C335
Computer Structures

Basics of Logic Design

Dr. Liqiang Zhang

Department of Computer and Information Sciences

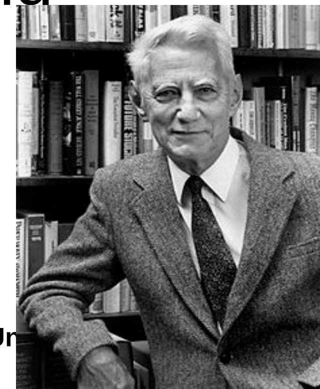
Adapted from Morgan Kaufmann and others

Boolean Algebra

- Named after mid-19th mathematician *George Boole*
- Two states: True (1) or False (0)
- Three operations
  - Binary operations:
    - AND (•), OR (+)
  - Unary operation
    - NOT (¯)

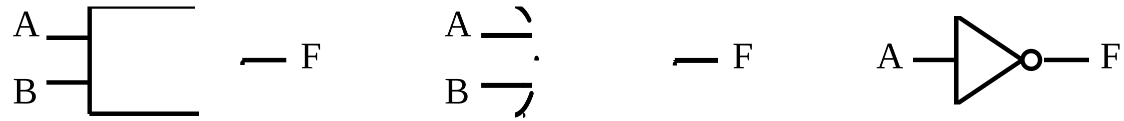
AND (•)	OR (+)	NOT (¯)
0 • 0 = 0	0 + 0 = 0	$\overline{0} = 1$
0 • 1 = 0	0 + 1 = 1	$\overline{1} = 0$
1 • 0 = 0	1 + 0 = 1	
1 • 1 = 1	1 + 1 = 1	

- This algebra was applied to electronic switching circuits by *Claude Shannon*, as a "Switching algebra"



Gates, Logic Equations, and Truth Table

- Gates



AND OR NOT

- Logic Equations

$F = A \bullet B$ $F = A + B$ $F = \overline{A}$ or (A')

- Truth Table

A B	F = A • B
0 0	0
0 1	0
1 0	0
1 1	1

A B	F = A + B
0 0	0
0 1	1
1 0	1
1 1	1

A	F = A'
0	1
1	0

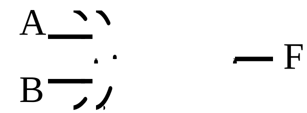
  - Tabular listing that fully describes a Logic Equation
  - Output value for all input combinations (valuations)

Gates, Logic Equations, and Truth Table

- Gates

NAND

NOR



XOR

- Logic Equations

$F = (A \cdot B)'$ $\quad$ $F = (A + B)'$ $\quad$ $F = A \oplus B$

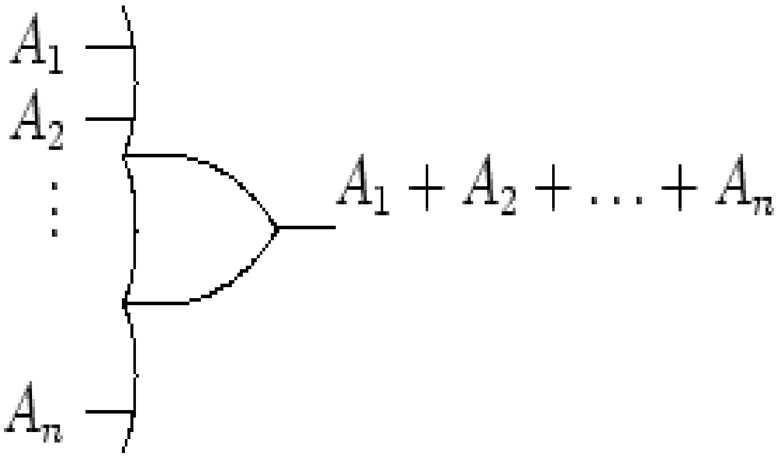
- Truth Table

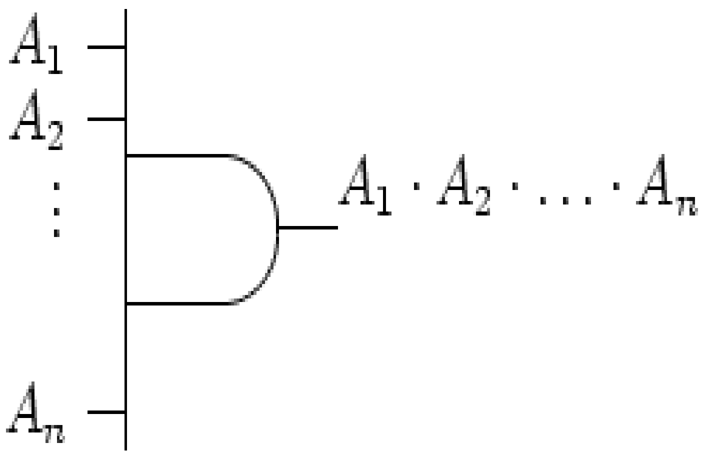
A	B	$F = (A \bullet B)'$
0	0	1
0	1	1
1	0	1
1	1	0

A	B	$F = (A + B)'$
0	0	1
0	1	0
1	0	0
1	1	0

A	B	$F = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Extends to N-input Logic Gates





Boolean Algebra: Notations

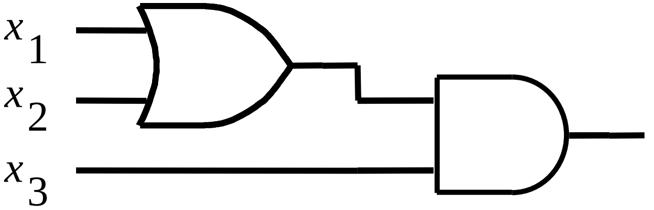
- A AND B → A•B, AB, or A&B (also called *product*)

- A OR B → A + B, A | B (also called *sum*)

- NOT A → A', $\overline{A}$, or !A (also called *Complementation* or *Inversion*)

Logic gates and networks

- A larger circuit is implemented by a network of gates
  - Called a logic network or logic circuit

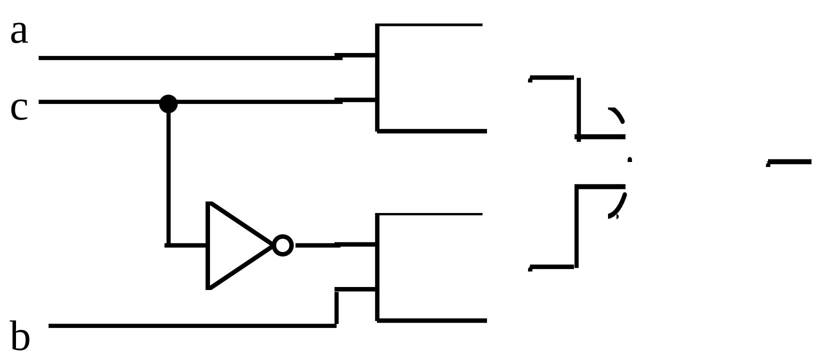


$f = (x_1 + x_2)\ x_3$

Logic gates and networks

- Draw the truth table and the logic circuit for the following logic equation
  - F = a•c+b•c’

a	b	c	ac	bc'	ac+bc'
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	1	0	1



Properties of Boolean Algebra (1/4)

- Axioms of Boolean Algebra
  - 1a $0 \cdot 0 = 0$
  - 1b $1 + 1 = 1$
  - 2a $1 \cdot 1 = 1$
  - 2b $0 + 0 = 0$
  - 3a $0 \cdot 1 = 1 \cdot 0 = 0$
  - 3b $1 + 0 = 0 + 1 = 1$
  - 4a If $x = 0$ then $x' = 1$
  - 4b If $x = 1$ then $x' = 0$

Properties of Boolean Algebra (2/4)

- Single-Variable Properties
  - 5a $x \cdot 0 =$ *Identity law*
  - *5b* $x + 1 =$
  - *6a* $x \cdot 1 =$ *Zero and one laws*
  - *6b* $x + 0 =$
  - *7a* $x \cdot x =$ *Idempotent law*
  - *7b* $x + x =$
  - *8a* $x \cdot x' =$ *Inverse law*
  - *8b* $x + x' =$
  - *9* $x'' =$
- Substitute the values x=0 and x=1 into the expressions and verify using the basic axioms

Properties of Boolean Algebra (2/4)

- Single-Variable Properties
  - 5a $x \cdot 0=0$ *Identity law*
  - *5b* $x+1=1$
  - *6a* $x \cdot 1=x$ *Zero and one laws*
  - *6b* $x+0=x$
  - *7a* $x \cdot x=x$ *Idempotent law*
  - *7b* $x+x=x$
  - *8a* $x \cdot x'=0$ *Inverse law*
  - *8b* $x+x'=1$
  - *9* $x''=x$
- Substitute the values x=0 and x=1 into the expressions and verify using the basic axioms

Properties of Boolean Algebra (3/4)

- Two & Three Variable Properties
  - 10a. $x \cdot y = y \cdot x$ *Commutative law*
  - 10b. $x + y = y + x$
  - 11a. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ *Associative law*
  - 11b. $x + (y + z) = (x + y) + z$
  - 12a. $x \cdot (y + z) = x \cdot y + x \cdot z$ *Distributive law*
  - 12b. $x + y \cdot z = (x + y) \cdot (x + z)$
  - 13a. $x + x \cdot y = x$ *Absorption law*
  - 13b. $x \cdot (x + y) = x$

Properties of Boolean Algebra (4/4)

- Two & Three Variable Properties
  - 14a. $x \cdot y + x \cdot y' = x$ *Combining law*
  - 14b. $(x+y) \cdot (x+y') = x$
  - 15a. $(x \cdot y)' = x' + y'$ *De Morgan's*
  - 15b. $(x+y)' = x' \cdot y'$ *Theorem*
  - 16a. $x + x' \cdot y = x + y$
  - 16b. $x \cdot (x' + y) = x \cdot y$

Induction proof of $x+x'\cdot y=x+y$

- Use perfect induction to prove $x+x'\cdot y=x+y$

x	y	$x'y$	$x+x'y$	$x+y$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

equivalent

Perfect induction example

- Use perfect induction to prove $(xy)' = x' + y'$

x	y	xy	$(xy)'$	x'	y'	$x'+y'$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

equivalent

Canonical Forms of Logic Equations

- It is possible to construct a truth table for any Logic Equation
  - The truth table is useful in that it does provide a complete, unique description of the Logic Equation, but is cumbersome
- We can derive from the truth table certain unique expressions which defines the function exactly
  - Minterm form (sum of products)
  - Maxterm from (product of sums)

Minterm Form

A	B	$F = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

$$F = A' \bullet B + A \bullet B'$$

Sum of Products form

- It is obtained by ANDing together the variables, or their complements, which have a 1 in the function column
- If the variable has value 1, the variable is taken; if not, its complement is taken
- Each of these ANDs of all the variables, or their complements, is called a minterm.
- The minterms are then ORed together to give the function specified in the truth table.

Maxterm Form

- *A Dual form of minterm form* is called *maxterm* form, or *product of sums* form.

$$
\begin{aligned}
F &= A' \cdot B + A \cdot B' \\
F' &= (A' \cdot B + A \cdot B')' \\
&= (A' \cdot B)' \cdot (A \cdot B')' && \text{//using De Morgan's Theorem} \\
&= (A + B') \cdot (A' + B) && \text{//using De Morgan's Theorem} \\
&= A \cdot A' + A \cdot B + B' \cdot A' + B' \cdot B && \text{//using Distributive Law} \\
&= A \cdot B + A' \cdot B' \\
F = (F')' &= (A \cdot B + A' \cdot B')' \\
&= (A \cdot B)' \cdot (A' \cdot B')' && \text{//using De Morgan's Theorem} \\
&= (A' + B') \cdot (A + B) && \text{//using De Morgan's Theorem}
\end{aligned}
$$

- It can be obtained easily from the truth table by ORing together all the variables or their complements which give a zero for the function.
- If the variable has value 0, the variable is taken; if not, its complement is taken

Minterm Form vs. Maxterm Form

	A	B	C	F	Minterms	Maxterms
0	0	0	0	1	A’ • B’ • C’	
1	0	0	1	0		A + B + C’
2	0	1	0	1	A’ • B • C’	
3	0	1	1	1	A’ • B • C	
4	1	0	0	0		A’ + B + C
5	1	0	1	0		A’ + B + C’
6	1	1	0	1	A • B • C’	
7	1	1	1	1	A • B • C	

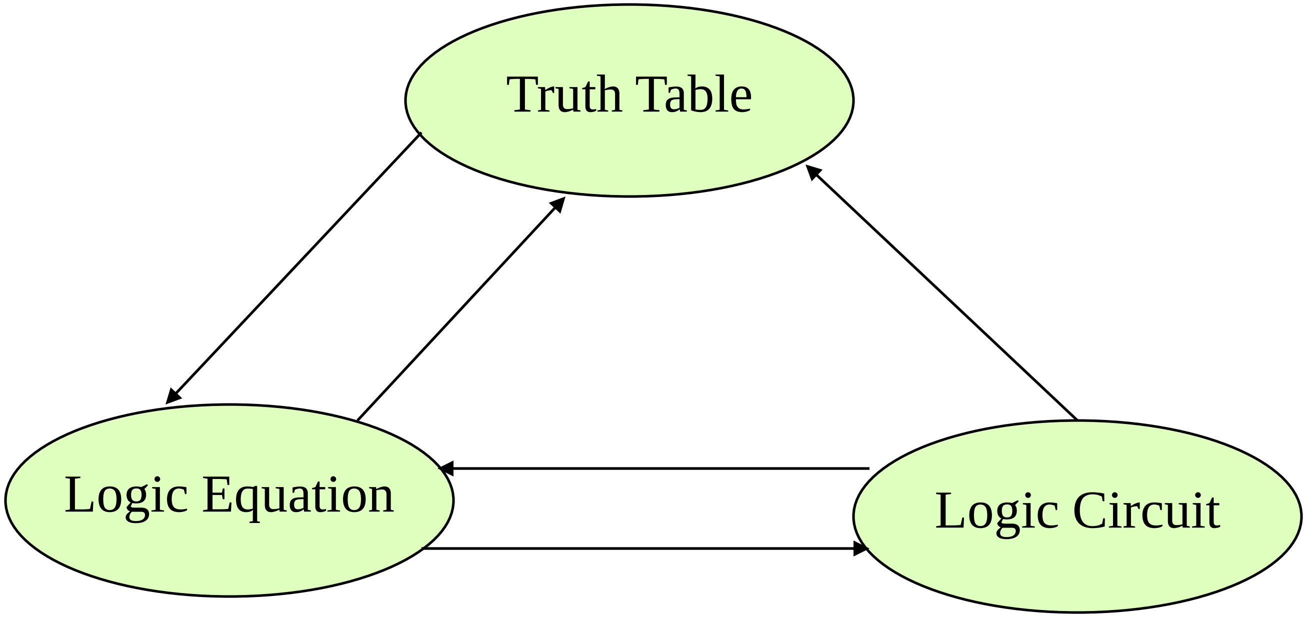
Minterm form:

F = A’ • B’ • C’ + A’ • B • C’ + A’ • B • C + A • B • C’ + A • B • C

Maxterm form:

F = (A + B + C’) • (A’ + B + C) • (A’ + B + C’)

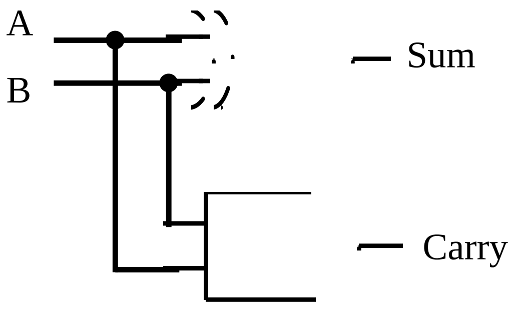
Logic Equations, Truth Table, and Logic Circuits

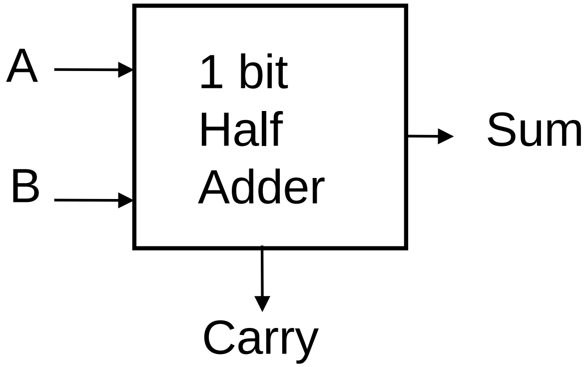


Design of a Half Adder

- **1 bit half adder**: a switching circuit which add together two binary digits (bits), producing two output bits, a sum bit, and a carry bit .

Input		Output	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

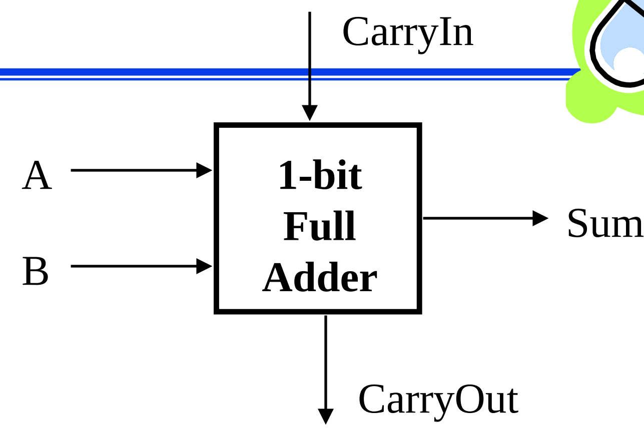




- Carry = A • B
- Sum = A'• B + A • B' = A ⊕ B

A One-bit Full Adder

- 1 bit full adder: a switching circuit which add together two binary digits (bits), and a third bit called a *CarryIn* bit which may have come from a previous full adder.



Inputs			Outputs		
A	**B**	**CarryIn**	**CarryOut**	**Sum**	**Comments**
0	0	0	0	0	0 + 0 + 0 = 00
0	0	1	0	1	0 + 0 + 1 = 01
0	1	0	0	1	0 + 1 + 0 = 01
0	1	1	1	0	0 + 1 + 1 = 10
1	0	0	0	1	1 + 0 + 0 = 01
1	0	1	1	0	1 + 0 + 1 = 10
1	1	0	1	0	1 + 1 + 0 = 10
1	1	1	1	1	1 + 1 + 1 = 11

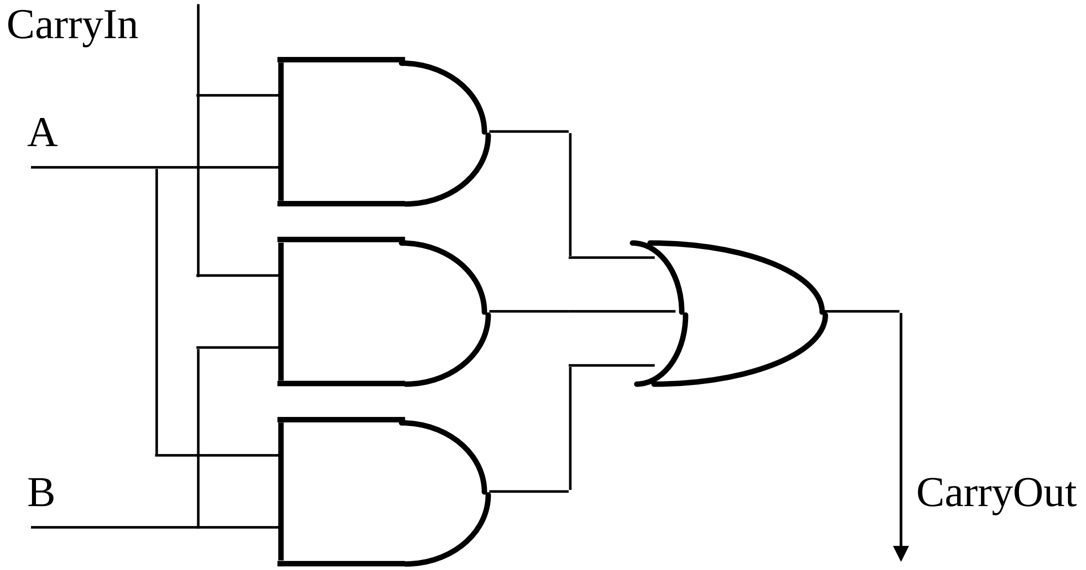
Logic Equation for CarryOut

Inputs			Outputs		Comments
A	**B**	**CarryIn**	**CarryOut**	**Sum**	**Comments**
0	0	0	0	0	0 + 0 + 0 = 00
0	0	1	0	1	0 + 0 + 1 = 01
0	1	0	0	1	0 + 1 + 0 = 01
0	1	1	1	0	0 + 1 + 1 = 10
1	0	0	0	1	1 + 0 + 0 = 01
1	0	1	1	0	1 + 0 + 1 = 10
1	1	0	1	0	1 + 1 + 0 = 10
1	1	1	1	1	1 + 1 + 1 = 11

- CarryOut = (A' • B • CarryIn) + (A • B' • CarryIn) + (A • B • CarryIn') + (A • B • CarryIn)
- CarryOut = B • CarryIn + A • CarryIn + A • B (*majority function*)

Logic Circuit for CarryOut

- CarryOut = B • CarryIn + A • CarryIn + A • B



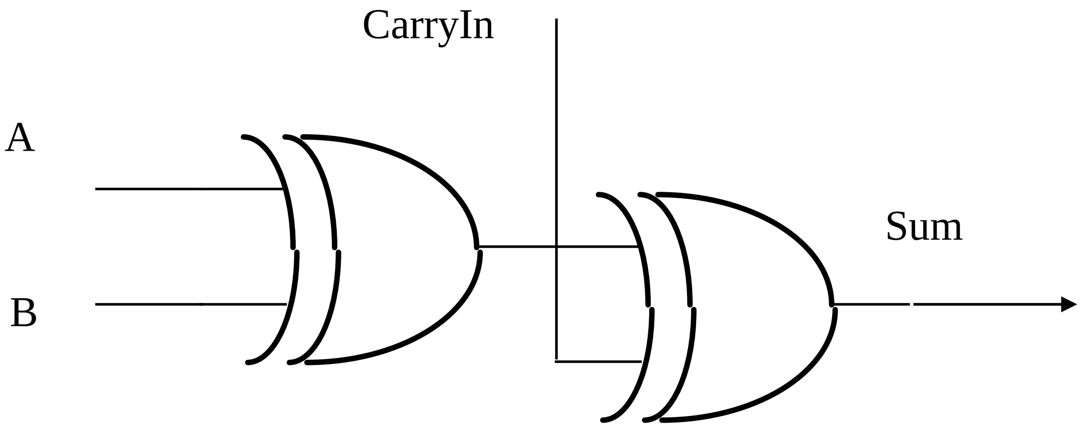
Logic Equation for Sum

Inputs			Outputs		Comments
A	B	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	0 + 0 + 0 = 00
0	0	1	0	1	0 + 0 + 1 = 01
0	1	0	0	1	0 + 1 + 0 = 01
0	1	1	1	0	0 + 1 + 1 = 10
1	0	0	0	1	1 + 0 + 0 = 01
1	0	1	1	0	1 + 0 + 1 = 10
1	1	0	1	0	1 + 1 + 0 = 10
1	1	1	1	1	1 + 1 + 1 = 11

- Sum = (A' • B' • CarryIn) + (A' • B • CarryIn') + (A • B' • CarryIn') + (A • B • CarryIn)
- Sum = A ⊕ B ⊕ CarryIn

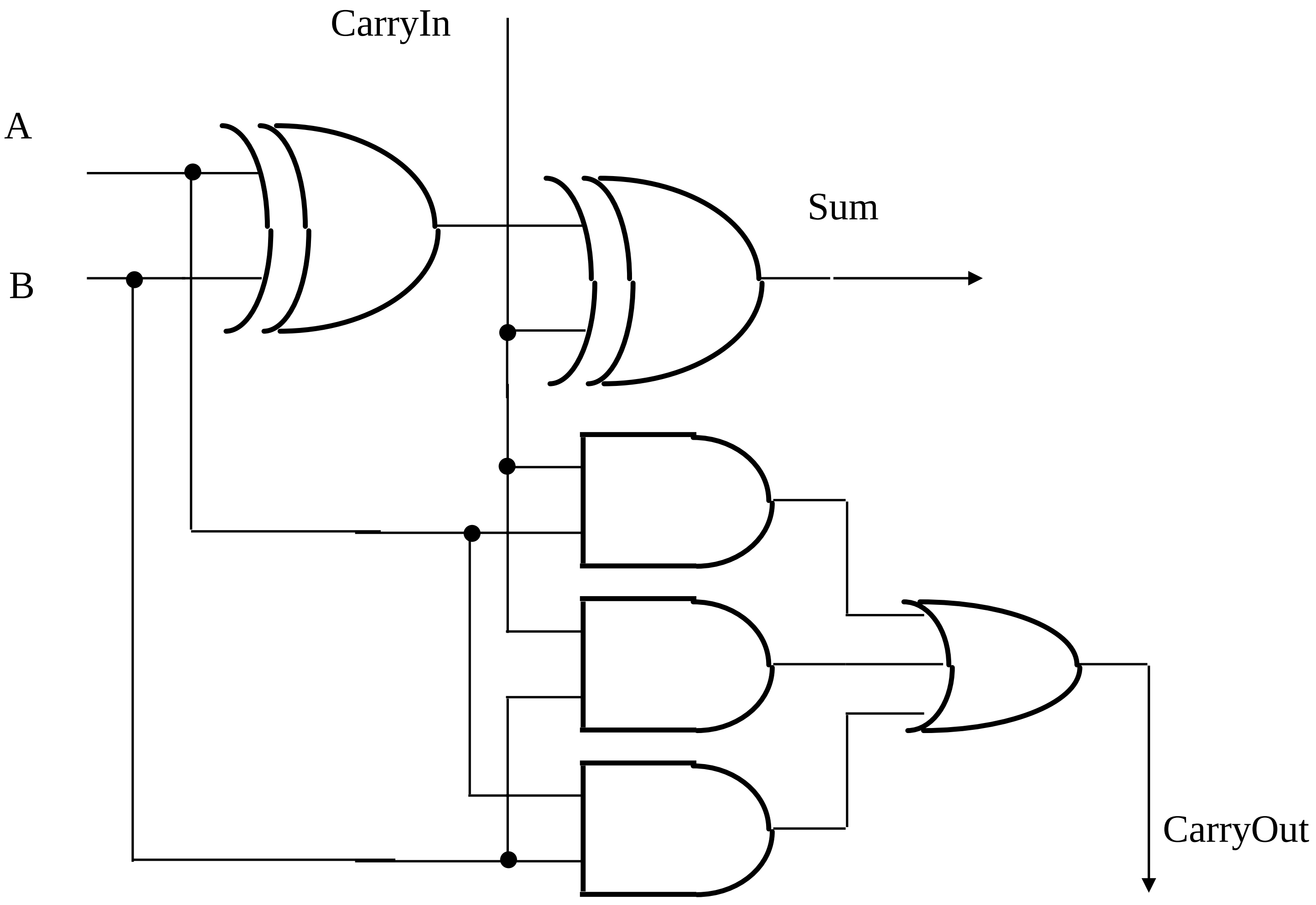
Logic Circuit for Sum

- Sum = A ⊕ B ⊕ CarryIn



Put Together

- Sum = A ⊕ B ⊕ CarryIn
- CarryOut = B • CarryIn + A • CarryIn + A • B

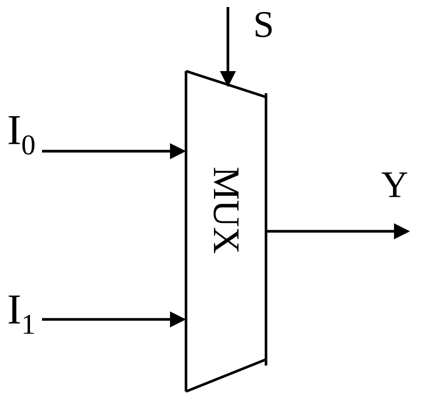


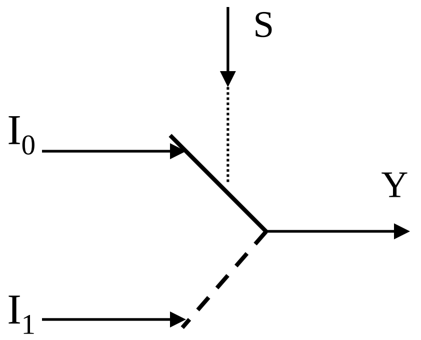
Design of a 2-line to 1-line Multiplexer

- Multiplexer: the output is selected to be equal to the input of the line given by the binary value of the select line(s).

S	Y
0	I_0
1	I_1

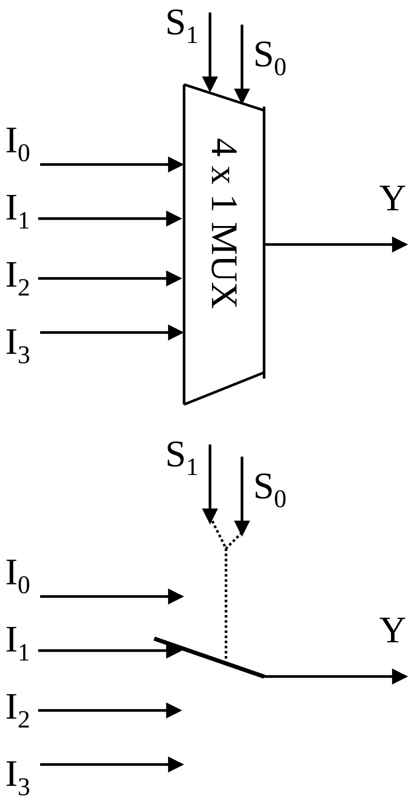
S	I_0	I_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1





- $Y = S' \bullet I_0 \bullet I_1' + S' \bullet I_0 \bullet I_1 + S \bullet I_0' \bullet I_1 + S \bullet I_0 \bullet I_1$
- $Y = S' \bullet I_0 + S \bullet I_1$

A 4-line to 1-line Multiplexer

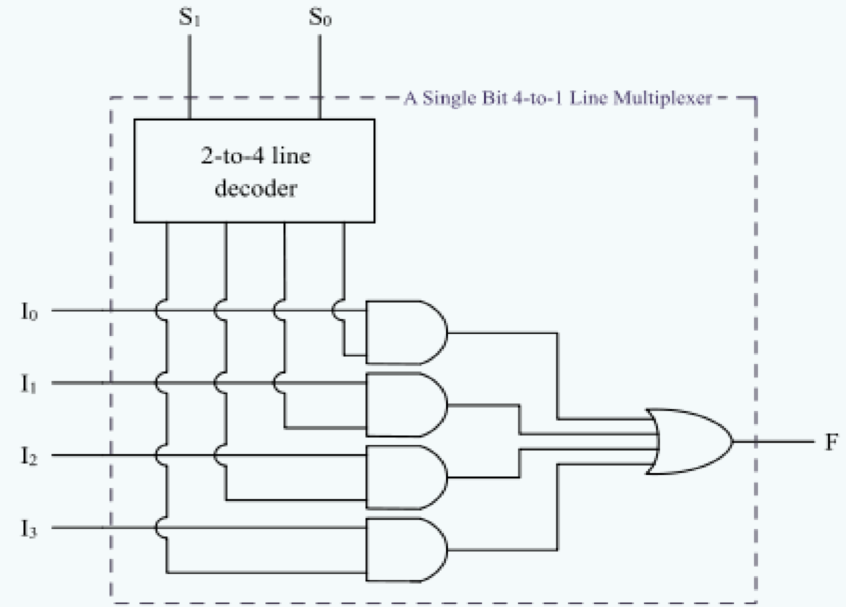


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

- For a 8-line to 1-line multiplexer, how many select lines do we need?
- For a 16-line to 1-line multiplexer, how many select lines do we need?

If you are interested…

A 4-line to 1-line Multiplexer



Truth Table

S_1	S_0	I_3	I_2	I_1	I_0	F
0	0	0	0	0	0	0
		0	0	0	1	1
		0	0	1	0	0
		0	0	1	1	1
		0	1	0	0	0
		0	1	0	1	1
		0	1	1	0	0
		0	1	1	1	1
		1	0	0	0	0
		1	0	0	1	1
		1	0	1	0	0
		1	0	1	1	1
		1	1	0	0	0
		1	1	0	1	1
		1	1	1	0	0
		1	1	1	1	1

S_1	S_0	I_3	I_2	I_1	I_0	F
0	1	0	0	0	0	0
		0	0	0	1	0
		0	0	1	0	1
		0	0	1	1	1
		0	1	0	0	0
		0	1	0	1	0
		0	1	1	0	1
		0	1	1	1	1
		1	0	0	0	0
		1	0	0	1	0
		1	0	1	0	1
		1	0	1	1	1
		1	1	0	0	0
		1	1	0	1	0
		1	1	1	0	1
		1	1	1	1	1

S_1	S_0	I_3	I_2	I_1	I_0	F
1	0	0	0	0	0	0
		0	0	0	1	0
		0	0	1	0	0
		0	0	1	1	0
		0	1	0	0	1
		0	1	0	1	1
		0	1	1	0	1
		0	1	1	1	1
		1	0	0	0	0
		1	0	0	1	0
		1	0	1	0	0
		1	0	1	1	0
		1	1	0	0	1
		1	1	0	1	1
		1	1	1	0	1
		1	1	1	1	1

S_1	S_0	I_3	I_2	I_1	I_0	F
1	1	0	0	0	0	0
		0	0	0	1	0
		0	0	1	0	0
		0	0	1	1	0
		0	1	0	0	0
		0	1	0	1	0
		0	1	1	0	0
		0	1	1	1	0
		1	0	0	0	1
		1	0	0	1	1
		1	0	1	0	1
		1	0	1	1	1
		1	1	0	0	1
		1	1	0	1	1
		1	1	1	0	1
		1	1	1	1	1

If you are interested…

- What is a decoder? (1-to-2, 2-to-4, 3-to-8…)
- How do you utilize a decoder to construct a multiplexer?

Summary

- Boolean Algebra:
  - two values, three operations
- Properties of Boolean Algebra
  - Axioms
  - Single-variable properties
  - Two & three-variable properties
- Relations among logic equations, truth tables, and logic circuits
- Design of a half adder and a full adder
  - Truth table → logic equation → logic circuit
- Multiplexer