



University of
Massachusetts
Amherst

ECE124
Intro-Digital and Computer Sys

Chapter 2

Boolean Algebra and Logic Gates

Algebras

- What is an algebra?
 - Mathematical system consisting of
 - » Set of elements
 - » Set of operators
 - » Axioms or postulates
- Why is it important?
 - Defines rules of “calculations”
- Example: arithmetic on natural numbers
 - Set of elements: $N = \{1, 2, 3, 4, \dots\}$
 - Operator: $+$, $-$, \bullet (or \times)
 - Axioms: associativity, distributivity, closure, identity elements, etc.
- Note: operators with two inputs are called binary
 - Relate to two objects
 - Does not mean they are restricted to binary numbers!
- Operator(s) with one input are called unary

Axioms of Algebraic Structures

- Common axioms (or postulates):

- 1. Closure:**

- » A set S is closed with respect to a binary operator $*$ if the operator specifies a rule for obtaining an element of S .
 - » Example:

- 2. Associativity:**

- » A binary operator $*$ on a set S is said to be associative if
$$(x * y) * z = x * (y * z)$$
 - » Example:

- 3. Commutativity:**

- » A binary operator $*$ on a set S is said to be commutative if
$$x * y = y * x$$
 - » Example:

Axioms of Algebraic Structures

4. Identity element:

- » Set S is said to have an identity element with respect to binary operation $*$ on S if there exists an element $e \in S$ such that

$$e * x = x * e = x \quad \text{for every } x \in S$$

- » Example:

5. Inverse:

- » A set S having the identity element e with respect to a binary operator $*$ is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that $x * y = e$

- » Example:

6. Distributivity:

- » If $*$ and $\#$ are two binary operators on a set S , $*$ is said to be distributive over $\#$ whenever $x * (y \# z) = (x * y) \# (x * z)$

- » Example:

Huntington's Postulates

- In 1904, E.V. Huntington defined Boolean Algebra by providing 6 postulates that must be satisfied, called Huntington's Postulates
 1. Closure with respect to the operators (operator + and operator \bullet):
Any logical operations yields a value in the set $\{0,1\}$
 2. Identity elements with respect to the operators
$$x + 0 = x, \quad x \bullet 1 = x$$
 3. Commutativity with respect to + and \bullet
$$x+y = y+x, \quad x \cdot y = y \cdot x$$
 4. Distributivity of \bullet over +, and + over \bullet
$$x \cdot (y+z) = (x \cdot y) + (x \cdot z) \quad \text{and} \quad x + (y \cdot z) = (x+y) \cdot (x+z)$$
 5. Complements exist for all the elements
$$x+x'=1, \quad x \cdot x'=0$$
 6. Existence of the complement
$$0 = \bar{1}, \quad 1 = \bar{0},$$

Two-valued Boolean Algebra

In 1938, Claude Shannon showed that a two-valued Boolean Algebra which he called switching algebra, could be used to describe digital circuits:

- Two elements $\{0,1\}$
- Two binary operator $+(OR)$, $\cdot(AND)$
- One unary Operator, the complement'

“+” operator:

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

“•” operator:

x	y	$x \bullet y$
0	0	
0	1	
1	0	
1	1	

complement ‘

x	x'
0	
1	

- Observation

- $+$ is *OR*, \bullet is *AND*, $'$ is *NOT* Other notation: $+$ = \vee , \bullet = \wedge , *NOT* = \neg

Check Huntington's Postulates

- Verify that postulates hold for Boolean algebra

1. Closure?

» Yes, both operations produce 0 or 1 for all input combinations.

2. Identity?

» Yes, 0 for + and 1 for •

3. Commutativity?

» Yes, for +: $0+0=0$, $1+1=1$, $0+1=1+0=1$

» Yes, for •: $0\cdot 0=0$, $1\cdot 1=1$, $0\cdot 1=1\cdot 0=0$

4. Distributivity?

» Yes, use truth table (p. 46 in Mano) for • over + and + over •

5. Complement?

» Yes, $x+x'=1$: $0+0'=0+1=1$ and $1+1'=1+0=1$

» Yes, $x\cdot x'=0$: $0\cdot 0'=0\cdot 1=0$ and $1\cdot 1'=1\cdot 0=0$

6. Two distinct values?

» Yes, $0\neq 1$

Huntington postulates:

Post. 1: closure

Post. 2: (a) $x+0=x$, (b) $x\cdot 1=x$

Post. 3: (a) $x+y=y+x$, (b) $x\cdot y=y\cdot x$

Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$

Post. 5: (a) $x+x'=1$, (b) $x\cdot x'=0$

x	y	x+y	x	y	x • y
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	1

Huntington postulates:

Post. 1: closure

Post. 2: (a) $x+0=x$, (b) $x\cdot 1=x$

Post. 3: (a) $x+y=y+x$, (b) $x\cdot y=y\cdot x$

Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$

Post. 5: (a) $x+x'=1$, (b) $x\cdot x'=0$

Boolean Functions

- Boolean elements and operators can express a function
 - Value of function determined by values of variables
 - Complete function defined for all possible values of variables
- Boolean function can be represented in a standard form as a Sum of Products (SOP) or by a truth table

- E.g., $F = x + y'z$

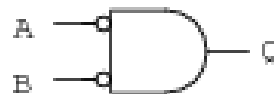
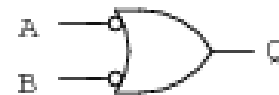
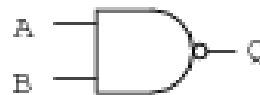
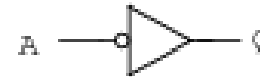
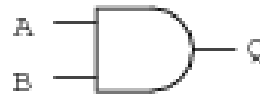
Can you think of an application that is modeled by F ?

- Evaluate in steps
 - E.g., first y' , then $y'z$, etc.
- A special type of product term is called *minterm*:
A minterm is a Boolean expression resulting in 1 for the output of a single cell

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

Boolean expression

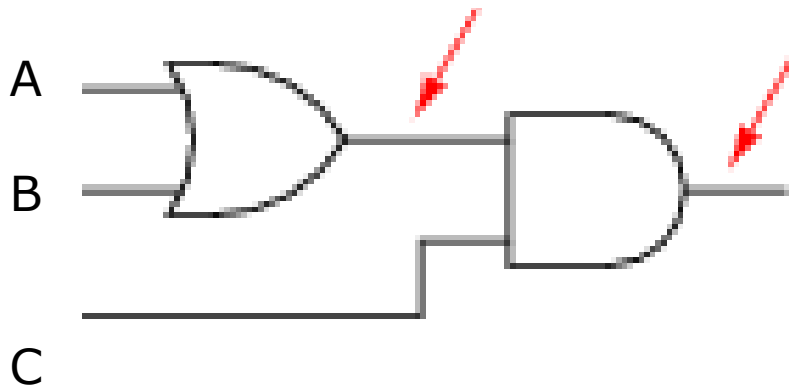
1. Write the Boolean expression for each of these logic gates, showing how the output (Q) algebraically relates to the inputs (A and B):



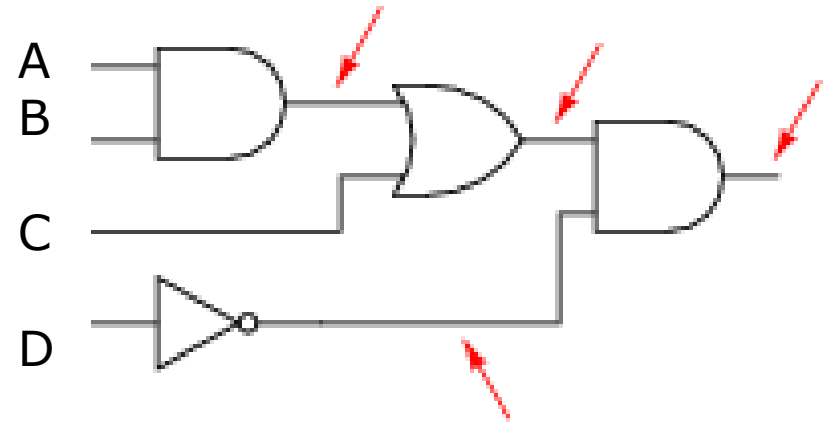
Boolean expression & Logic gate

1. Convert the following logic gate circuit into a Boolean expression, writing Boolean sub-expressions next to each gate output in the diagram:

(1)



(2)



Logic Circuit Diagram of Function

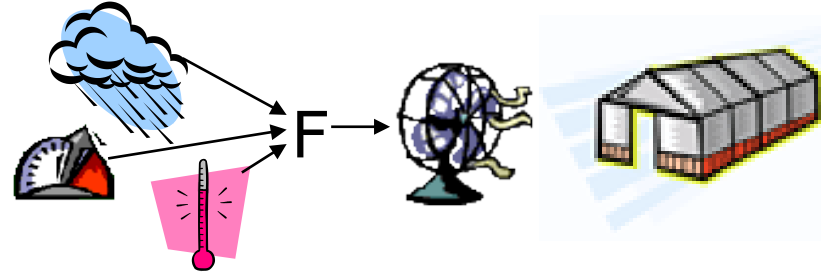
- Logic function can be expressed as circuit diagram
 - Direct translation from algebraic expression
- Example: $F = x + y' z$
- Problem: possibly multiple equivalent algebraic expressions
 - Difficult to compare Boolean functions

Design Problem

- Design the control logic for a fan in a greenhouse.

The logic must sense three environmental conditions:

1. It is raining outside (variable x)
2. It is humid inside (variable y)
3. It is hot inside (variable z)



The fan should turn on when

- » (It is not raining) AND (it is humid) AND (it is hot)

$$x' y z$$

OR

- » It is not humid AND { (it is raining) OR (it is not raining AND it is hot) }

$$y' (x + x'z)$$

- Create a Boolean function that controls the fan.

- $F = yx'z + y'(x'z+x)$
- Other possibilities: $F = x'yz+x'y'z+xy'$ or $F = x'z+xy'$

- Question: How to evaluate these solutions (equivalent functions)?

Boolean Function Representations

Greenhouse example function can be expressed as:

- Analytically, in standard form as sum of products:

$$yx'z + y'(x'z+x) = yx'z + y'x'z + y'x$$

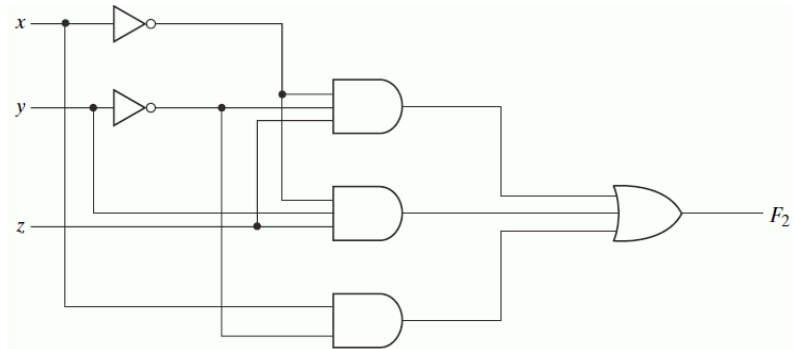
- As a truth table (canonical form),

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$= x'yz + x'y'z + xy'(z+z')$$

$$= x'yz + x'y'z + xy'z + xy'z'$$

- As a logic network



- How to derive different solutions ?
- How to evaluate these solutions ?

Boolean Theorems

- Huntington's postulates define some rules

Post. 1: closure

Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$

Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$

Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$

Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$

- Need more rules to modify algebraic expressions

- Theorems that are derived from postulates

- What is a theorem?

- A formula or statement that is derived from postulates (or other proven theorems)

- Basic theorems of Boolean algebra

- Theorem 1 (a): $x + x = x$ (b): $x \cdot x = x$
- Looks straightforward, but needs to be proven !

Boolean Theorems

- Other theorems that can be derived
 - Theorem 1(a): $x+x=x$
 - Theorem 1(b): $x \cdot x=x$
 - Theorem 2(a): $x+1=1$
 - Theorem 2(b): $x \cdot 0=0$
 - Theorem 3: $(x')'=x$
 - Theorem 4(a): $x+(y+z)=(x+y)+z$
 - Theorem 4(b): $x(yz)=(xy)z$
 - Theorem 6(a): $x+xy=x$
 - Theorem 6(b): $x(x+y)=x$
- Theorem 5: DeMorgan's theorem (duality)
 - Theorem 5(a): $(x+y)' = x'y'$
 - Theorem 5(b): $(xy)' = x'+y'$

Duality of Boolean Algebra

- Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged
 - Two-valued Boolean algebra:
 - » Exchange 1 and 0
 - » Exchange + and •
- Example:
 - Theorem 2(a): $x + 1 = 1$ and 2(b): $x \cdot 0 = 0$
- Implication for Boolean functions:
 - If identity elements and operators are interchanged, the result is the complement of the original function
 - [De Morgan's theorem](#): key to simplify Boolean expressions

De Morgan's Theorem

- Prove De Morgan's theorem by truth tables

Proof of $x+x=x$

- We can only use Huntington postulates:

Huntington postulates:

Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$

Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$

Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$

Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$

- Show that $x+x=x$.

$$\begin{aligned} x+x &= && \text{by 2(b)} \\ &= && \text{by 5(a)} \\ &= && \text{by 4(b)} \\ &= && \text{by 5(b)} \\ &= && \text{by 2(a)} \end{aligned}$$

Q.E.D.

- We can now use Theorem 1(a) in future proofs

Proof of $x \cdot x = x$

- Similar to previous proof

Huntington postulates:

Post. 2: (a) $x+0=x$, (b) $x \cdot 1=x$

Post. 3: (a) $x+y=y+x$, (b) $x \cdot y=y \cdot x$

Post. 4: (a) $x(y+z) = xy+xz$,
(b) $x+yz = (x+y)(x+z)$

Post. 5: (a) $x+x'=1$, (b) $x \cdot x'=0$

Th. 1: (a) $x+x=x$

- Show that $x \cdot x = x$.

$$\begin{aligned} x \cdot x &= && \text{by 2(a)} \\ &= && \text{by 5(b)} \\ &= && \text{by 4(a)} \\ &= && \text{by 5(a)} \\ &= && \text{by 2(b)} \end{aligned}$$

Q.E.D.

Comparison of Boolean Functions

- How to compare expressions?
- From the Greenhouse Example:
 - Is $yx'z + y'(x'z+x) = x'z+xy'$? (are these functions *equivalent*?)
 $yx'z + y'(x'z+x) \quad =$
 $\quad \quad \quad =$
 $\quad \quad \quad =$
 - Yes, both expressions describe the same function (equivalent)
- Problem: hard to compare algebraic expressions
 - Easier with gate level comparison?

Comparison of Boolean functions

- Comparison by algebraic expression
 - Good: can handle many variables
 - Bad: not clear how to get from one expression to another
- Comparison on gate level
 - Bad: same as algebraic expression
 - Bad: hard to modify for comparison
- Comparison by truth table
 - Good: Only one representation in truth table
 - Bad: cumbersome for 4 or more variables
 - Bad: hard to derive gate level implementation
- Comparison with canonical form
 - Good: standardized form for algebraic expression
 - Bad: not necessarily solution with least number of gates

Canonical Form

- A form is *canonical* if representation of a function in this form is unique
 - Truth table is canonical representation
 - Uses *minterms* as basic component
- A minterm is a product (*ANDing*) of all variables (or complements)
 - Each variable of function appears in minterm
 - Variable can appear in normal form (x) or in complemented form (x')
- Example: minterms for a 3-variable function
- All algebraic expressions can be converted into canonical form

x	y	z	minterm	designation
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Canonical Form

- Function is expressed as sum of minterms
- Greenhouse example: $yx'z + y'(x'z+x) =$

=
=
=
=

x	y	z	F	minterm	designation
0	0	0	0	$x'y'z'$	m_0
0	0	1	1	$x'y'z$	m_1
0	1	0	0	$x'yz'$	m_2
0	1	1	1	$x'yz$	m_3
1	0	0	1	$xy'z'$	m_4
1	0	1	1	$xy'z$	m_5
1	1	0	0	xyz'	m_6
1	1	1	0	xyz	m_7

- Function expression = sum of all minterms where $F=1$

Canonical Form

- Canonical form: sum of minterms for $F=1$

- Example: $F=A+B'C$

- Expansion of minterms:

$$A = ABC + ABC' + AB'C + AB'C'$$

$$B'C = AB'C + A'B'C$$

- $F = ABC + ABC' + AB'C + AB'C' + A'B'C$

- Alternate representations:

- $F = m_1 + m_4 + m_5 + m_6 + m_7$
- $F = \sum(1, 4, 5, 6, 7)$

- Is this the only canonical form?

- No, other forms exist (dual form, K-maps, etc.)

A	B	C	F	minterm	designation
0	0	0	0	$A'B'C'$	m_0
0	0	1	1	$A'B'C$	m_1
0	1	0	0	$A'BC'$	m_2
0	1	1	0	$A'BC$	m_3
1	0	0	1	$AB'C'$	m_4
1	0	1	1	$AB'C$	m_5
1	1	0	1	ABC'	m_6
1	1	1	1	ABC	m_7

De Morgan's Theorem

- A key theorem in simplifying Boolean expressions
- De Morgan's theorem expresses duality:
 - For two variables:

$$(x + y)' = x' \cdot y'$$

$$(x \cdot y)' = x' + y'$$

- Can be generalized to arbitrary number of variables

$$(a+b+c+ \dots +z)' = a' \cdot b' \cdot c' \dots \cdot z', \quad \text{or} \quad (\sum x_i)' = \prod x_i'$$

$$(a \cdot b \cdot c \cdot \dots \cdot z)' = a' + b' + c' + \dots + z', \quad \text{or} \quad (\prod x_i)' = \sum x_i'$$

where: \sum = OR, \prod = AND

Dual Canonical Form

- Sum of minterms can be converted to product of maxterms
- Recall:
 - A minterm is a product (AND) of all variables in respective polarities (positive or complemented)
 - e.g., $a' b c'$ is a minterm for a 3-variable function
 - Note: $a b'$ is not a minterm in a 3-variable function (it is a *product term*)
 - Function is OR of minterms where $F = 1$
- A maxterm is a sum (OR) of all variables in respective polarities
 - Maxterm contains every variable in the function
 - All the variables are OR-ed together in a maxterm
 - Function is product (AND) of maxterms where $F = 0$

Dual Canonical Form

- Product of maxterms

- Each maxterm contains every variable in the function
- A maxterm is an *OR* of variables in respective polarities
- Function is a product (*AND*-ing) of maxterms where $F = 0$

- Due to duality:

- $m'_j = M_j$

- Example:

- $m_0 = A'B'C'$
- $M_0 = (A'B'C')'$
 $= A+B+C$

- Duality:

- $F = \sum m_i = (\prod m'_i)'$
 $= (\prod M_i)'$

A	B	C	F	Maxterm	minterm
0	0	0	0		
0	0	1	1		
0	1	0	0		
0	1	1	0		
1	0	0	1		
1	0	1	1		
1	1	0	1		
1	1	1	1		

Dual Canonical Form

- Function is product (AND) of maxterms where $F = 0$
- Duality (DeMorgan):

- Example: $F = (m1, m4, m5, m6, m7)$
 - Examine the complement of F :
 $F' =$
 - Take a complement (DeMorgan):
 $(F')' =$
 - Since $(F')' = F$:
 $F =$

A	B	C	F	Maxterm	Designation
0	0	0	0	$A+B+C$	M_0
0	0	1	1	$A+B+C'$	M_1
0	1	0	0	$A+B'+C$	M_2
0	1	1	0	$A+B'+C'$	M_3
1	0	0	1	$A'+B+C$	M_4
1	0	1	1	$A'+B+C'$	M_5
1	1	0	1	$A'+B'+C$	M_6
1	1	1	1	$A'+B'+C'$	M_7

- Both canonical forms express same function:
 - $F = \sum m_i = \prod M_j$ over complementary index set

Sum of Product (SOP) Form

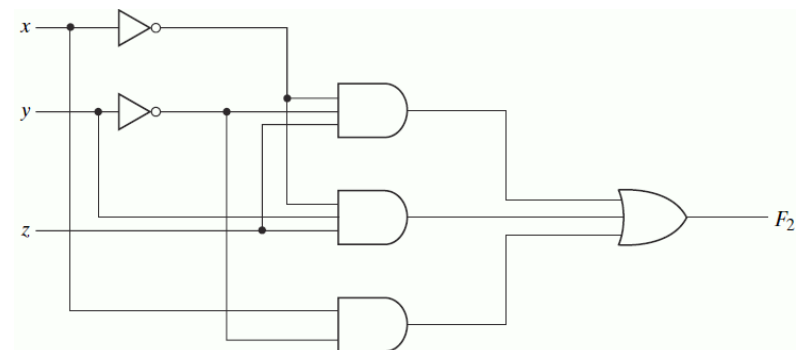
- Any function can be expressed as Sum of Products (SOP)
 - A product term is a product (AND) of literals (variables in resp. polarities)
 - Minterm is a special case of product term (has all variables)
 - SOP is not canonical, many SOP forms exist for same function
- Greenhouse example: $yx'z + y'(x'z+x) = x'yz + x'y'z + xy' = x'z + xy'$
- SOP table for $x'yz + x'y'z + xy'$

(list only for $F=1$)

x	y	z	F

- What is SOP table and gate diagram for $x'z + y'z$?

SOP has natural representation in practice, as *two-level* logic network

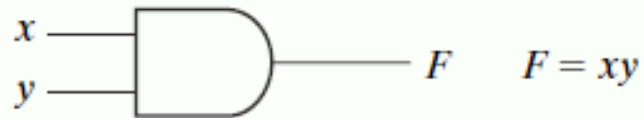


Counting Boolean Functions

- Now we can express algebraic functions uniquely
 - Unique sum of minterms, or
 - Unique product of maxterms
- How many 2-input Boolean functions exist?
 - How many different combinations of minterms can one pick?
 - There are 4 minterms in a 2-input function
 - Each minterm can be present in the sum or not
 - » Two choices for each minterm
 - Total $2^4=16$ combination
- How many Boolean functions exist with n variables?
 - 2^n minterms
 - Total: 2^{2^n} combinations

Digital Logic Gates

AND



x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR



x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Inverter



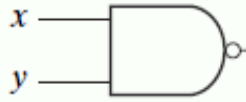
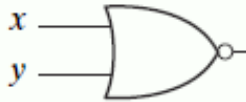
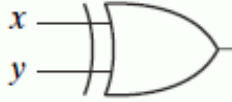

x	F
0	1
1	0

Buffer



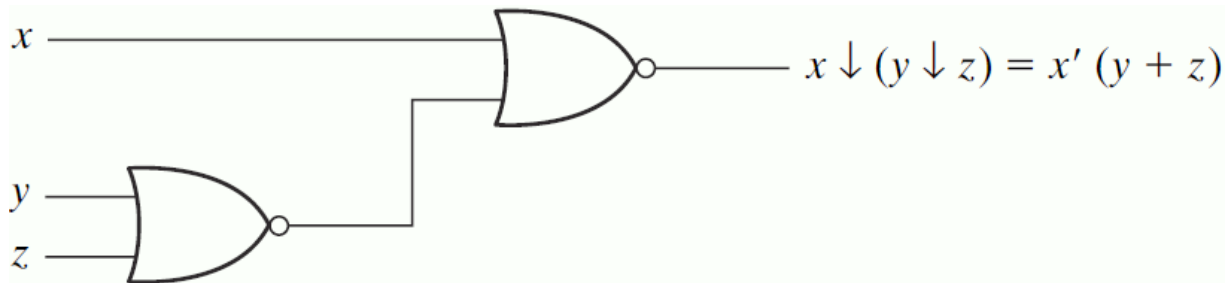
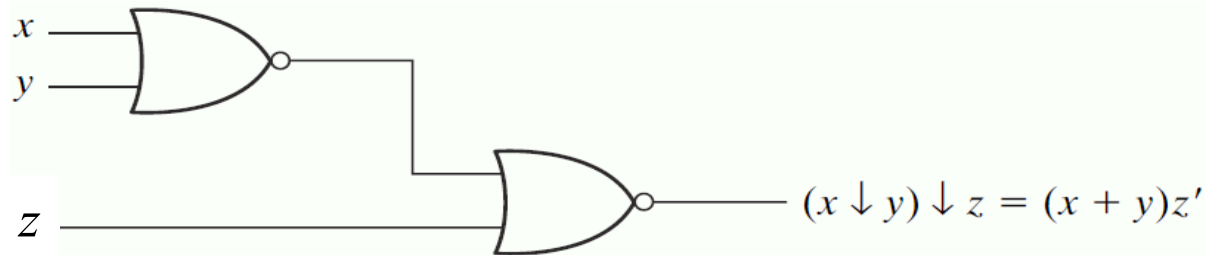
x	F
0	0
1	1

Digital Logic Gates

NAND	 $F = (xy)'$	<table> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F															
0	0	1															
0	1	1															
1	0	1															
1	1	0															
NOR	 $F = (x + y)'$	<table> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
Exclusive-OR (XOR)	 $F = xy' + x'y$ $= x \oplus y$	<table> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F															
0	0	0															
0	1	1															
1	0	1															
1	1	0															
Exclusive-NOR or equivalence	 $F = xy + x'y'$ $= (x \oplus y)'$	<table> <tr> <th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F															
0	0	1															
0	1	0															
1	0	0															
1	1	1															

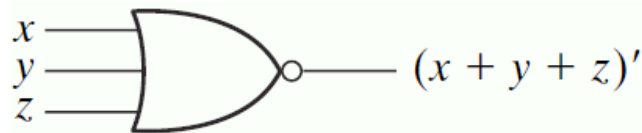
Multiple-input Gates

- NOR does not extend easily
 - Associativity does not hold for NOR (only for OR, AND, XOR)



Different functions !

- Multiple-input gate:



Multiple-input XOR

- Extension of exclusive-OR not straightforward
 - What is the definition for 3 variables?
- Turns into “odd” function
 - Function is 1 when there is an odd number of 1’s in input

x	y	z	$x \oplus y \oplus z$
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

- Uncommon in hardware implementations

2.14 Implement the Boolean function. $F = xy + x'y' + y'z$
(b) With OR and inverter gates

2.14 Implement the Boolean function. $F = xy + x'y' + y'z$
(d) With NAND and inverter gates

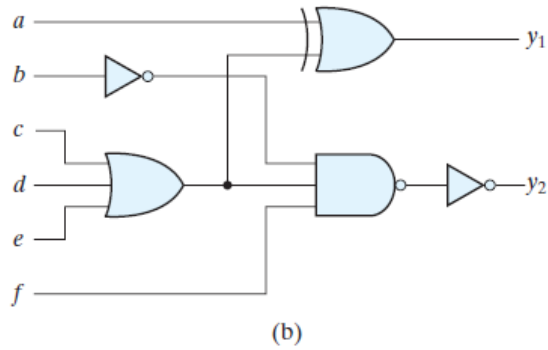
2.20 Express the complement of the following functions in sum of minterms form:

(a) $F(w,x,y,z) = \sum(2,4,6,8,12,14)$

2.21 Convert each of following to the other canonical form:

(b) $F(A,B,C,D) = \prod(3,5,8,11)$

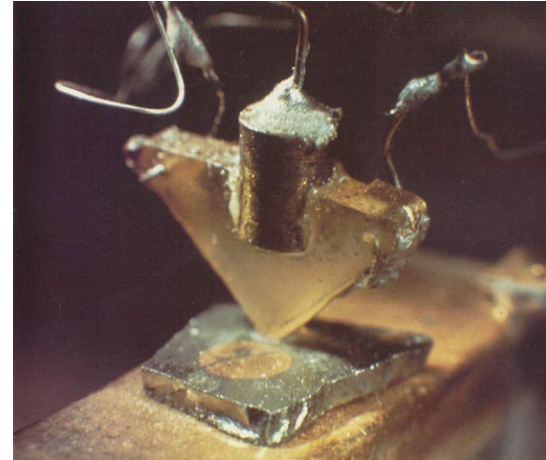
2.28 Write Boolean expression and construct the truth table describing the output of the logic gate circuit below.



Technology Context: IC (R)evolution

Transistor Invention (1947)

- Shockley, Bardeen & Brattan, Bell Labs
- Bipolar junction transistor (BJT)
- 1956 Nobel Prize in Physics
- Switch/Amplify Signal

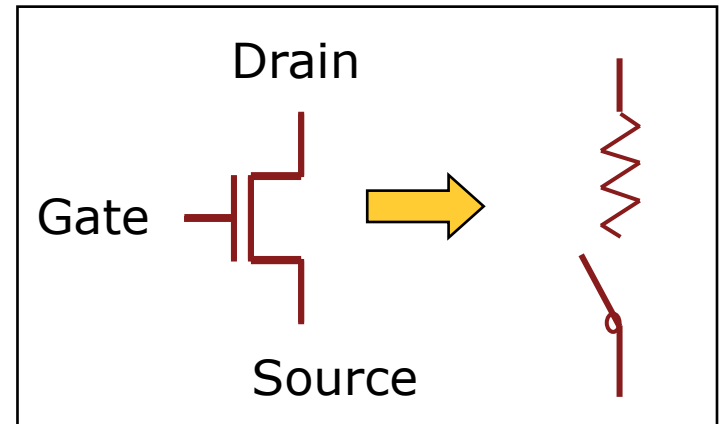
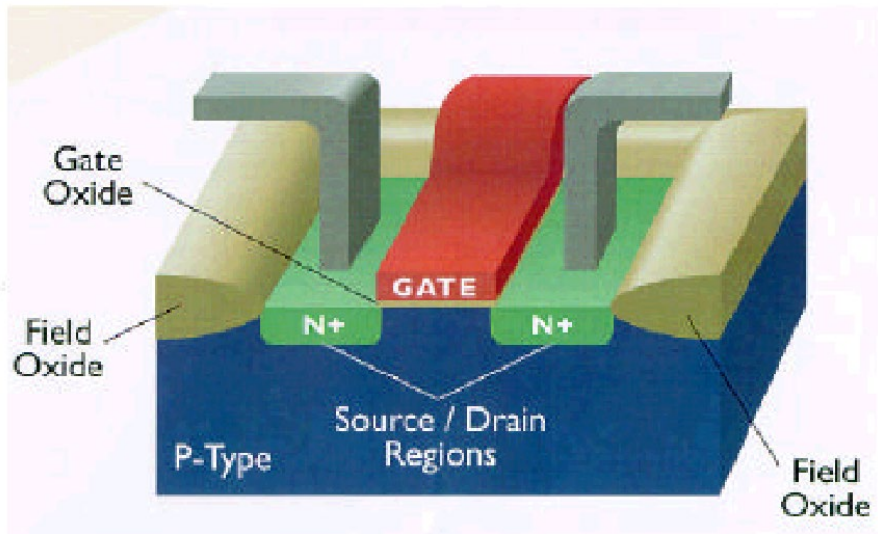


First Integrated Circuit (1958)

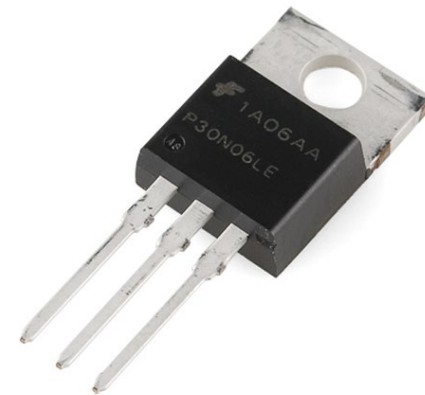
- Jack Kilby, Texas Instruments
- A few components on single chip
- 2000 Nobel Prize in Physics

MOSFET Transistor

- MOSFET Transistor—functions as a voltage-controlled switch

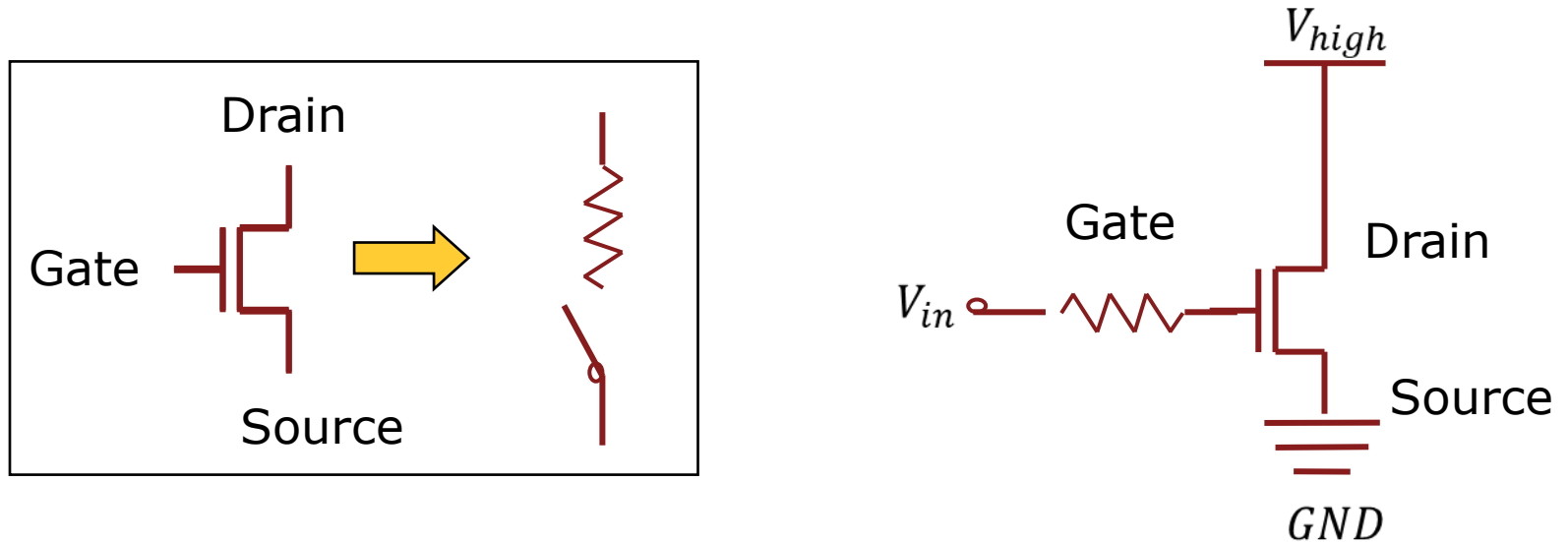


- Modern (Tri-Gate) Transistor
Like MOSFET, but better at
extreme dimensions (~10nm)



MOSFET Transistor

- MOSFET Transistor—functions as a voltage-controlled switch



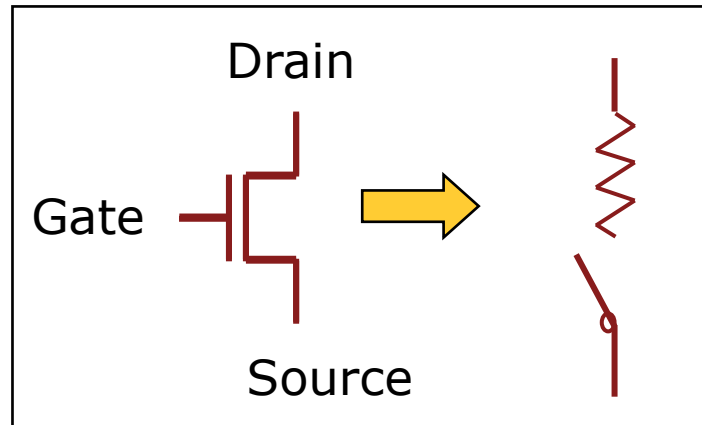
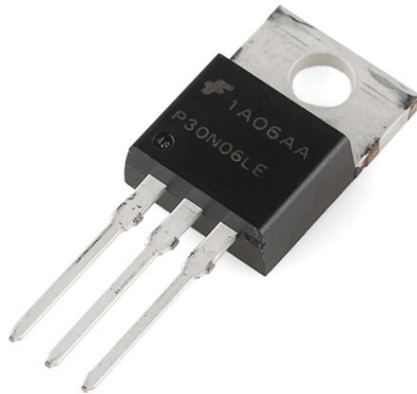
Working Mechanism:

when voltage is applied between the gate and the source, current is allowed to flow between the drain and the source.

- $V_{gate} = \text{high}(1)$, drain is connected to source;
- $V_{gate} = \text{low}(0)$, drain is disconnected to the source;

MOSFET Transistor

- MOSFET Transistor—functions as a voltage-controlled switch



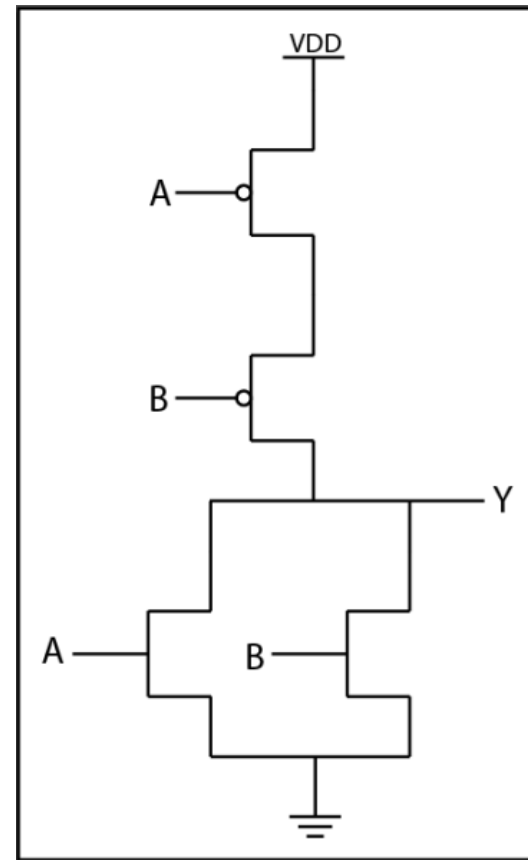
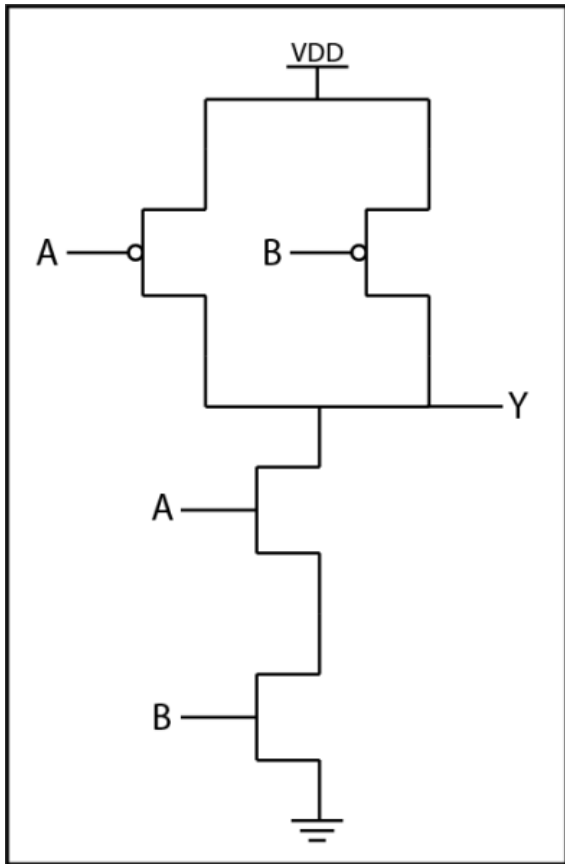
Resistance

There are variable resistance controlled by voltage which means depending on the voltage applied between the gate and the source, the resistance between the drain and the source will vary.

- $V_{gate} = 1$, drain is connected to source, the resistance from the drain to the source is very low, it will act as a wire.
- $V_{gate} = 0$, drain is disconnected to the source, the resistance from the drain to the source is very high, it will act as an open switch.

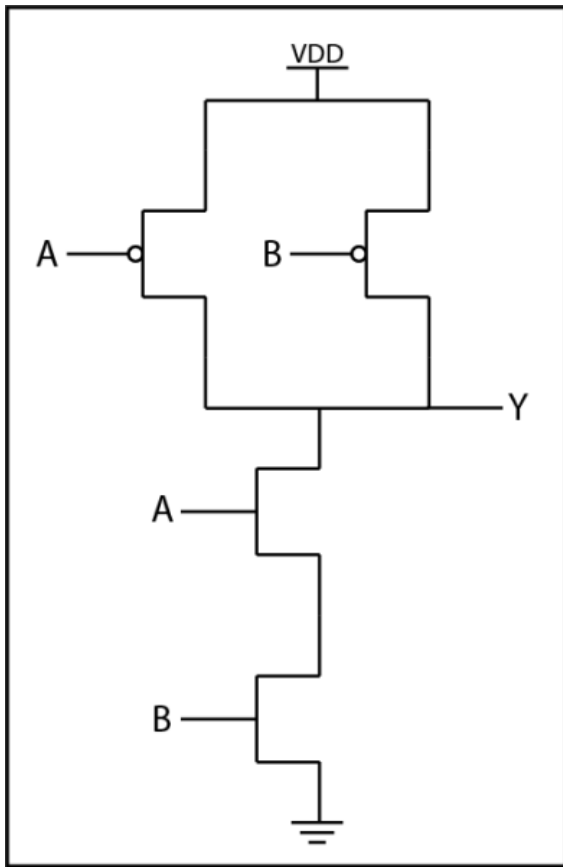
Logic Gates = connected switches

Several-transistor circuits that perform elementary logic functions



Logic Gates = connected switches

Several-transistor circuits that perform elementary logic functions



1. 0 \Rightarrow switch is disconnected
2. 1 \Rightarrow switch is connected

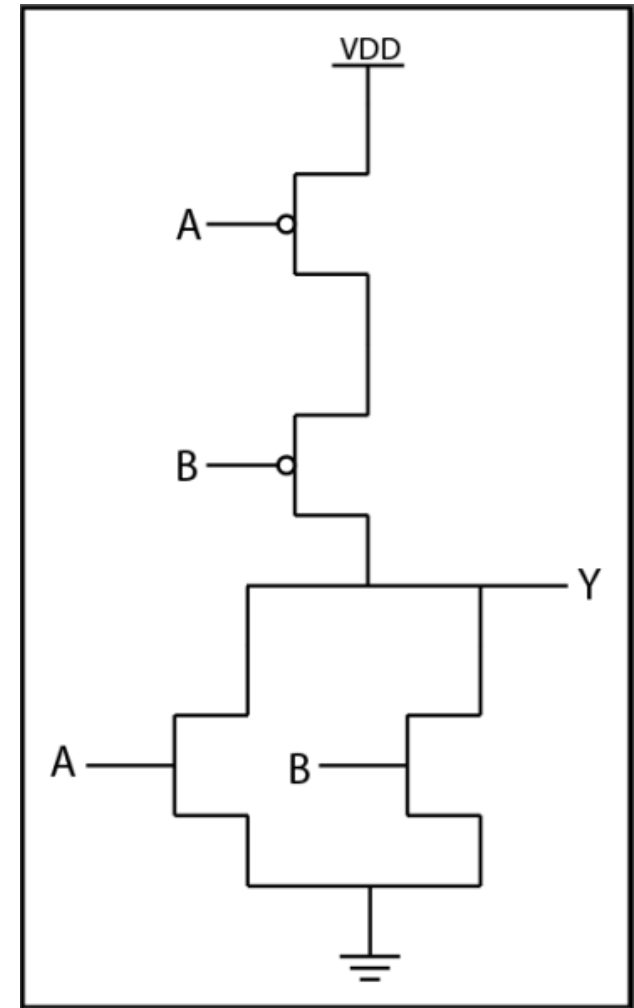
A	B	Vout
0	0	
0	1	
1	0	
1	1	

Logic Gates = connected switches

Several-transistor circuits that perform elementary logic functions

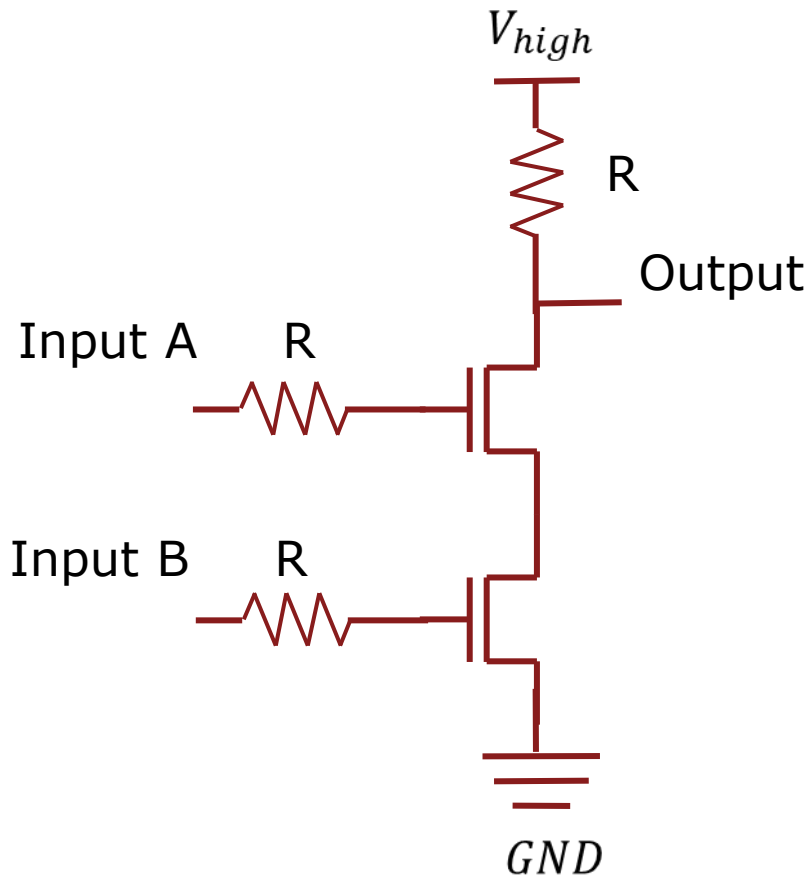
1. 0 => switch is disconnected
2. 1 => switch is connected

A	B	Vout
0	0	
0	1	
1	0	
1	1	



Example

Please describe which gate does the following scheme represent out of transistors and a resistor:

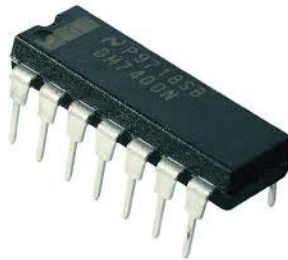


Input A	Input B	Output
0	0	
0	1	
1	0	
1	1	

Digital ICs: Logic Gates

■ Venerable “7400 Series”

- “Small-scale” ICs (SSI)
- Individual gates
- Useful circuits
- Many circuits from ECE 124!



■ Pin Diagram (or “Pinout”)

- Shows connections to
 - » Gate inputs and outputs
 - » Power Supply (V_{cc} : +V)
 - » Ground (GND: 0 V)

- Easy to build circuits with these chips, a breadboard, and wires
- See parts wall in M5!

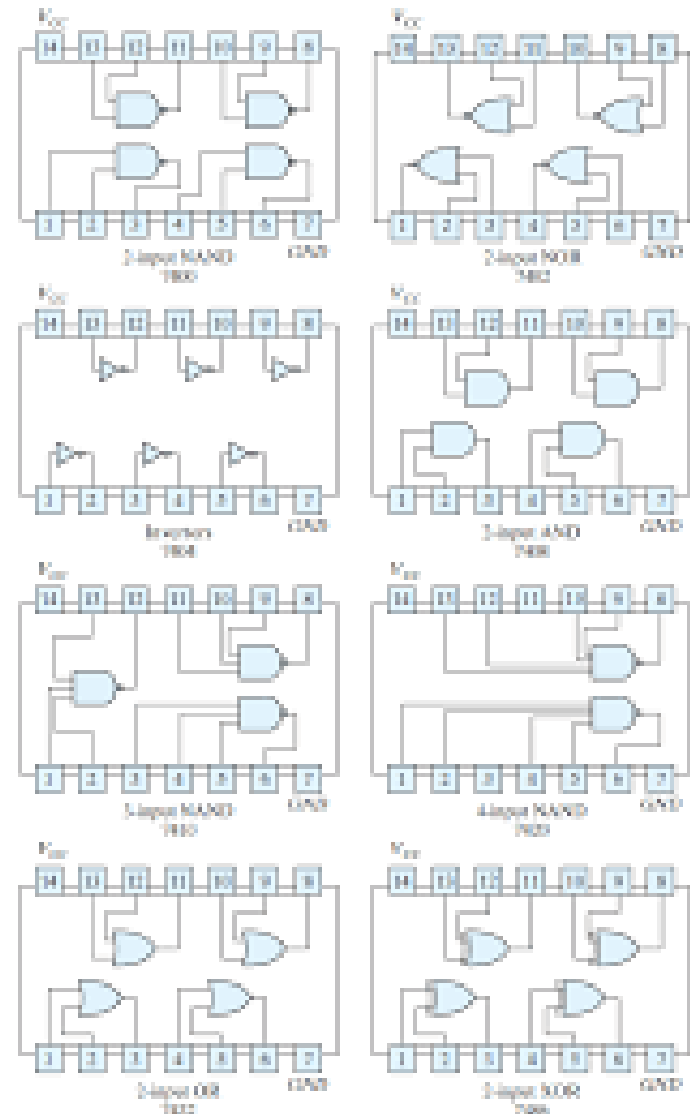


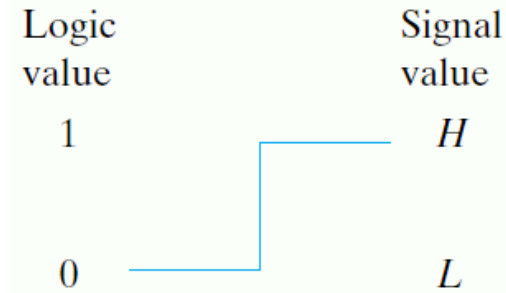
FIGURE 9.1

Digital gates in IC packages with identification numbers and pin assignments.

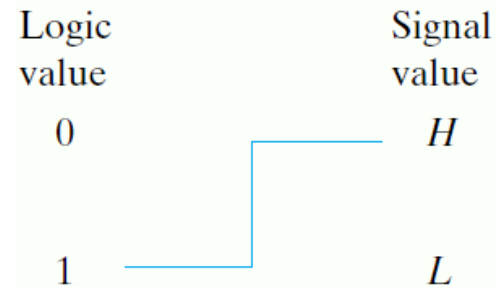
Positive and Negative Logic

- Signal levels are represented as H (high) and L (low)
 - e.g., H is 5V and L is 0V

- Positive logic
 - H corresponds to logic 1
 - L corresponds to logic 0



- Negative logic
 - H corresponds to logic 0
 - L corresponds to logic 1

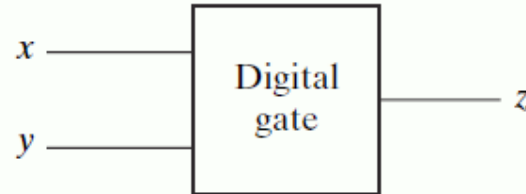


- User has choice of logic used

Positive and Negative Logic

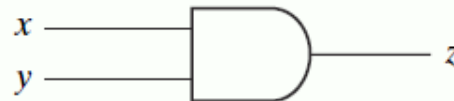
- Truth table with H and L

x	y	F
L	L	L
L	H	L
H	L	L
H	H	H



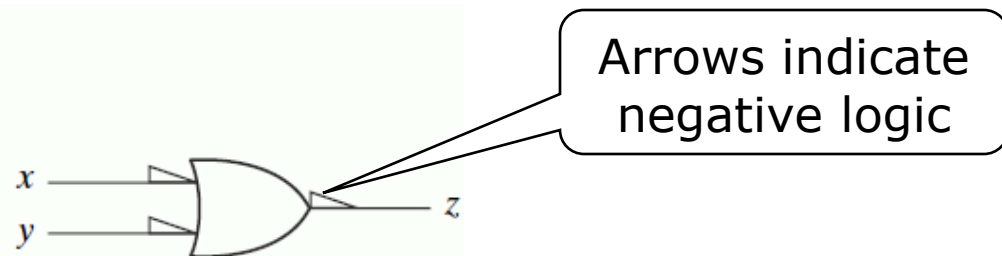
- In positive logic: AND

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1



- In negative logic: OR

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0



Integrated Circuits

- Small-scale integration (SSI)
 - 10s of logic gates
 - Early 60's for Apollo and Minuteman missiles
- Medium-scale integration (MSI)
 - 100s – 1,000s of logic gates
 - Late 60's
- Large-scale integration (LSI)
 - 1,000s – 10,000s of logic gates
 - Mid 70's
- Very large-scale integration (VLSI)
 - 100,000s of logic gates
 - 80's
- Ultra large-scale integration (ULSI)
 - Millions of logic gates
 - 90's and 00's

Chip Design

- Why is it better to have more gates on a single chip?
 - Easier to build systems
 - Lower power consumption
 - Higher clock frequencies

- What are the drawbacks of large circuits?
 - Complex to design
 - Chips have design constraints
 - Hard to test

- Need tools to help develop integrated circuits
 - Computer Aided Design (CAD) tools
 - Automate tedious steps of design process
 - Hardware description language (HDL) describe circuits
 - Verilog is one such system