direction:

A1. Identities:

$$a + 0 = a \quad \text{and} \quad a \cdot 1 = a$$

two different DNFs implies two different functions

Proof: Idea: consider the value of the functions for the 0-1 assignments

$$f_1(a, b, c) = \bar{a} \cdot b \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c}$$

f = 1?

011

010

001

000

110

f = 0?

100

101

111

A normal term is 1 for a single 0-1 assignment and 0 otherwise

For 0-1 assignments either all terms are 0 or exactly one term is 1

Other direction:
two different DNFs implies two different functions

Proof: Idea: consider the value of the functions for the 0-1 assignments

$$f_1(a, b, c) = \bar{a} \cdot b \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c}$$

$$f_2(a, b, c) = \bar{a} \cdot b \cdot c + \bar{a} \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c}$$

$$f_1(1, 1, 0) = 1$$

$$f_2(1, 1, 0) = 0$$

If two DNFs are different, there is a term that appears in one and does not appear in the other

The 0-1 assignment that 'corresponds' to this term results in a 1 for one DNF and in a 0 for the other.

Counting Functions

DNF is a representation: two Boolean functions are equal if and only if their DNFs are identical.

Q1: How many different Boolean functions of n variables?

Answer: 2^{2^n}

This is the number of different DNFs of n variables

How is the **all-0** function represented?

How is the **all-1** function represented?

$$\bar{a} \cdot \bar{b} + \bar{a} \cdot b + a \cdot \bar{b} + a \cdot b = (a + \bar{a}) \cdot (b + \bar{b}) = 1$$

More 0-1 Ideas

Yes

Q2: Are Boolean functions determined by their 0-1 entries?

By the proof of the DNF Representation Theorem:
Two DNFs are identical if and only if they are equal for all 0-1 assignments

DNF is a representation: Two Boolean functions are equal if and only if their DNFs are identical

Hence:

Two Boolean functions are equal if and only if they are equal for all 0-1 assignments

DNF from Look Up Tables

Idea: construct a DNF by adding the normal terms that correspond to $f=1$.

ab	XOR(a,b)
00	0
01	1
10	1
11	0

$$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$$

More Elements

$\oplus$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\emptyset$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\{a\}$	$\{a\}$	$\emptyset$	$\{a, b\}$	$\{b\}$
$\{b\}$	$\{b\}$	$\{a, b\}$	$\emptyset$	$\{a\}$
$\{a, b\}$	$\{a, b\}$	$\{b\}$	$\{a\}$	$\emptyset$

$$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$$

The empty places are determined given the algebra and the 0-1 values

$\oplus$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\emptyset$	$\emptyset$			$\{a, b\}$
$\{a\}$				
$\{b\}$				
$\{a, b\}$	$\{a, b\}$			$\emptyset$

DNF from Look Up Tables

$\oplus$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\emptyset$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$
$\{a\}$	$\{a\}$	$\emptyset$	$\{a, b\}$	$\{b\}$
$\{b\}$	$\{b\}$	$\{a, b\}$	$\emptyset$	$\{a\}$
$\{a, b\}$	$\{a, b\}$	$\{b\}$	$\{a\}$	$\emptyset$

$$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$$

$$a_i^{x_i} = \begin{cases} a_i & \text{if } x_i = 1 \\ \bar{a}_i & \text{if } x_i = 0 \end{cases}$$

$$f(a_1, a_2, \dots, a_n) = \sum_{X \in \{0,1\}^n} f(x_1, x_2, \dots, x_n) a_1^{x_1} a_2^{x_2} \cdots a_n^{x_n}$$

Boole (Shannon) Decomposition

$$f(a_1, a_2, \dots, a_n) = \bar{a}_1 \cdot f(0, a_2, \dots, a_n) + a_1 \cdot f(1, a_2, \dots, a_n)$$

Proof:

DNF representation

$$f(a_1, a_2, \dots, a_n) = \sum_{X \in \{0,1\}^n} f(x_1, x_2, \dots, x_n) a_1^{x_1} a_2^{x_2} \cdots a_n^{x_n}$$

$\boxed{0^{x_1} = 1}$ $\boxed{\bar{a}_1}$

$$f(0, a_2, \dots, a_n) = \sum_{X \in \{0,1\}^n} f(0, x_2, \dots, x_n) a_2^{x_2} \cdots a_n^{x_n}$$

$\boxed{1^{x_1} = 1}$ $\boxed{a_1}$

$$f(1, a_2, \dots, a_n) = \sum_{X \in \{0,1\}^n} f(1, x_2, \dots, x_n) a_2^{x_2} \cdots a_n^{x_n}$$

Boole (Shannon) Decomposition

$$a \oplus b = \bar{a} \cdot b + a \cdot \bar{b}$$

ab	XOR(a,b)
00	0
01	1
10	1
11	0

Current State

Representation Theorem (Stone 1936):

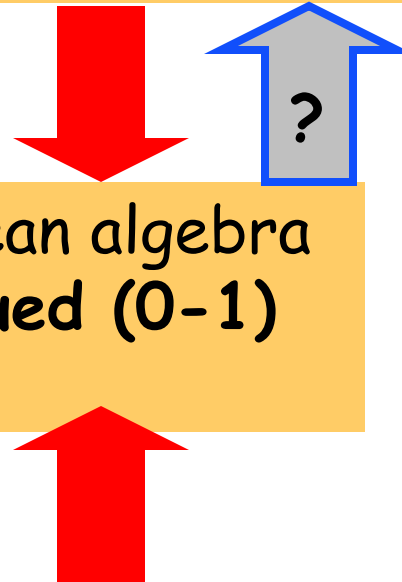
Every *finite* Boolean algebra is isomorphic to a Boolean algebra of subsets of some *finite* set S .

0-1 Theorem:

An identity is true for **any** finite Boolean algebra if and only if it is true for a **two-valued (0-1)** Boolean algebra.

DNF Representation Theorem:

DNF is a representation: two Boolean functions are equal if and only if their DNFs is identical.



"The further back you look,
the further forward you can see"
Winston Churchill

Gottfried Leibniz
1646-1716



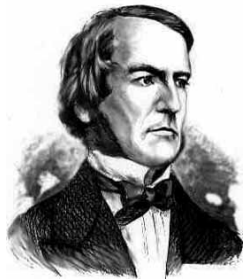
- Logic and Binary system
- Calculus

Shannon
1916-2001



Boolean Algebra to
Electrical Circuits

George Boole
1815-1864



Connected Logic
with Algebra

Edward Huntington
1874-1952



Boolean axiomatic
system

Marshal Stone
1903-1989



Boolean
representation
theorem

Gottfried Leibniz
1646-1716



Leibniz - Binary System

§71 This gives me the opportunity to point out that all numbers could be written by 0 and 1 in the binary or dual progression. Thus:

1
10
1000
10000
100000
1000000

1
2
8
16
32
64

10 is equal to 2
100 is equal to 4
1000 is equal to 8
etc.

Gottfried Leibniz
1646-1716



Leibniz - Binary System

And accordingly, numbers are expressed as follows:

0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

10000	16
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	31
100000	32
etc.	etc

Gottfried Leibniz
1646-1716



Leibniz - Binary System

These terms correspond with the hypothesis;
for example:

$$111 = 100 + 10 + 1 = 4 + 2 + 1 = 7$$

$$11001 = 10000 + 1000 + 1 = 16 + 4 + 1 = 21$$

8

Gottfried Leibniz
1646-1716



Leibniz - Binary System

Algorithm for generating the tables

§71a But, to continue, if one wishes to make a table expressing terms for all the natural numbers in order, one need not calculate, since it is sufficient to note that each column is periodic, the same periodicity recurring ad infinitum: the first column runs 0, 1, 0, 1, 0, 1, etc.; the second 0, 0, 1, 1, 0, 0, 1, 1, etc.; the third 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, etc.; the fourth 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, etc. And so on with further columns, assuming that the empty places above each column are filled with zeroes. Thus one can write these columns at once and accordingly make up the table of natural numbers without any calculations. This is what one can call enumeration.

Gottfried Leibniz
1646-1716



Leibniz - Binary System

Binary addition algorithm

§72 As for addition, it is simply done by counting and making periods when there are numbers to add together, adding up each column as usual, which will be done as follows: count the unities of the column for example, for 29, look how this number is written in the table, to wit, by 11101; thus you write 1 under the column and put periods under the second, third and fourth column thereafter. These periods denote that it is necessary to count out one unity further in the column following.

Gottfried Leibniz
1646-1716



§72 As for addition, it is simply done by counting and making periods when there are numbers to add together, adding up each column as usual, which will be done as follows: count the unities of the column; for example, for 29, look how this number is written in the table, to wit, by 11101; thus you write 1 under the column and put periods under the second, third and fourth column thereafter. These periods denote that it is necessary to count out one unity further in the column following.

§73 Subtraction is just as easy. Multiplication is reduced to simple additions and has no need of the Pythagorean table, it sufficing to know that 0 times 0 is 0, that 0 times 1 is 0, that 1 times 0 is 0, and that 1 times 1 is 1.

§74 In division there is no need to tally as in ordinary calculation. One must only see if the divisor is greater or less than the preceding remainder.

§75 These are simplifications that have been proposed by a clever man since the introduction of this Arithmetic into certain calculations.¹⁹¹ But the principal utility of this binary system is that it can do much to perfect the science of numbers because all calculations are made according to periodicity. It is some achievement that the numerical powers of the

¹⁹¹ Leibniz is referring to *Arithmetica perfectus, Qui Tria numerare nescit, seu Arithmetica dualis* (Prague, 1712) by W. J. Pelican. Pelican apparently showed how one can use the binary system for other calculations as well, i.e., fractional arithmetic, roots, and proportions. See Zacher, p. 211.

Gottfried Leibniz
1646-1716

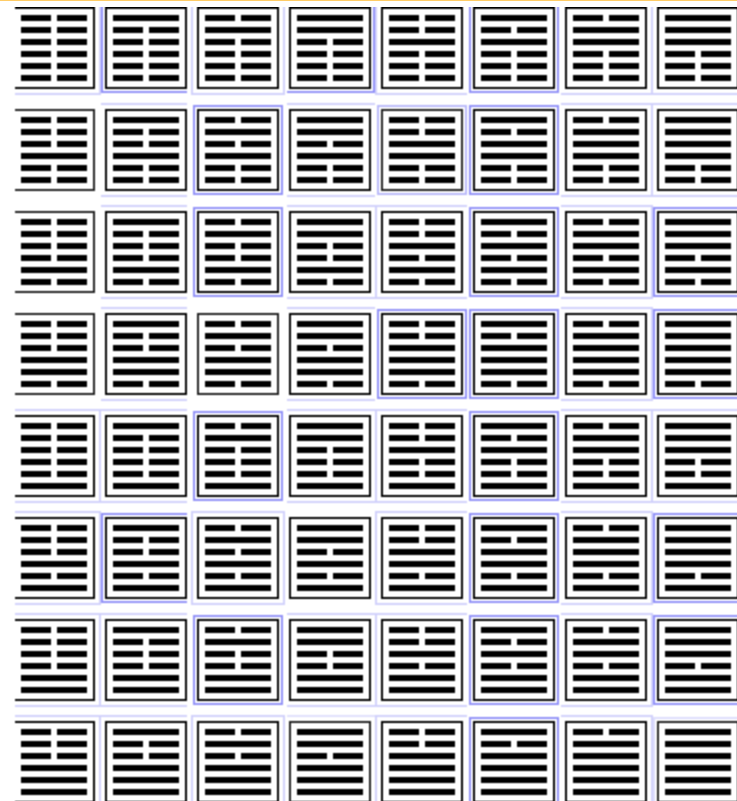


Leibniz - Binary System

Wen Wang (who flourished in about 1150 BC) is traditionally thought to have been author of the present hexagrams

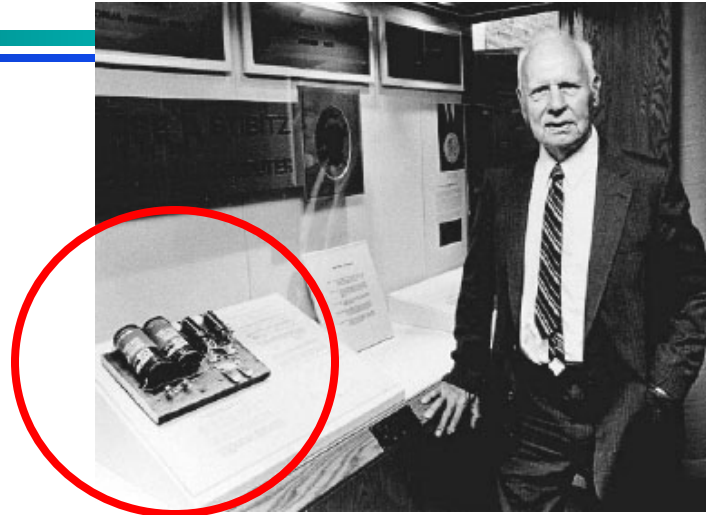


Figures of Fohi			
0	--	0	0
1	—	1	1
00	==	0	0
01	==	1	1
10	==	10	2
11	==	11	3
000	===	0	0
001	===	1	1
010	===	10	2
011	===	11	3
100	===	100	4
101	===	101	5
110	===	110	6
111	===	111	7



The First Digital Adder

George Stibitz, 1904-1995



He worked at Bell Labs in New York.

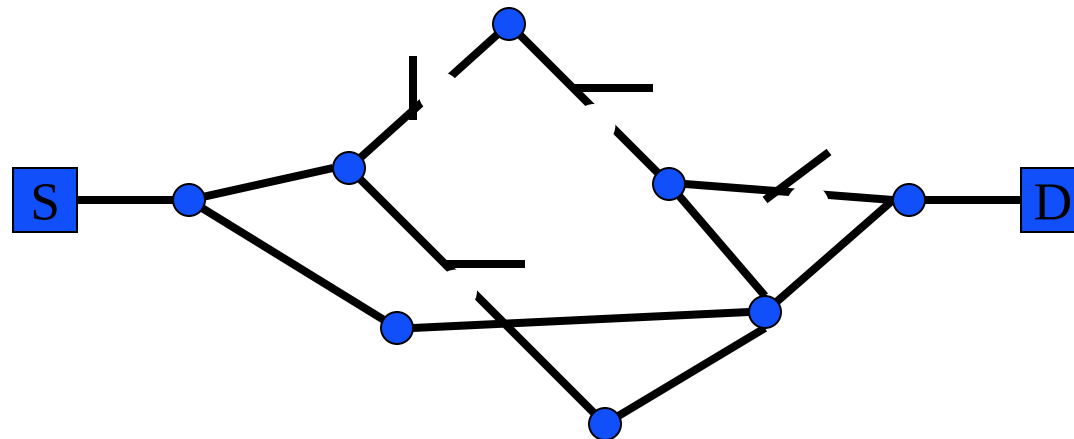
In the **fall of 1937** Stibitz used surplus relays, tin can strips, flashlight bulbs, and other common items to construct his "**Model K**" (K stands for kitchen table).

Model K was designed to display the result of the **addition of two bits**.

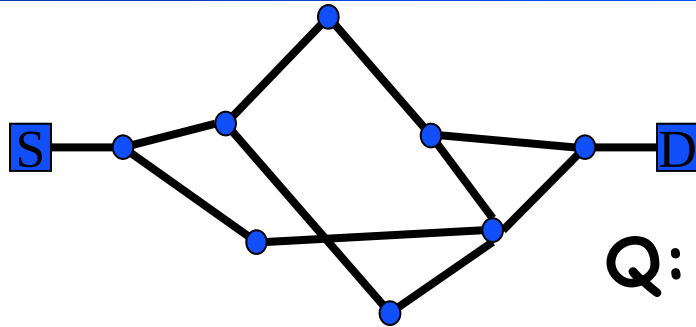
Connection Between Boolean Calculus and Physical Circuits Shannon 1938

The basic framework:

- A graph with edges and vertices
- Source and destination vertices (**S** and **D**)
- Boolean variables control the flow in the edges
- The function is **1** if **S** and **D** are **connected**
- The function is **0** if **S** and **D** are **disconnected**

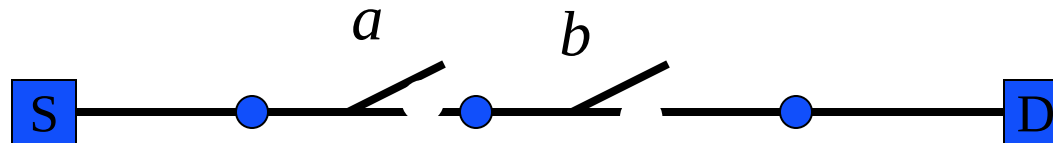


Connection Between Boolean Calculus and Physical Circuits Shannon 1938

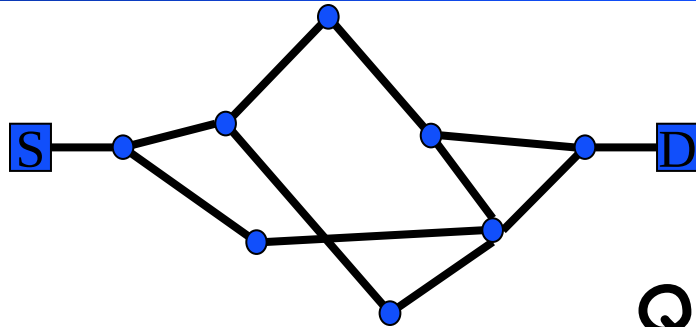


Q: How to design an AND gate?

$$a \cdot b$$

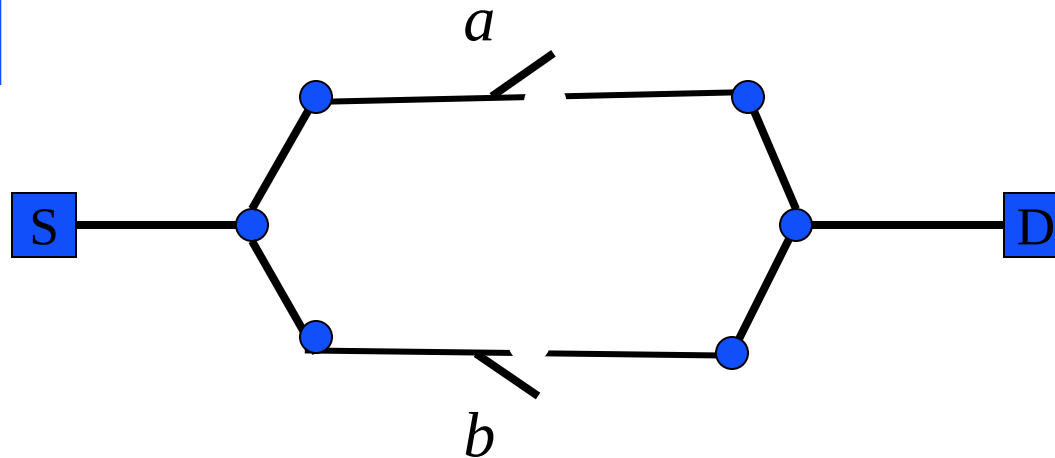


Connection Between Boolean Calculus and Physical Circuits Shannon 1938

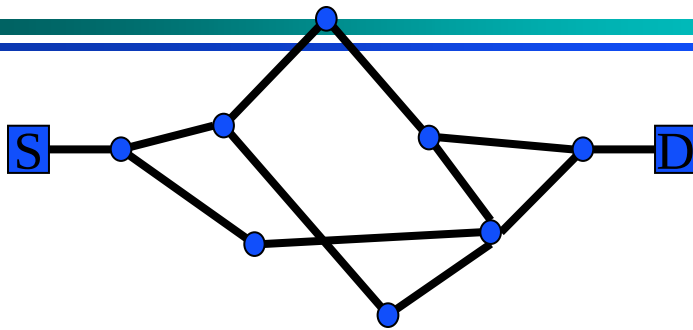


Q: How to design an OR gate?

$$a + b$$



Connection Between Boolean Calculus and Physical Circuits Shannon 1938



Shannon's paper:

he preferred to have 0 = connected and 1 = disconnected



Figure 2 (right). Interpretation of addition

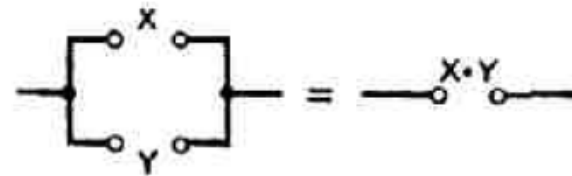


Figure 3 (middle). Interpretation of multiplication

Connection Between Boolean Calculus and Physical Circuits Shannon 1938

We shall limit our treatment of circuits containing only relay contacts and switches, and therefore at any given time the circuit between any two terminals must be either open (infinite impedance) or closed (zero impedance). Let us associate a symbol X_{ab} or more simply X , with the terminals a and b . This variable, a function of time, will be called the hindrance of the two-terminal circuit $a - b$. The symbol 0 (zero) will be used to represent the hindrance of a closed circuit, and the symbol 1 (unity) to represent the hindrance of an open circuit. Thus when the circuit $a - b$ is open $X_{ab} = 1$ and when closed $X_{ab} = 0$. Two hindrances X_{ab} and



Figure 2 (right). Interpretation of addition

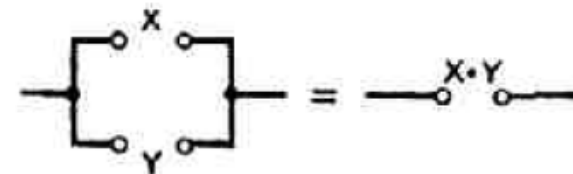
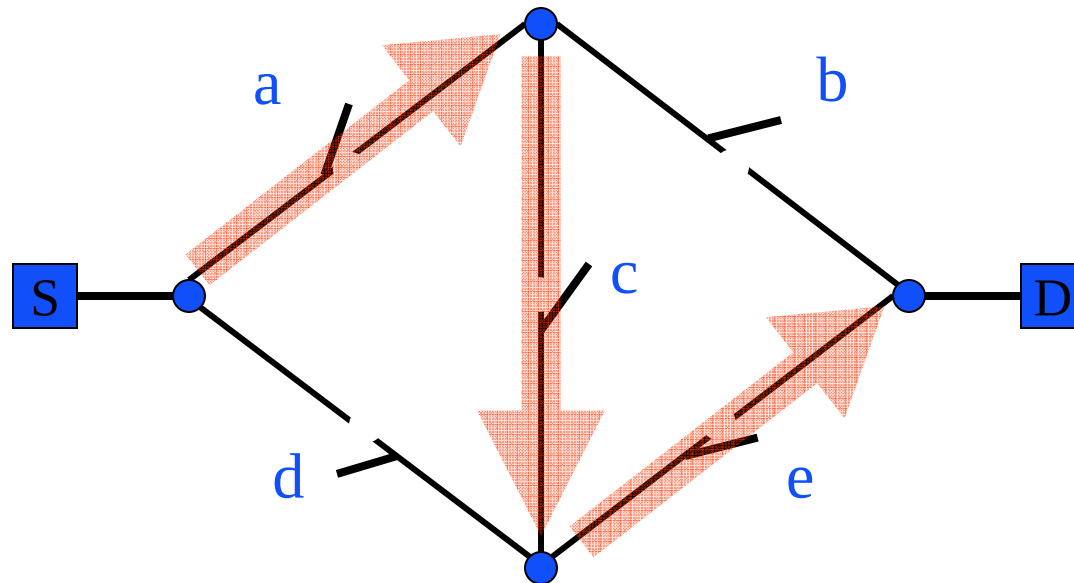


Figure 3 (middle). Interpretation of multiplication

Connection Between Boolean Calculus and Physical Circuits Shannon 1938

Q: relay circuits to Boolean calculus?

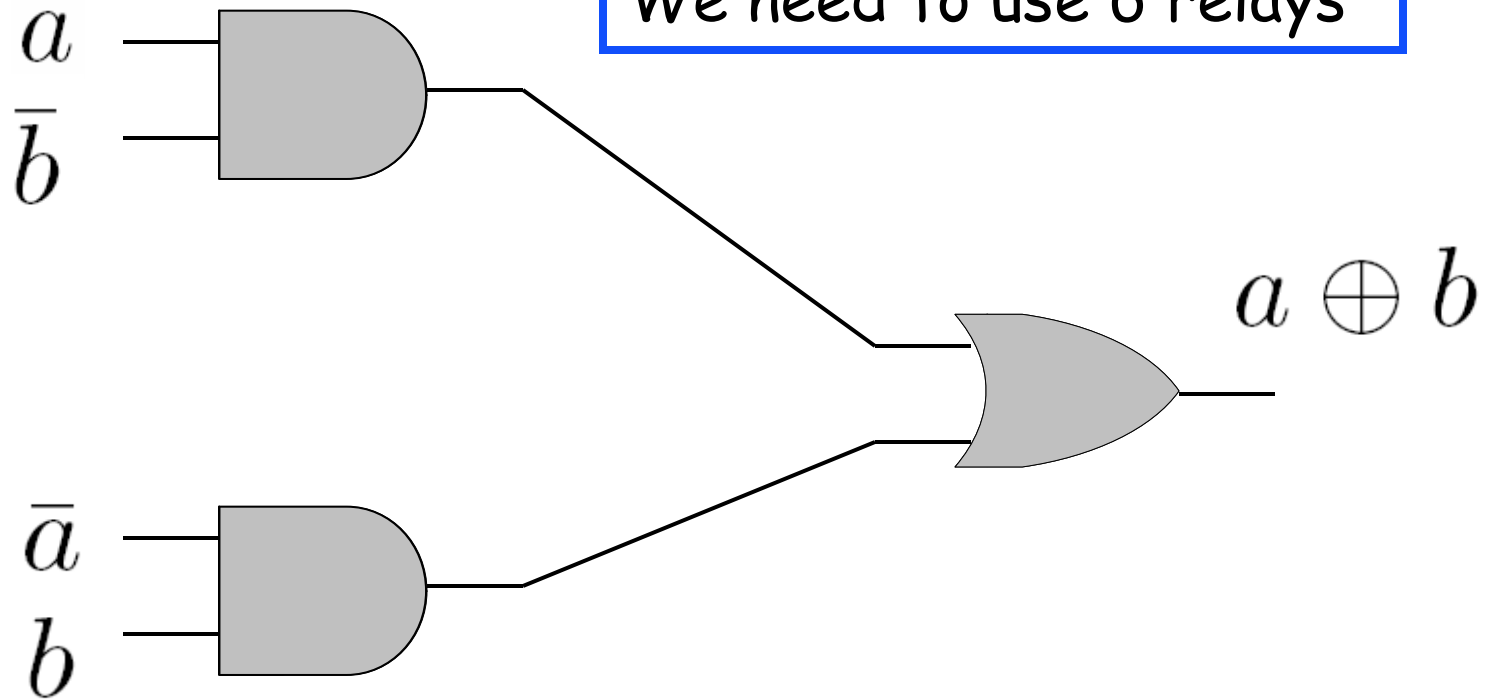


$$f = ab + de + ace + dcb$$

XOR of TWO Variables with Relays?

$$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$$

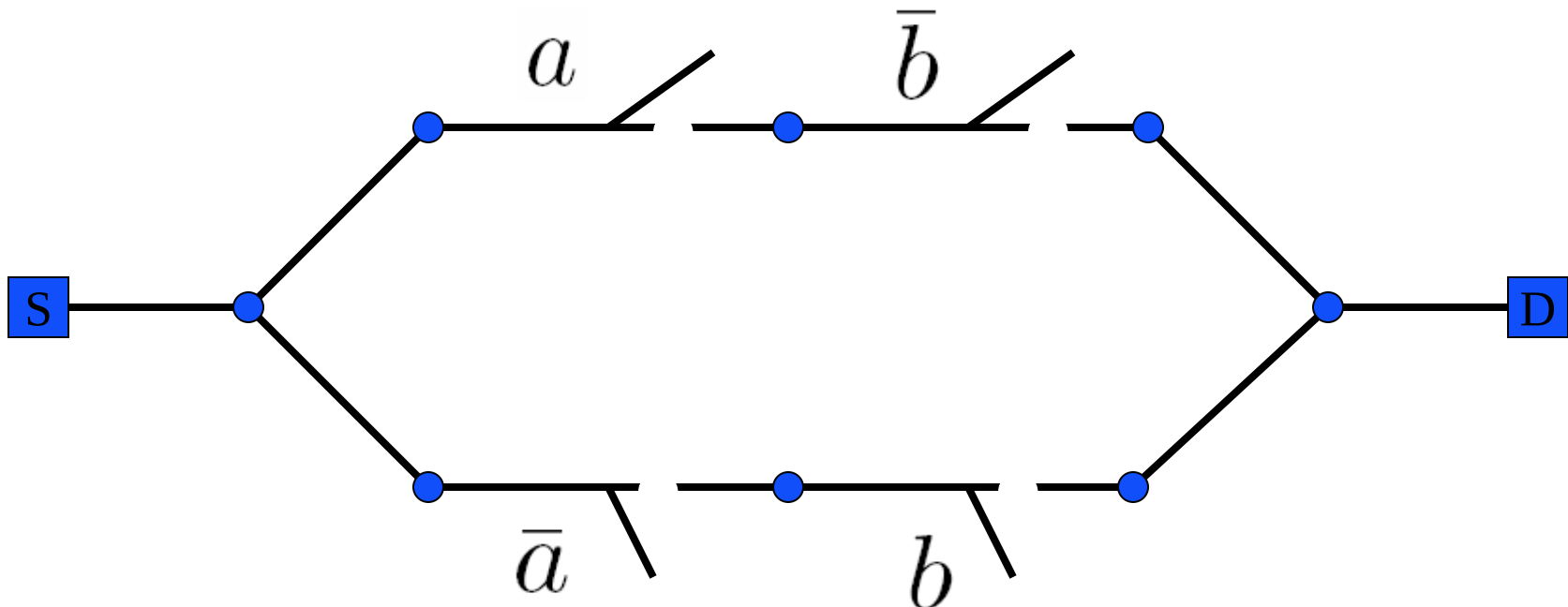
We need to use 6 relays



XOR of TWO Variables with Relays?

$$a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$$

Q: Can we compute XOR with less than 6 relays?



XOR of More Variables

$$XOR(x_1, x_2, \dots, x_n) = \begin{cases} 0 & \text{if } |X| \text{ (number of 1's in } X) \text{ is even} \\ 1 & \text{if } |X| \text{ is odd} \end{cases}$$

$$a \oplus b \oplus c = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot c$$

abc	XOR(a,b,c)
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

Every 0-1 Boolean function
can be implemented by a depth-2
AON circuit (using the DNF).

$$\mathbf{AON} = \mathbf{A}_{\text{ND}} \mathbf{O}_{\text{R}} \mathbf{N}_{\text{OT}}$$