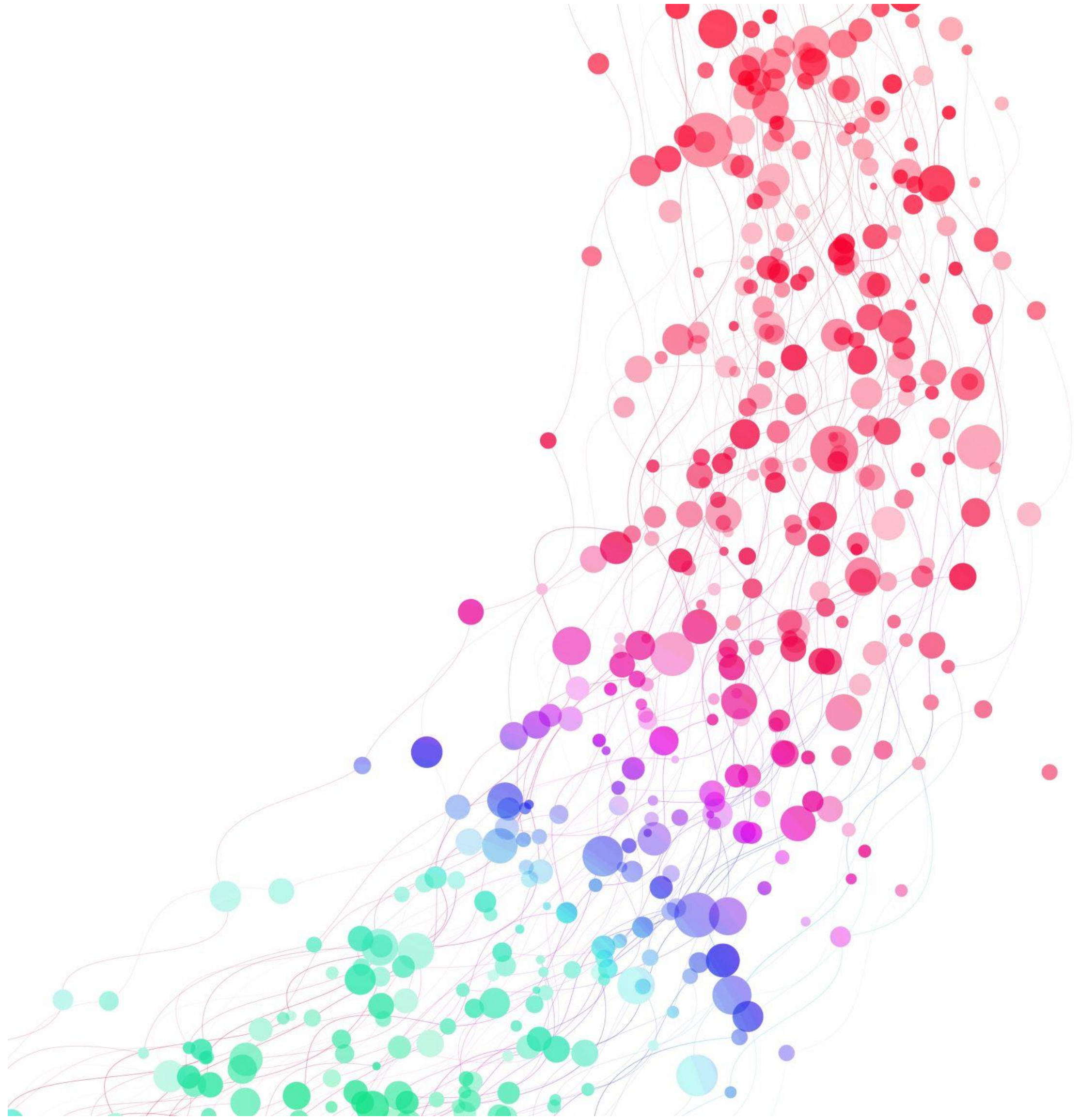
Fundamentals of Electronics

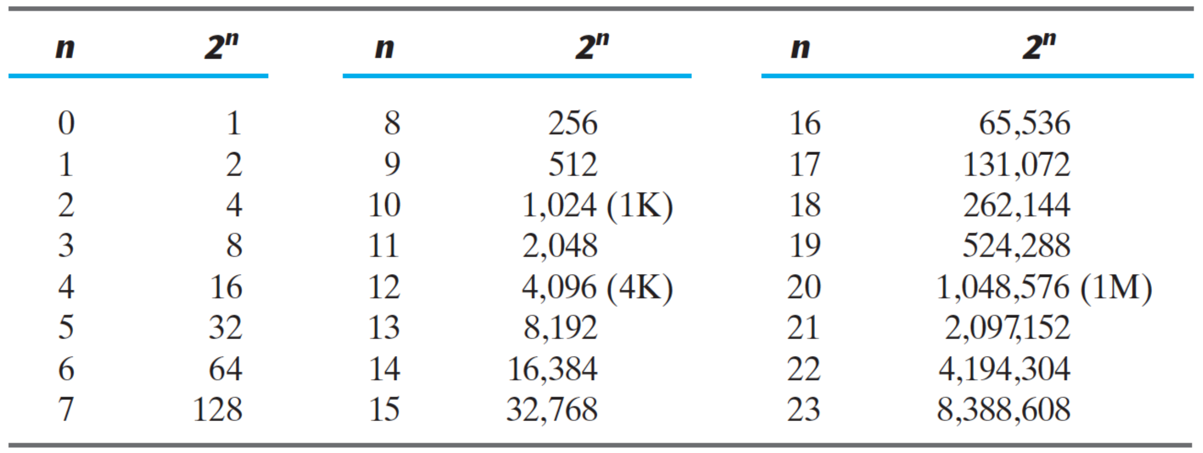
ECE 101



Digital Electronics

Digital computing

In digital computing:



n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024 (1K)	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096 (4K)	20	1,048,576 (1M)
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

- 2^{10} is referred to as K (kilo)
- 2^{20} as M (mega)
- 2^{30} as G (giga)
- 2^{40} as T (tera)
- 1 byte = 8 bits (one byte can store one character, e. g. 'A' or 'x' or '$')
- Thus, 4K = 2^{12} = 4,096 and 16M = 2^{24} = 16,777,216.

Complement of numbers: Radix and Diminished radix

r' s= complement= $\{(r^n)_{10}\}_r - N$

(r-1)' s complement= $\{(r^n)_{10} - 1\}_r - N$

The n is the number of digits in the number.
The N is the given number.
The r is the radix or base of the number.

Examples: 1. $(1011000)_2$

1's complement: $\{(2^7)_{10}-1\} - (1011000)_2 = \{(128)_{10}-1\}_2 - (1011000)_2 = 1111111_2 - 1011000_2 = 0100111$

2's complement: $\{(2^7)_{10}\} - (1011000)_2 = \{(128)_{10}\}_2 - (1011000)_2 = 10000000_2 - 1011000_2 = 0101000_2$

2. $(155)_{10}$

9's complement: $\{(10^3)_{10}-1\} - (155)_{10} = (1000-1) - 155 = 999 - 155 = (844)_{10}$

10's complement: $\{(10^3)_{10}\} - (155)_{10} = (1000) - 155 = 1000 - 155 = (845)_{10}$

3. $(174)_8 = ?$

Subtraction with complements

1. Add the minuend M to the r's complement of the subtrahend N. **Mathematically:** $M + (r^n - N) = M - N + r^n$.
2. If **M > N**, the sum will produce an end carry r^n, which can be discarded; what is left is the result **M - N**.
3. If **M < N**, the sum does not produce an end carry and is equal to $\mathbf{r^n - (N - M)}$, which is the r's complement of (N - M). To obtain the answer in a familiar form, **take the r's complement of the sum** and **place a negative sign in front**.

Example 1: Using 10's complement, subtract **72532 - 3250**

$$\begin{aligned} M &= 72532 \\ \text{10's complement of } N &= +\ \underline{96750} \\ \text{Sum} &= 169282 \\ \text{Discard end carry } 10^5 &= -\ \underline{100000} \\ \mathit{Answer} &= 69282 \end{aligned}$$

Subtraction with complements

Example 2: Using 10's complement, subtract **3250 - 72532**

$$\begin{aligned} M &= \ 03250 \\ \text{10's complement of } N &= +\ \underline{27468} \\ \text{Sum} &= \ 30718 \end{aligned}$$

the answer is $-(\text{10's complement of } 30718) = -69282.$

Example 3: Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) X - Y and (b) Y - X by using 2's complements.

Subtraction with complements

Example 3: Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) X - Y and (b) Y - X by using 2's complements.

(a)

$$\begin{array}{rr}
X = & 1010100 \\
\text{2's complement of } Y = & +\ \underline{0111101} \\
\text{Sum} = & 10010001 \\
\text{Discard end carry } 2^7 = & -10000000 \\
\textit{Answer: } X - Y = & 0010001
\end{array}$$

(b)

$$\begin{array}{rr}
Y = & 1000011 \\
\text{2's complement of } X = & +\ 0101100 \\
\text{Sum} = & 1101111
\end{array}$$

There is no end carry. Therefore, the answer is $Y - X = -(\text{2's complement of } 1101111) = -0010001$.

Subtraction with complements

Example 4: Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) X - Y and (b) Y - X by using 1's complements.

(a) $X - Y = 1010100 - 1000011$

$$\begin{array}{rr}
X = & 1010100 \\
\text{1's complement of } Y = + & \underline{0111100} \\
\text{Sum} = & 10010000 \\
\text{End-around carry } = + & 1 \\
\textit{Answer}\text{: } X - Y = & 0010001
\end{array}$$

(b) $Y - X = 1000011 - 1010100$

$$\begin{array}{rr}
Y = & 1000011 \\
\text{1's complement of } X = + & 0101011 \\
\text{Sum} = & 1101110
\end{array}$$

There is no end carry. Therefore, the answer is $Y - X = -(\text{1's complement of } 1101110) = -0010001$.

Signed binary numbers

Computers must represent everything with binary digits.

It is customary to represent the sign with a bit placed in the leftmost position of the number.

The convention is to make the sign bit 0 for positive and 1 for negative.

01001 = 9 or +9

11001 = 25 or -9 **Signed-magnitude convention**

Signed-complement system

A negative number is indicated by its complement

- Either the 1's or the 2's complement, but the 2's complement is the most common

01001 = 9 or +9

signed-magnitude representation:	10001001
signed-1's-complement representation:	11110110
signed-2's-complement representation:	11110111

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Arithmetic Addition/Subtraction

Generally, addition/subtraction involves comparison of signs and magnitudes and performing operations.

- If the signs are the same, we add the two magnitudes and give the sum the common sign.
- If the signs are different, we subtract the smaller magnitude from the larger and give the difference the sign of the larger magnitude.
- The same procedure applies to binary numbers in signed-magnitude representation.
- The rule for adding numbers in the signed-complement system does not require a comparison or subtraction, but only addition.
- **The addition of two signed binary numbers (negative numbers represented in signed-2's-complement form) is obtained from the addition of the two numbers, including their sign bits. A carry out of the sign-bit position is discarded.**

Arithmetic Addition/Subtraction

- **The addition of two signed binary numbers (negative numbers represented in signed-2's-complement form) is obtained from the addition of the two numbers, including their sign bits. A carry out of the sign-bit position is discarded.**

$$
\begin{array}{rr}
+\ 6 & 00000110 \\
+13 & 00001101 \\
\hline
+19 & 00010011
\end{array}
\qquad
\begin{array}{rr}
-\ 6 & 11111010 \\
+13 & 00001101 \\
\hline
+\ 7 & 00000111
\end{array}
\qquad
\begin{array}{rr}
+\ 6 & 00000110 \\
-13 & 11110011 \\
\hline
-\ 7 & 11111001
\end{array}
\qquad
\begin{array}{rr}
-\ 6 & 11111010 \\
-13 & 11110011 \\
\hline
-19 & 11101101
\end{array}
$$

- In order to obtain a correct answer, we must ensure that the result has a sufficient number of bits to accommodate the sum.

- **Subtraction:** Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit). A carry out of the sign-bit position is discarded.

Arithmetic Addition/Subtraction

- **Subtraction:** Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit). A carry out of the sign-bit position is discarded.

$$(\pm A) - (+B) = (\pm A) + (-B);$$
$$(\pm A) - (-B) = (\pm A) + (+B).$$

- Changing a positive number to a negative number is easily done by taking the 2's complement of the positive number.
- The reverse is also true.

- **Binary numbers in the signed-complement system are added and subtracted by the same basic addition and subtraction rules as unsigned numbers. Therefore, computers need only one common hardware circuit to handle both types of arithmetic.**

Information: Codes

Binary Coded Decimal (BCD)

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

BCD numbers are decimal numbers and not binary numbers, although they use bits in their representation.

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD Addition

$$\begin{array}{rr} 4 & 0100 \\ +5 & +0101 \\ \hline 9 & 1001 \end{array} \qquad \begin{array}{rr} 4 & 0100 \\ +8 & +1000 \\ \hline 12 & 1100 \\ & +0110 \\ \cline{2-2} & 10010 \end{array} \qquad \begin{array}{rr} 8 & 1000 \\ +9 & 1001 \\ \hline 17 & 10001 \\ & +0110 \\ \cline{2-2} & 10111 \end{array}$$

ASCII Code

American Standard Code for Information Interchange (ASCII)

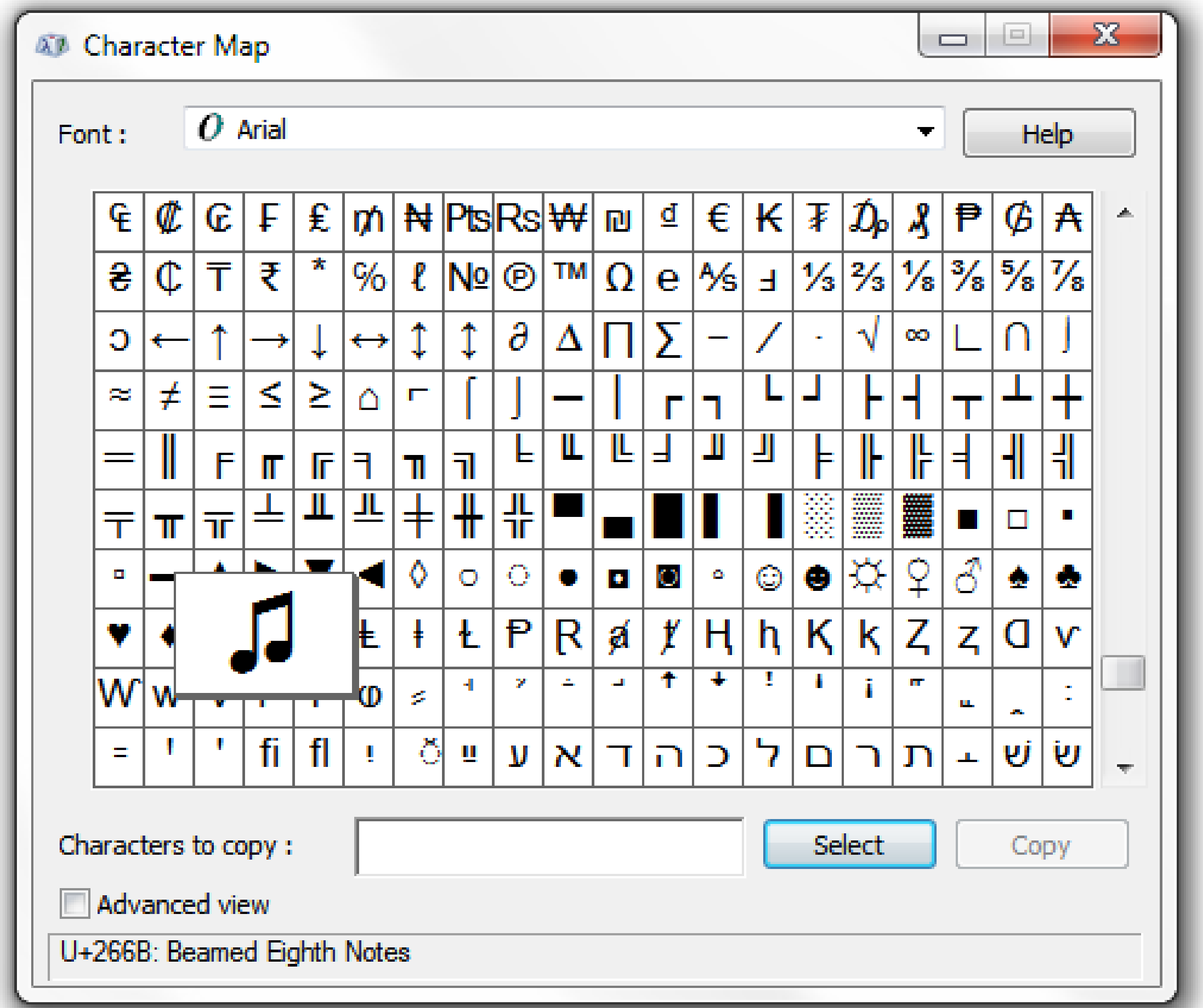
	$b_7b_6b_5$							
$b_4b_3b_2b_1$	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	−	o	DEL

Control Characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Unicode

Universal Coded Character Set



U+266B: Beamed Eighth Notes

UTF-8

UTF-16

UTF-32

Information storage and transfer

Register

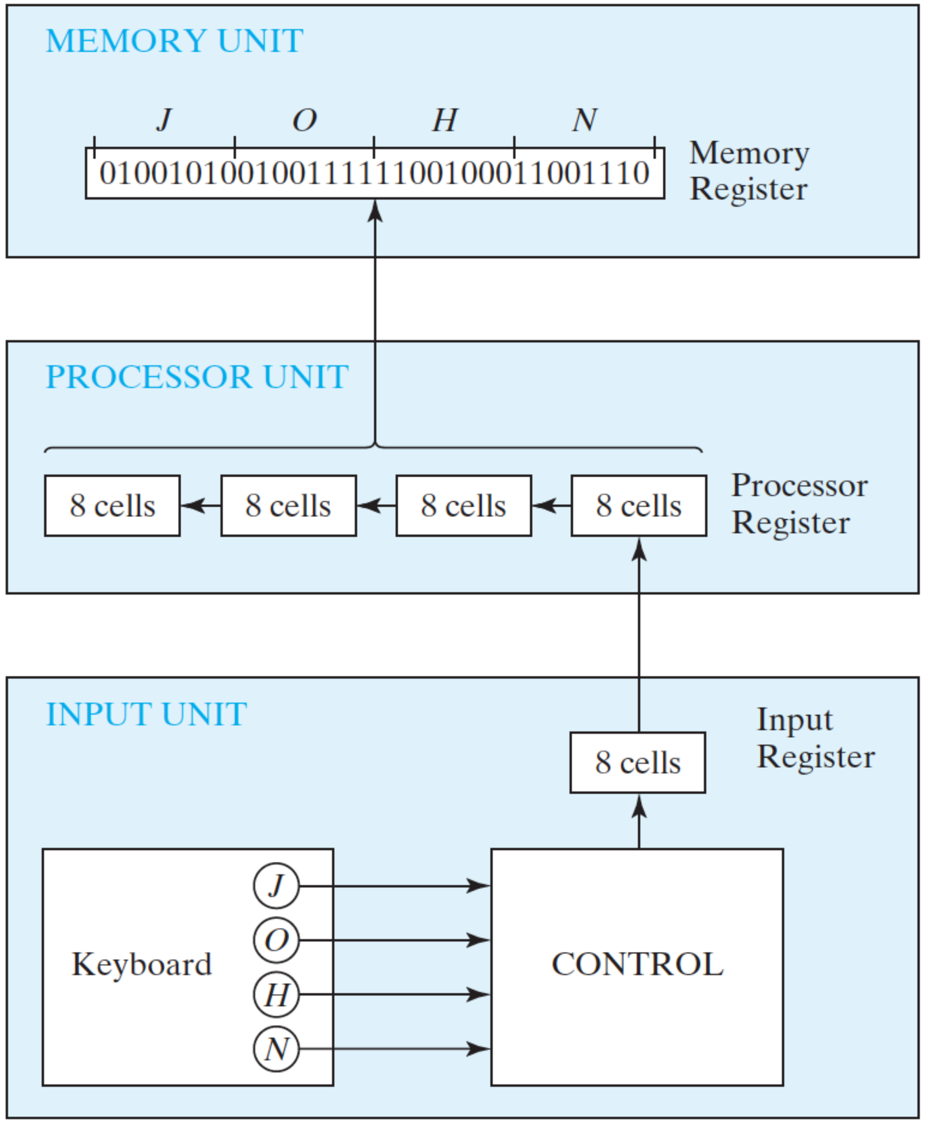
A binary cell is a device that possesses two stable states and is capable of storing one bit (0 or 1) of information.

A register is a group of binary cells.

A register with n cells can store any discrete quantity of information that contains n bits.

1100001111001001

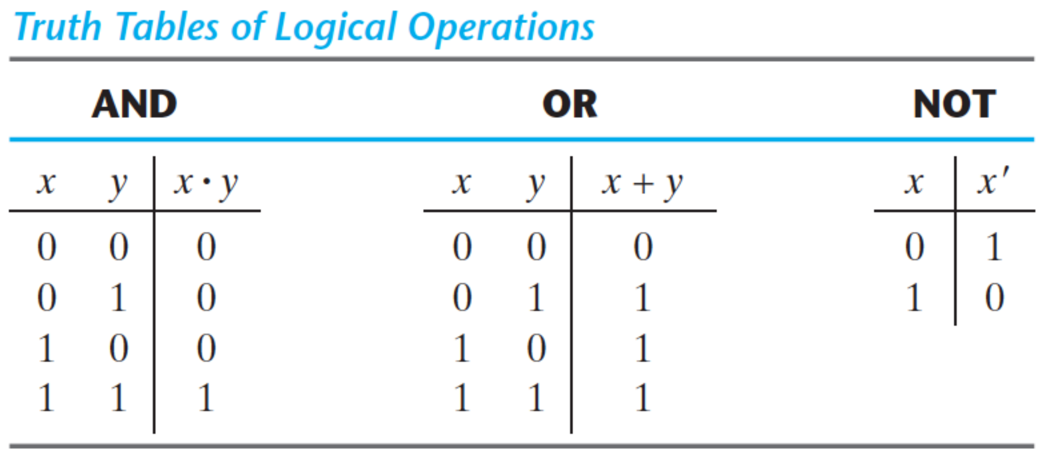
Register transfer



Binary Logic/Boolean Algebra

- **Field:** binary numbers (0, 1)
- **Operation(s):** ∧ (AND), ∨ (OR), and ¬ ("complement" or "not")

$a \lor (b \lor c) = (a \lor b) \lor c$	$a \land (b \land c) = (a \land b) \land c$	**Associativity**
$a \lor b = b \lor a$	$a \land b = b \land a$	**Commutativity**
$a \lor (a \land b) = a$	$a \land (a \lor b) = a$	**Absorption**
$a \lor 0 = a$	$a \land 1 = a$	**Identity**
$a \lor (b \land c) = (a \lor b) \land (a \lor c)$	$a \land (b \lor c) = (a \land b) \lor (a \land c)$	**Distributivity**
$a \lor \neg a = 1$	$a \land \neg a = 0$	**Complements**


Truth Tables of Logical Operations

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

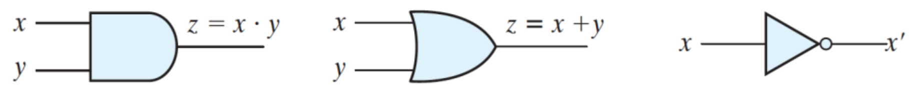
OR

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

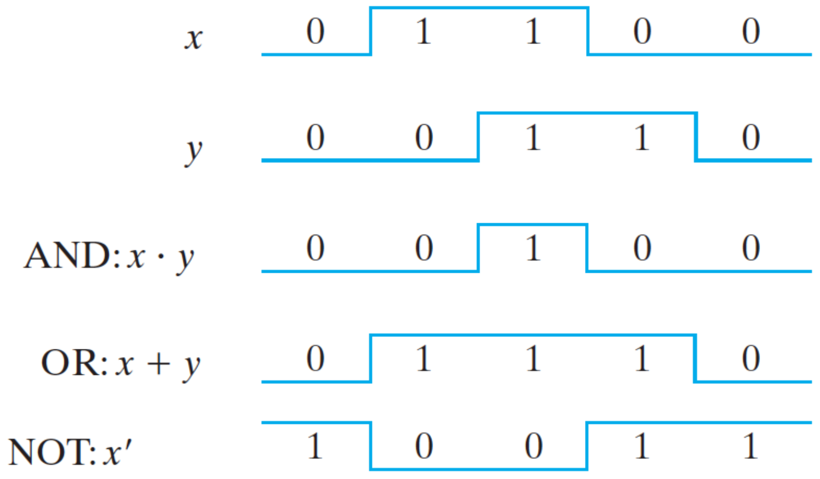
NOT

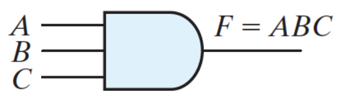
x	x'
0	1
1	0

Digital logic gates

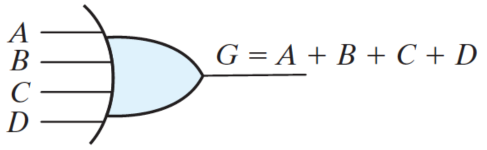


(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

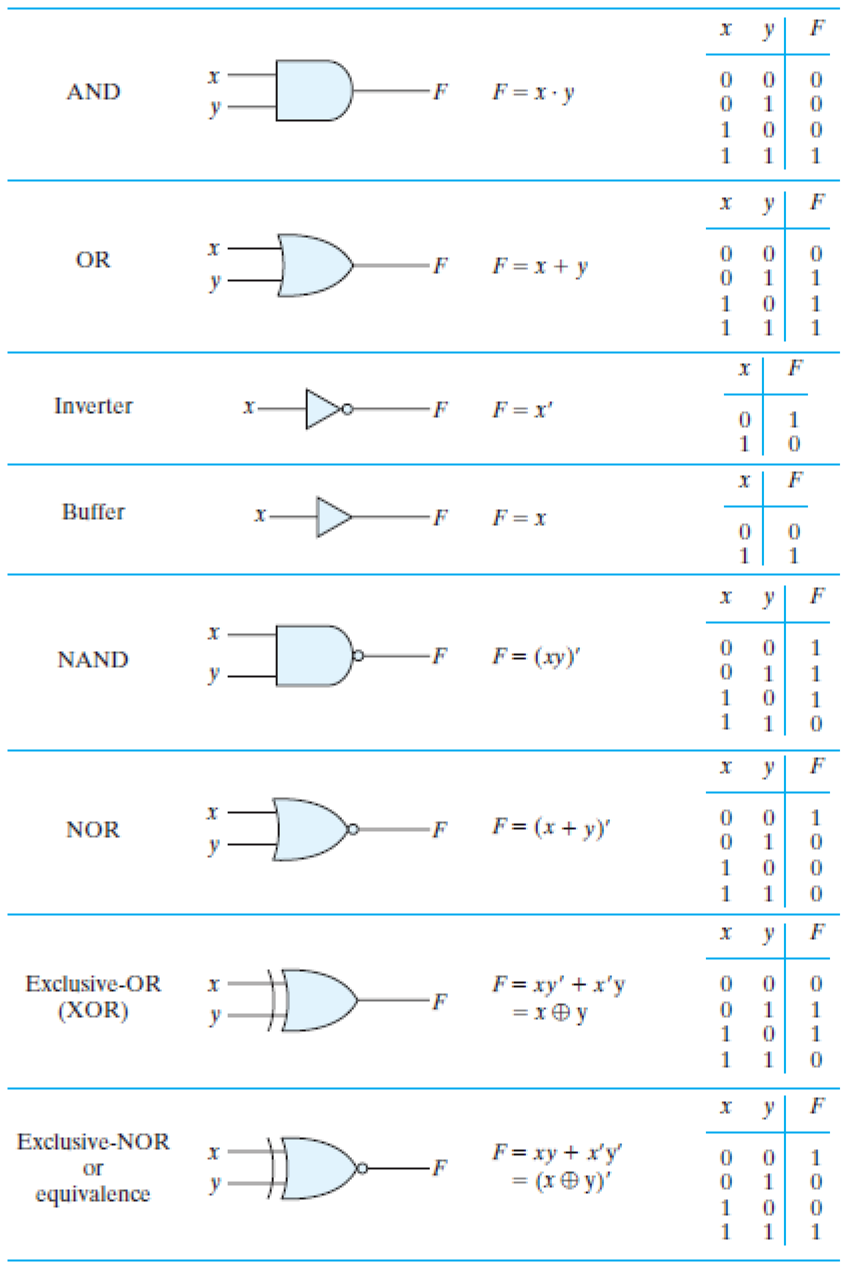




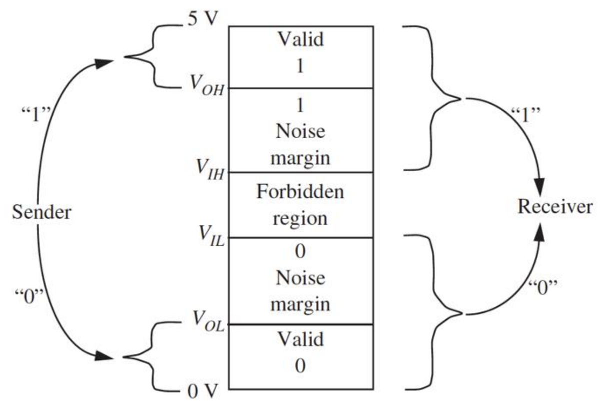
(a) Three-input AND gate



(b) Four-input OR gate

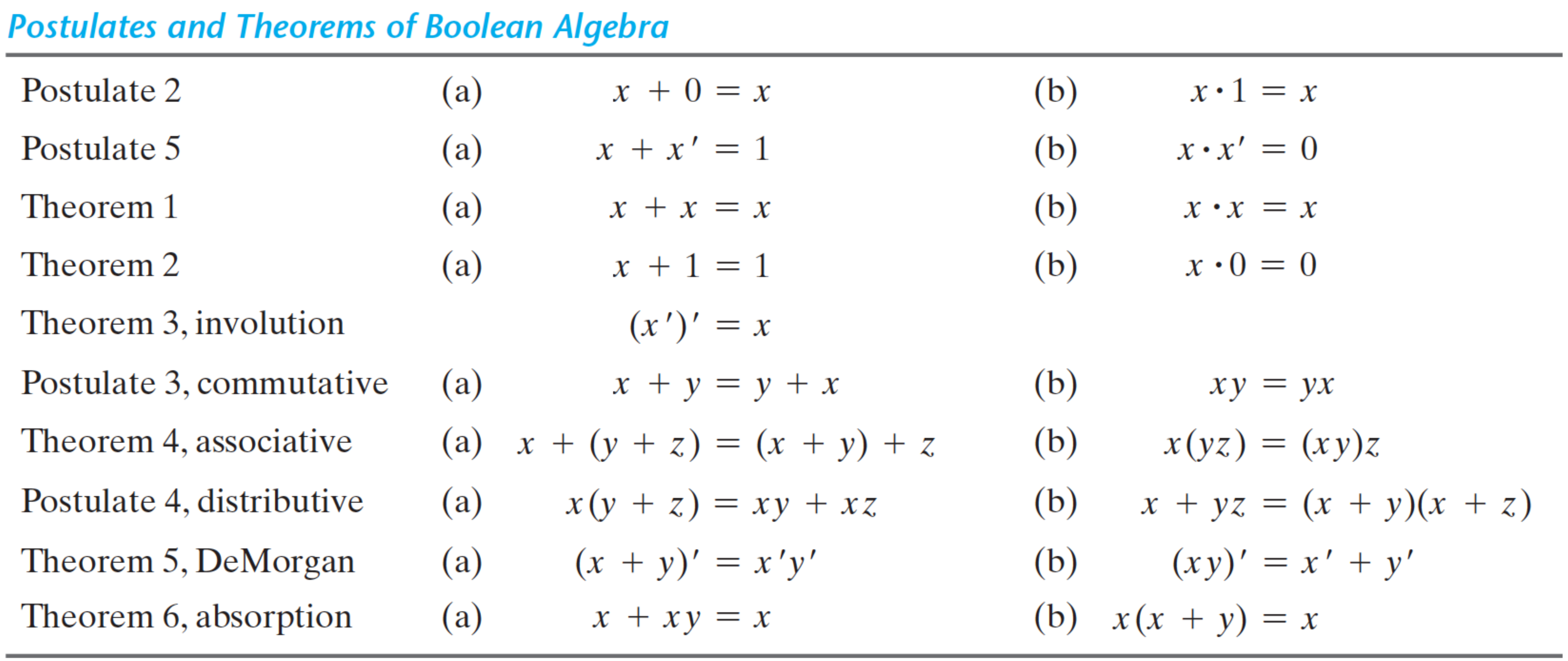


		x	y	F
AND	$F = x \cdot y$	0	0	0
		0	1	0
		1	0	0
		1	1	1
OR	$F = x + y$	0	0	0
		0	1	1
		1	0	1
		1	1	1
Inverter	$F = x'$	0		1
		1		0
Buffer	$F = x$	0		0
		1		1
NAND	$F = (xy)'$	0	0	1
		0	1	1
		1	0	1
		1	1	0
NOR	$F = (x + y)'$	0	0	1
		0	1	0
		1	0	0
		1	1	0
Exclusive-OR (XOR)	$F = xy' + x'y = x \oplus y$	0	0	0
		0	1	1
		1	0	1
		1	1	0
Exclusive-NOR or equivalence	$F = xy + x'y' = (x \oplus y)'$	0	0	1
		0	1	0
		1	0	0
		1	1	1



Apart from gates, we have registers, counters, memory elements.

Basic Theorems of Boolean Algebra



Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Duality Principle:

- Every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.
- If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1’s by 0’s and 0’s by 1’s.

Basic Theorems of Boolean Algebra

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

Statement	Justification
$x + x = (x + x) \cdot 1$	postulate 2(b)
$= (x + x)(x + x')$	5(a)
$= x + xx'$	4(b)
$= x + 0$	5(b)
$= x$	2(a)

THEOREM 1(b): $x \cdot x = x$.

Statement	Justification
$x \cdot x = xx + 0$	postulate 2(a)
$= xx + xx'$	5(b)
$= x(x + x')$	4(a)
$= x \cdot 1$	5(a)
$= x$	2(b)

THEOREM 2(a): $x + 1 = 1$.

Statement	Justification
$x + 1 = 1 \cdot (x + 1)$	postulate 2(b)
$= (x + x')(x + 1)$	5(a)
$= x + x' \cdot 1$	4(b)
$= x + x'$	2(b)
$= 1$	5(a)

THEOREM 2(b): $x \cdot 0 = 0$ by duality.

THEOREM 3: $(x')' = x$. From postulate 5, we have $x + x' = 1$ and $x \cdot x' = 0$, which together define the complement of x. The complement of x' is x and is also $(x')'$.

Basic Theorems of Boolean Algebra

THEOREM 6(a): $x + xy = x$.

Statement	Justification
$x + xy = x \cdot 1 + xy$	postulate 2(b)
$= x(1 + y)$	4(a)
$= x(y + 1)$	3(a)
$= x \cdot 1$	2(a)
$= x$	2(b)

THEOREM 6(b): $x(x + y) = x$ by duality.

Operator Precedence

The operator precedence for evaluating Boolean expressions is

(1) parentheses

(2) NOT

(3) AND

(4) OR.

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

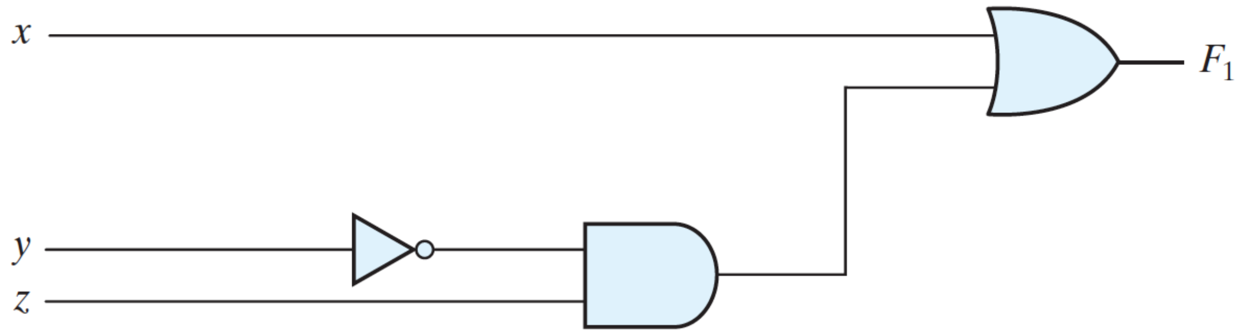
$$(xy)' = x' + y'$$

Boolean Functions

$$f : \{0,1\}^k \rightarrow \{0,1\}$$

x	y	z	F_1
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

$$F_1 = x + y'z$$



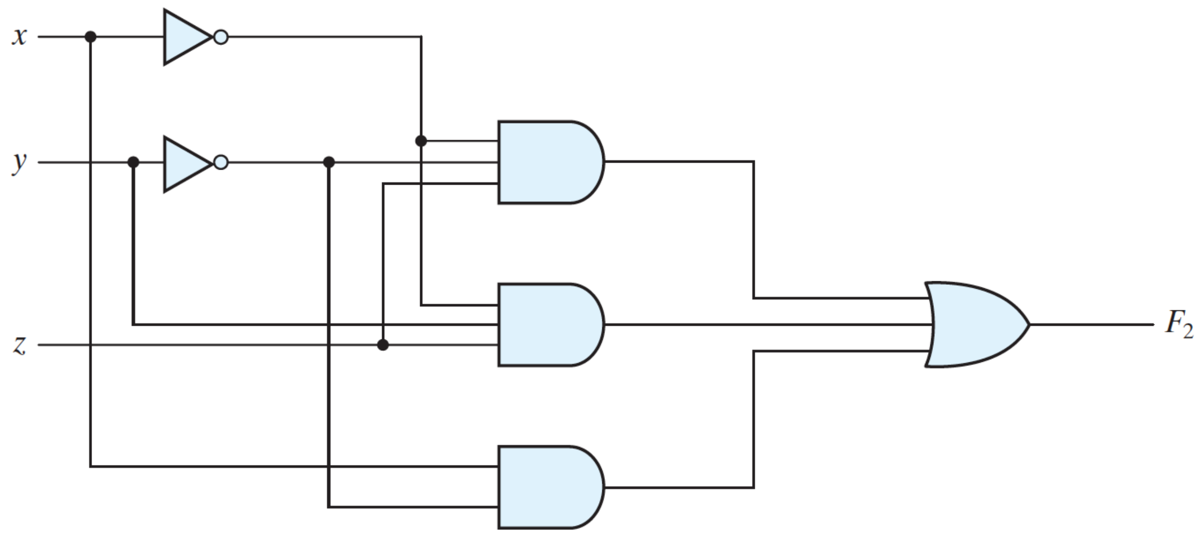
$$F_2 = x'y'z + x'yz + xy'$$

Boolean Functions

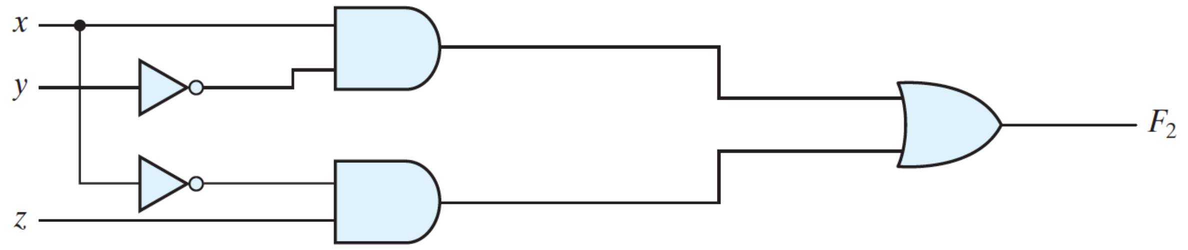
$$f : \{0,1\}^k \rightarrow \{0,1\}$$

$$F_2 = x'y'z + x'yz + xy'$$

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$



(a) $F_2 = x'y'z + x'yz + xy'$



(b) $F_2 = xy' + x'z$

Other Boolean Functions

Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Expressions for the 16 Functions of Two Variables

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Algebraic manipulation

Complement of a Function:

$$\begin{aligned}(A + B + C)' &= (A + x)' \\ &= A'x' \\ &= A'(B + C)' \\ &= A'(B'C') \\ &= A'B'C'\end{aligned}$$

$$(A + B + C + D + \cdots + F)' = A'B'C'D' \ldots F'$$

$$(ABCD \ldots F)' = A' + B' + C' + D' + \cdots + F'$$

Examples:

$$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$$

$$\begin{aligned}F_2' = [x(y'z' + yz)]' = x' + (y'z' + yz)' &= x' + (y'z')'(yz)' \\ &= x' + (y + z)(y' + z') \\ &= x' + yz' + y'z\end{aligned}$$

Algebraic manipulation

Examples:

$F_1 = x'yz' + x'y'z.$
The dual of F_1 is $(x' + y + z')(x' + y' + z).$
Complement each literal: $(x + y' + z)(x + y + z') = F_1'.$

$F_2 = x(y'z' + yz).$
The dual of F_2 is $x + (y' + z')(y + z).$
Complement each literal: $x' + (y + z)(y' + z') = F_2'.$

Canonical and Standard Forms

Minterms and Maxterms

Product (AND), Sum (OR)

- Each maxterm is obtained from an OR term of the n variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1
- Each maxterm is the complement of its corresponding minterm and vice versa.

			Minterms		Maxterms	
x	***y***	***z***	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Representation of Boolean Functions

Minterms: A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

Example:

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

- **Form a maxterm for each combination of the variables that produces a 0 in the function,**
- **Then form the AND of all those maxterms.**

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in canonical form

Sum of minterms

The minterms whose sum defines the Boolean function are those which give the 1's of the function in the truth table.

Product of maxterms

The maxterms whose product defines the Boolean function are those which give the 0's of the function in the truth table.

Sum of minterms representation: $F = A + B'C$

$$A = A(B + B') = AB + AB'$$

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

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

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \end{aligned}$$

$$F = A + B'C$$

$$\begin{aligned} F &= A'B'C + AB'C + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

Product of maxterms representation:

$$F = xy + x'z$$

$$\begin{aligned} F = xy + x'z &= (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0M_2M_4M_5 \end{aligned}$$

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

Conversion between minterm and maxterm forms

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

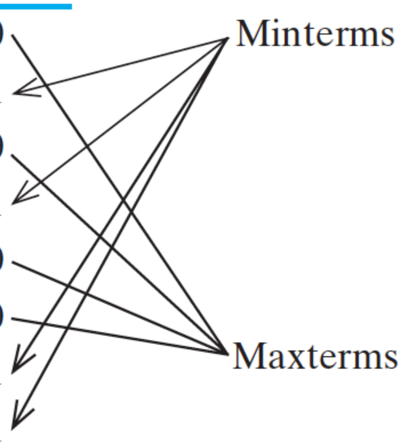
$$m_j' = M_j$$

$$F = xy + x'z$$

$$F(x, y, z) = \Sigma(1, 3, 6, 7)$$

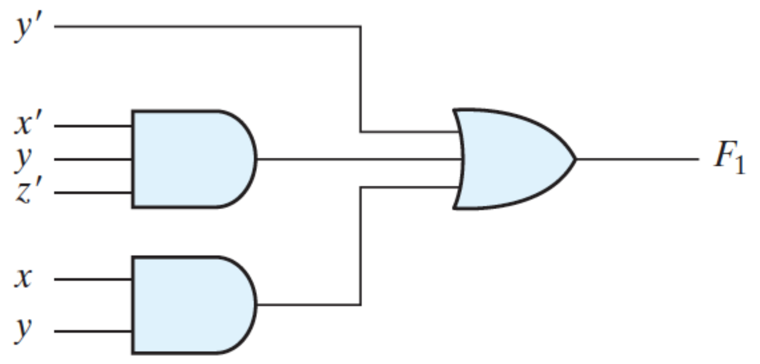
Truth Table for $F = xy + x'z$

x	y	z	F
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



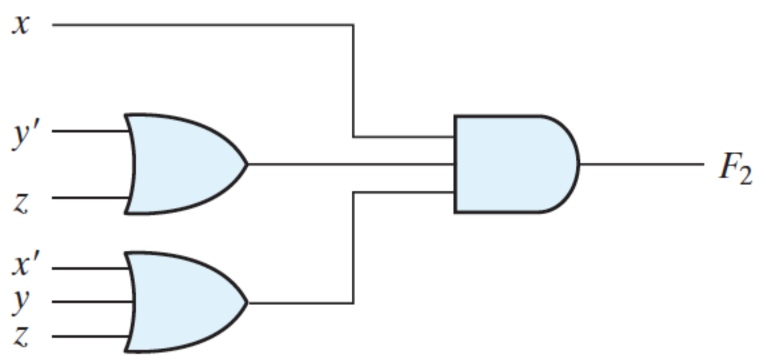
Standard Forms: SOPs and POSs

$$F_1 = y' + xy + x'yz'$$



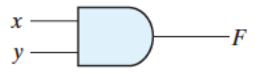
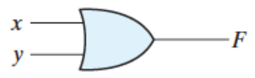
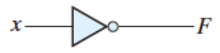
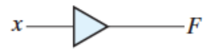
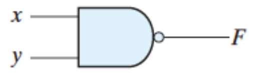
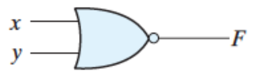
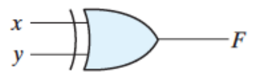
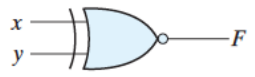
(a) Sum of Products

$$F_2 = x(y' + z)(x' + y + z')$$



(b) Product of Sums

Digital Logic Gates

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	$x\ y \mid F$ 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	$x\ y \mid F$ 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	$x \mid F$ 0 | 1 1 | 0
Buffer		$F = x$	$x \mid F$ 0 | 0 1 | 1
NAND		$F = (xy)'$	$x\ y \mid F$ 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	$x\ y \mid F$ 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y \mid F$ 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y \mid F$ 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Thank you