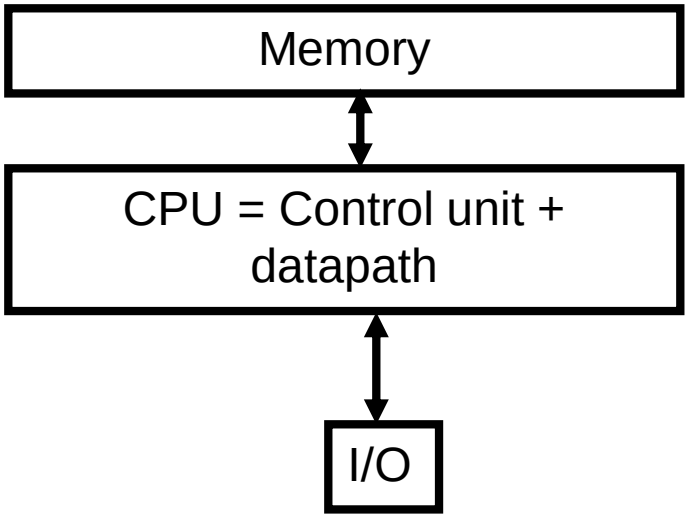
Digital Logic Design(D.L.D-LAB)

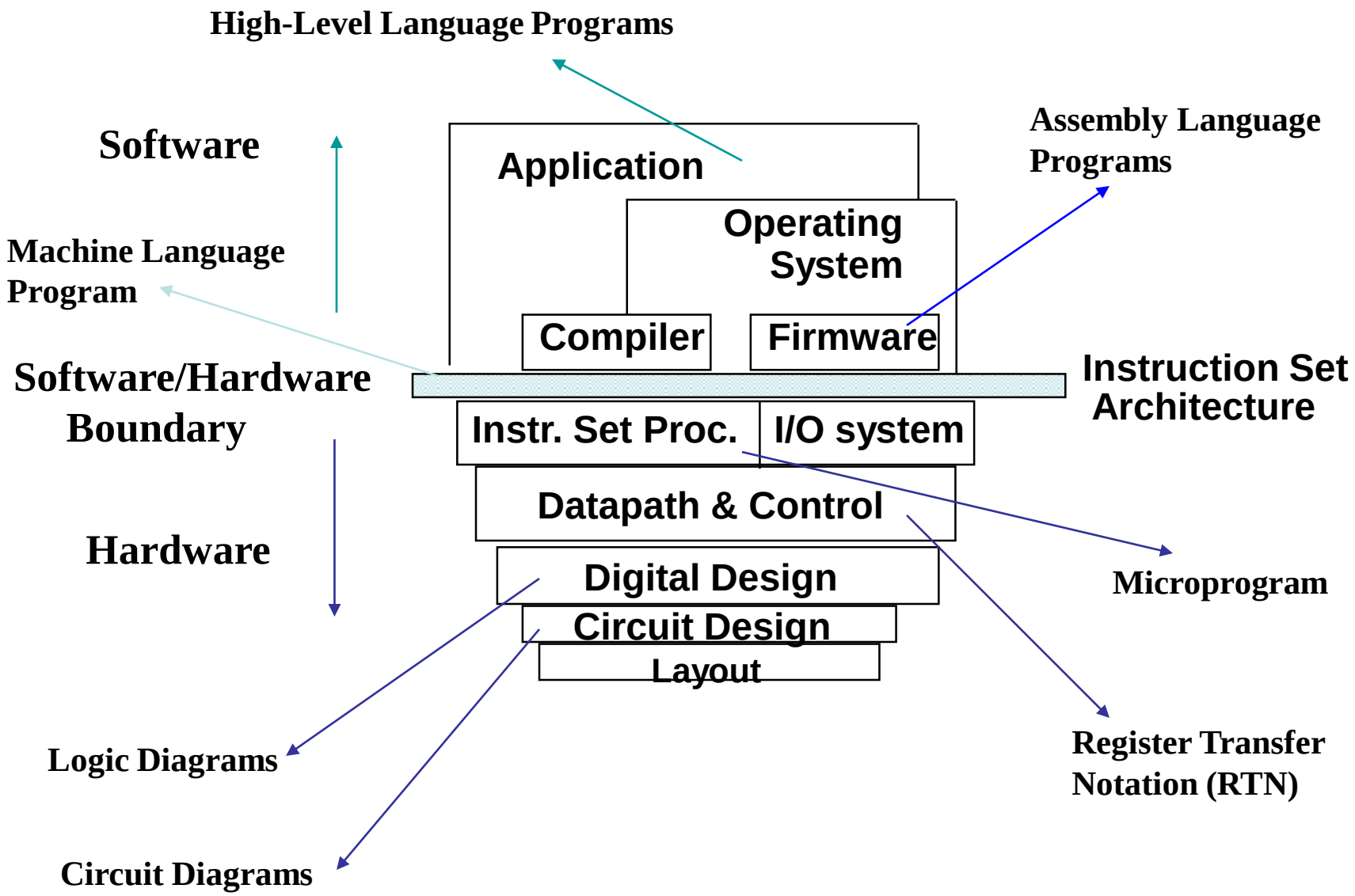
- An Overview of Computer Organization
- Switches and Transistors
- Boolean Algebra and Logic
- Binary Arithmetic and Number Systems
- Combinational Logic and Circuits
- Sequential Logic and Circuits
- Memory Logic Design
- The DataPathUnit

Basic Definitions

- Computer **Architecture** is the programmer's perspective on functional behavior of a computer (e.g., 32 bits to represent an integer value)
- Computer **organization** is the internal structural relationships not visible to a programmer…e.g., physical memory



Hierarchy of Computer Architecture



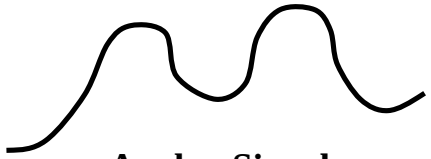
Basic Definitions

- Architectural levels: Programs and applications to transistors
- Electrical Signals: discrete, atomic elements of a digital system…binary values…

An ideal switch

Introduction to Digital Systems

- Analog devices and systems process time-varying signals that can take on any value across a continuous range.



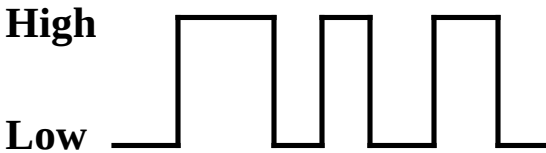
Analog Signal

- Digital systems use digital circuits that process digital signals which can take on one of two values, we call:

  0 and 1 (digits of the binary number system)

  or LOW and HIGH

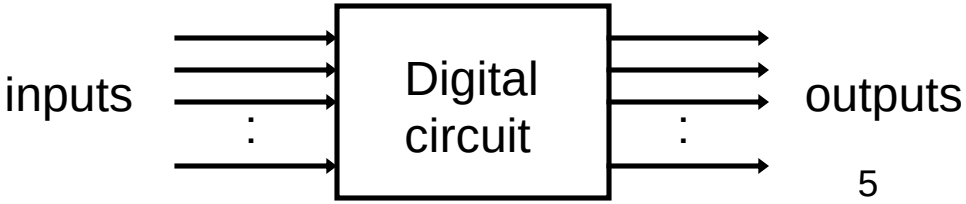
  or FALSE and TRUE



Digital Signal

- Digital computers represent the most common digital systems.
- Once-analog Systems that use digital systems today:

  - **Audio recording (CDs, DAT, mp3)**
  - **Phone system switching**
  - **Automobile engine control**
  - **Movie effects**
  - **Still and video cameras….**



Eight Advantages of Digital Systems Over Analog Systems

1. Reproducibility of the results
2. Accuracy of results
3. More reliable than analog systems due to better immunity to noise.
4. Ease of design: No special math skills needed to visualize the behavior of small digital (logic) circuits.
5. Flexibility and functionality.
6. Programmability.
7. Speed: A digital logic element can produce an output in less than 10 nanoseconds (10^{-8} seconds).
8. Economy: Due to the integration of millions of digital logic elements on a single miniature chip forming low cost integrated circuit (ICs).

Boolean Algebra

What is an Algebra? (e.g. algebra of integers)

set of elements (e.g. 0,1,2,..)
set of operations (e.g. +, -, *,..)
postulates/axioms (e.g. 0+x=x,..)

- Boolean Algebra named after George Boole who used it to study human logical reasoning – calculus of proposition.
- Elements : *true* or *false* *(0, 1)*
- Operations: a *OR* b; a *AND* b, *NOT* a

e.g. 0 OR 1 = 1 0 OR 0 = 0

1 AND 1 = 1 1 AND 0 = 0

NOT 0 = 1 NOT 1 = 0

Digital (logic) Elements: Gates

- Digital devices or gates have one or more inputs and produce an output that is a function of the current input value(s).
- All inputs and outputs are binary and can only take the values 0 or 1
- A gate is called a *combinational circuit* because the output only depends on the current input combination.
- Digital circuits are created by using a number of connected gates such as the output of a gate is connected to to the input of one or more gates in such a way to achieve specific outputs for input values.
- Digital or logic design is concerned with the design of such circuits.

Boolean Algebra

- Set of Elements: {0,1}
- Set of Operations: {., + , ¬ }

x	y	x . y
0	0	0
0	1	0
1	0	0
1	1	1

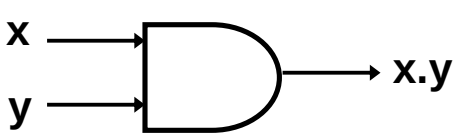
AND

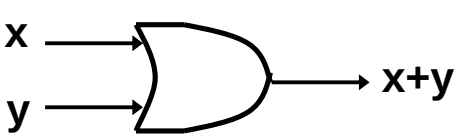
x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

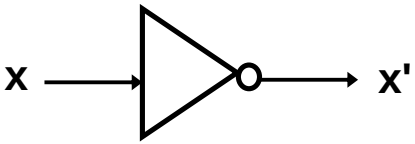
OR

x	¬x
0	1
1	0

NOT

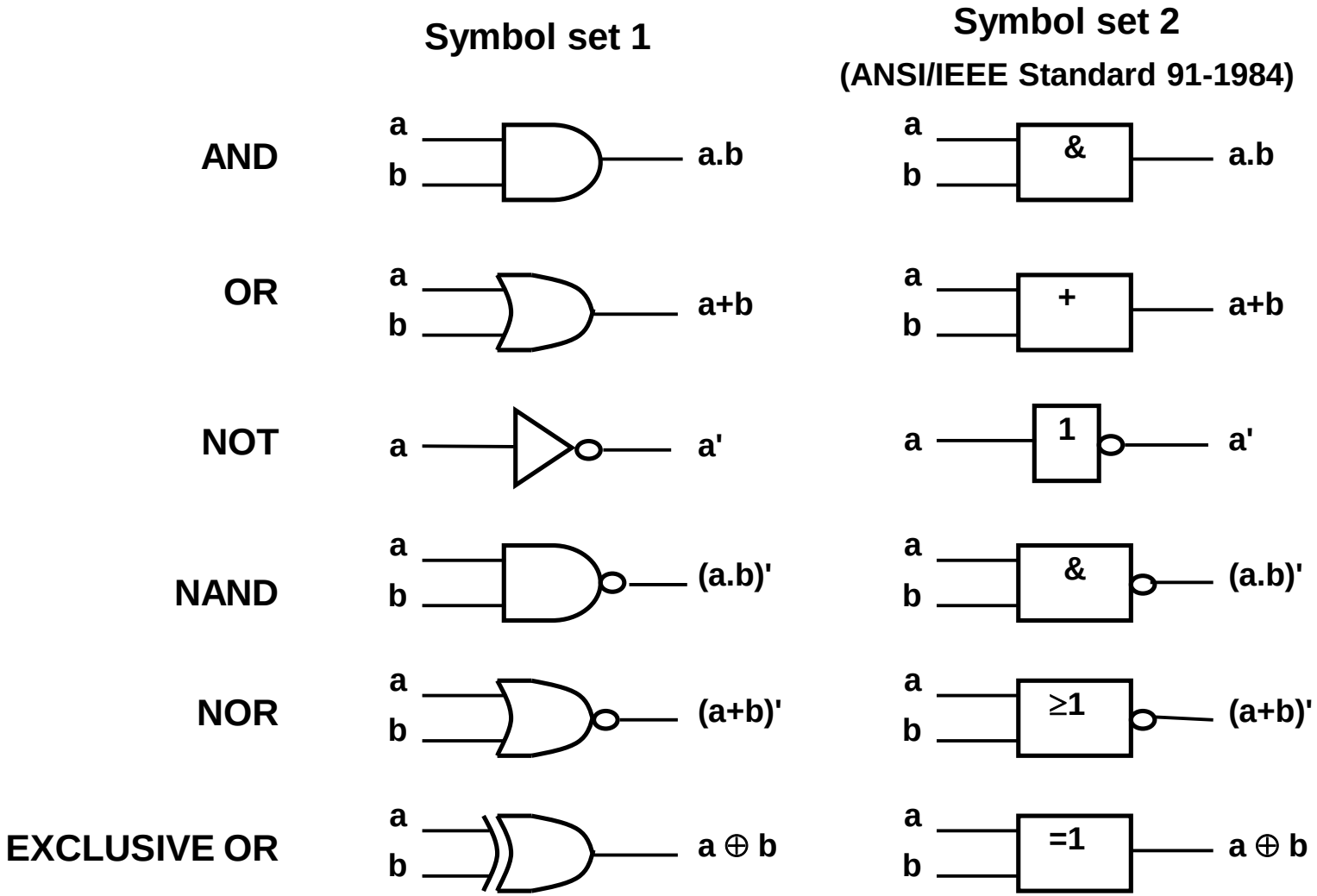






Signals: High = 5V = 1; Low = 0V = 0

Logic Gates


Symbol set 1
Symbol set 2
(ANSI/IEEE Standard 91-1984)
AND
a
b
a.b
&
OR
a+b
+
NOT
a'
1
NAND
(a.b)'
NOR
(a+b)'
≥1
EXCLUSIVE OR
a ⊕ b
=1

Truth Tables

- Provide a listing of every possible combination of values of binary inputs to a digital circuit and the corresponding outputs.

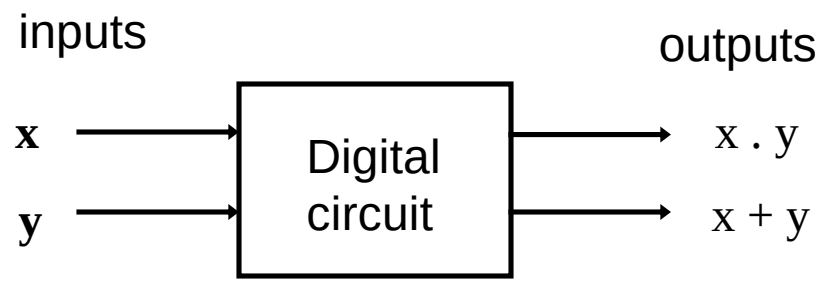
INPUTS	OUTPUTS
…	…
…	…

Truth table

inputs outputs

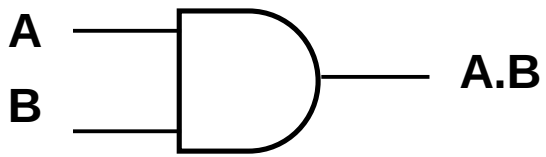
x	y	x . y	x + y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

- Example (2 inputs, 2 outputs):



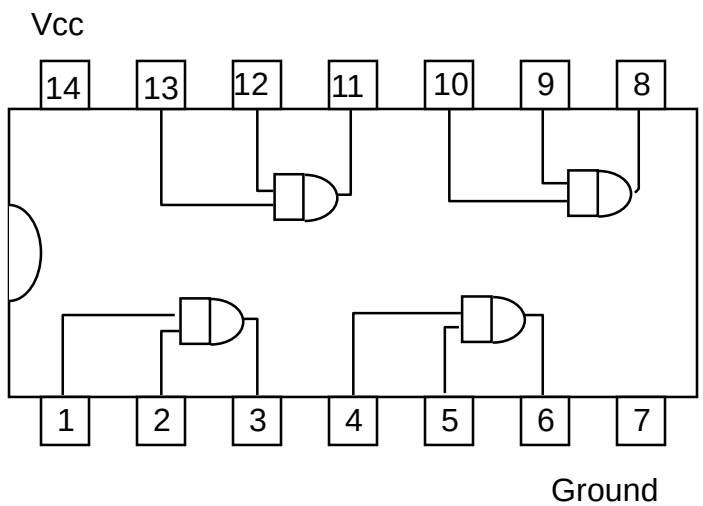
Logic Gates: The AND Gate

- The AND Gate



A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

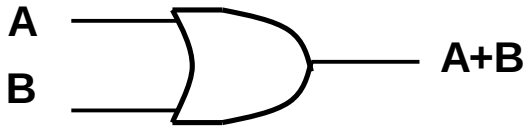
Truth table



Top View of a TTL 74LS family 74LS08 Quad 2-input AND Gate IC Package

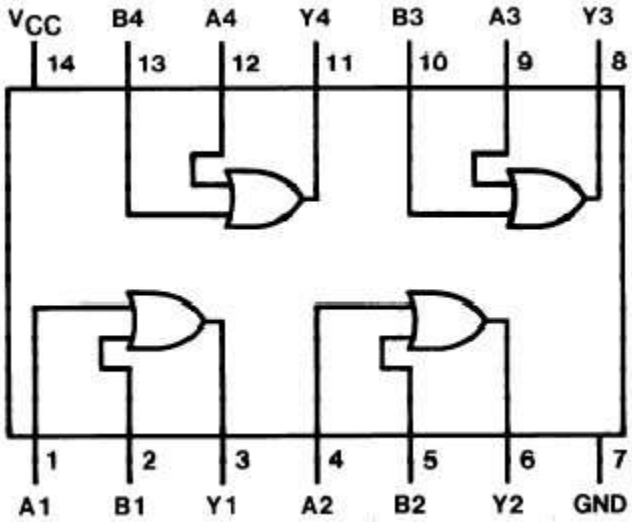
Logic Gates: The OR Gate

- The OR Gate



A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

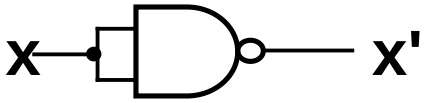


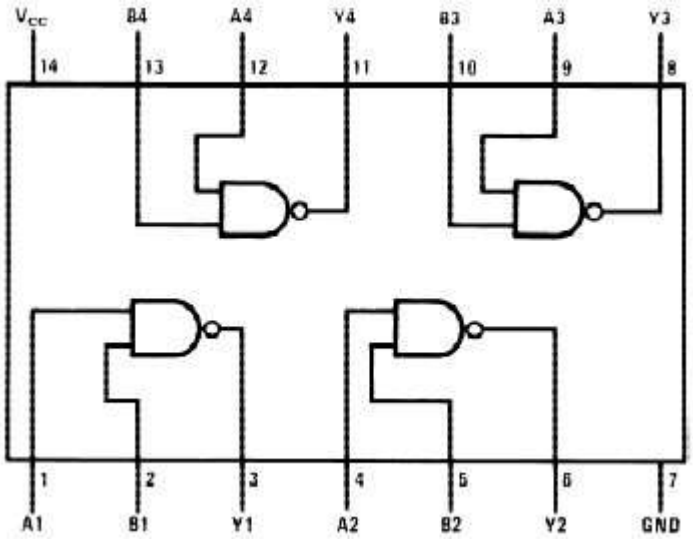
Top View of a TTL 74LS family 74LS08 Quad 2-input OR Gate IC Package

Logic Gates: The NAND Gate

- The NAND Gate

- NAND gate is self-sufficient (can build any logic circuit with it).
- Can be used to implement AND/OR/NOT.
- Implementing an inverter using NAND gate:





Top View of a TTL 74LS family 74LS00 Quad 2-input NAND Gate IC Package

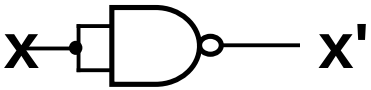
A	B	(A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

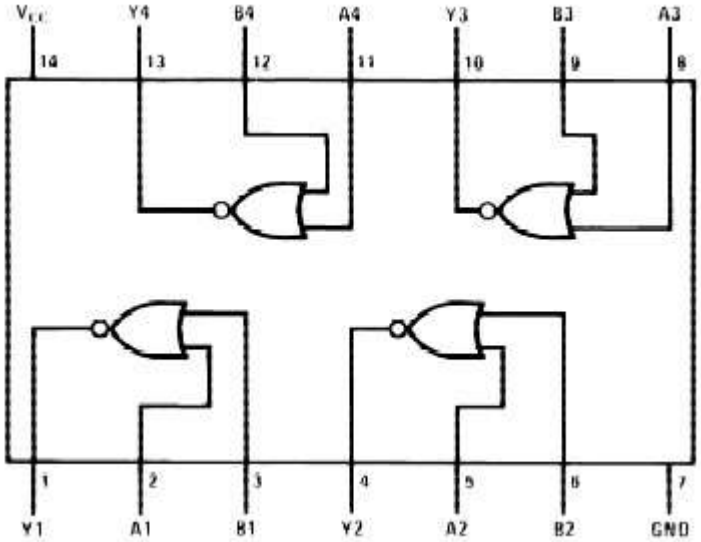
Truth table

Logic Gates: The NOR Gate

- The NOR Gate

- NOR gate is also self-sufficient (can build any logic circuit with it).
- Can be used to implement AND/OR/NOT.
- Implementing an inverter using NOR gate:





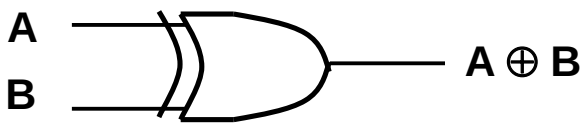
A	B	(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

Top View of a TTL 74LS family 74LS02 Quad 2-input NOR Gate IC Package

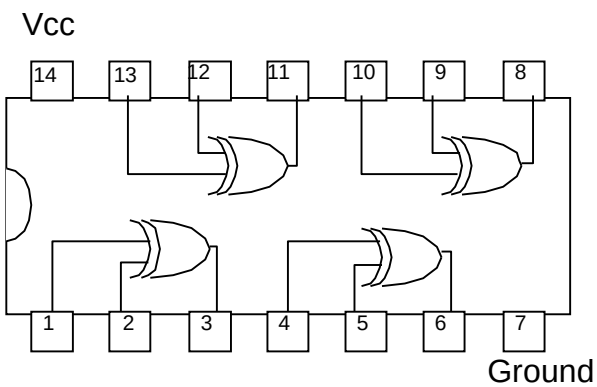
Logic Gates: The XOR Gate

- The XOR Gate



A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Truth table



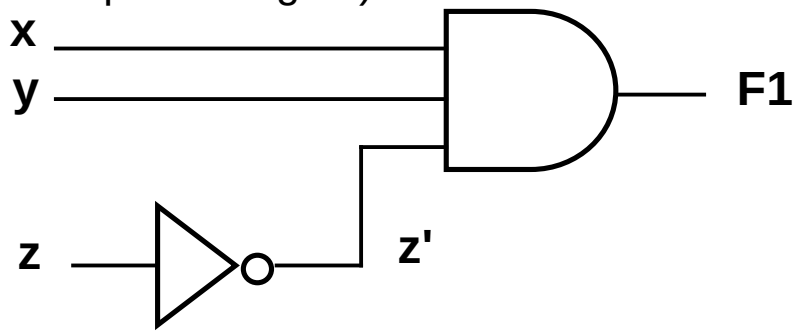
Top View of a TTL 74LS family 74LS86 Quad 2-input XOR Gate IC Package

Drawing Logic Circuits

- When a Boolean expression is provided, we can easily draw the logic circuit.
- Examples:

$$F1 = xyz'$$

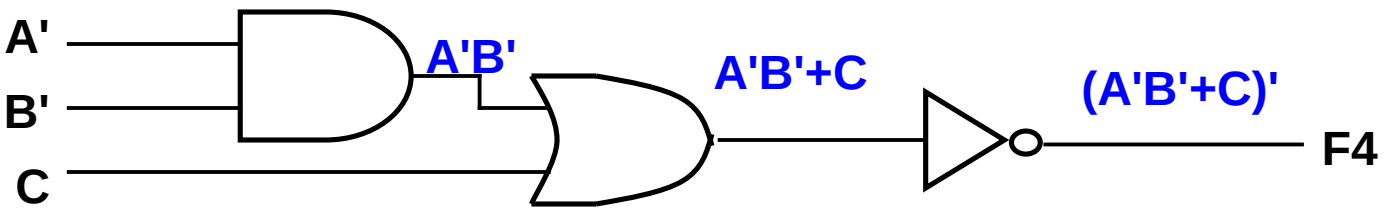
(note the use of a 3-input AND gate)



Analyzing Logic Circuits

- When a logic circuit is provided, we can analyze the circuit to obtain the logic expression.
- Example: What is the Boolean expression of F4?

F4 = (A'B'+C)'



Integrated Circuits

- An Integrated circuit (IC) is a number of logic gated fabricated on a single silicon chip.
- ICs can be classified according to how many gates they contain as follows:
  - **Small-Scale Integration (SSI):** Contain 1 to 20 gates.
  - **Medium-Scale Integration (MSI):** Contain 20 to 200 gates. Examples: Registers, decoders, counters.
  - **Large-Scale Integration (LSI):** Contain 200 to 200,000 gates. Include small memories, some microprocessors, programmable logic devices.
  - **Very Large-Scale Integration (VLSI):** Usually stated in terms of number of transistors contained usually over 1,000,000. Includes most microprocessors and memories.

Computer Hardware Generations

- The First Generation, 1946-59: Vacuum Tubes, Relays, Mercury Delay Lines:
  - **ENIAC (Electronic Numerical Integrator and Computer): First electronic computer, 18000 vacuum tubes, 1500 relays, 5000 additions/sec.**
  - **First stored program computer: EDSAC (Electronic Delay Storage Automatic Calculator).**
- The Second Generation, 1959-64: Discrete Transistors.

  (e.g **IBM 7000 series, DEC PDP-1)**
- The Third Generation, 1964-75: Small and Medium-Scale Integrated (SSI, MSI) Circuits. **(e.g. IBM 360 mainframe)**
- The Fourth Generation, 1975-Present: The Microcomputer. VLSI-based Microprocessors.

Intentionally left blank

Positional Number Systems

- A number system consists of an order set of symbols (digits) with relations defined for +,-,*, /
- The radix (or base) of the number system is the total number of digits allowed in the the number system.
  - **Example, for the decimal number system:**
    - Radix, r = 10, Digits allowed = 0,1, 2, 3, 4, 5, 6, 7, 8, 9
- In positional number systems, a number is represented by a string of digits, where each digit position has an associated weight.
- The value of a number is the weighted sum of the digits.
- The general representation of an unsigned number D with whole and fraction portions number in a number system with radix r:

$$D_r = d_{p-1}\, d_{p-2} \ldots\ldots d_1\, d_0.d_{-1}\, d_{-2} \ldots\ldots D_{-n}$$

- The number above has p digits to the left of the radix point and n fraction digits to the right.
- A digit in position i has as associated weight r^i
- The value of the number is the sum of the digits multiplied by the associated weight r^i :

$$D = \sum_{i=-n}^{p-1} d_i \times r^i$$

Number Systems Used in Computers

Name of Radix	Radix	Set of Digits	Example
Decimal	r=10	{0,1,2,3,4,5,6,7,8,9}	255_{10}
Binary	r=2	{0,1}	11111111_2
Octal	r= 8	{0,1,2,3,4,5,6,7}	377_8
Hexadecimal	r=16	{0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F}	FF_{16}

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Binary numbers

- a bit: a binary digit representing a 0 or a 1.
- Binary numbers are base 2 as opposed to base 10 typically used.
- Instead of decimal places such as 1s, 10s, 100s, 1000s, etc., binary uses powers of two to have 1s, 2s, 4s, 8s, 16s, 32s, 64s, etc.

$\mathbf{101}_2 = (\mathbf{1}\times\mathbf{2}^2)+(\mathbf{0}\times\mathbf{2}^1)+(\mathbf{1}\times\mathbf{2}^0) = \mathbf{4}10 + \mathbf{1}10 = \mathbf{5}_{10}$

$\mathbf{10111}_2 = (\mathbf{1}\times\mathbf{2}^4)+(\mathbf{0}\times\mathbf{2}^3)+(\mathbf{1}\times\mathbf{2}^2)+(\mathbf{1}\times\mathbf{2}^1)+(\mathbf{1}\times\mathbf{2}^0) = \mathbf{23}_{10}$

41_{10} = 41/2 + remainder = 1→1LSB
= 20/2 + remainder = 0 →2SB
= 10/2 + remainder = 0 →3SB
= 5/2 + remainder = 1 → 4SB
= 4/2 + remainder = 0 → 5SB
= 2/2 = 1 → 6SB
101001_2

Largest numbers

- the largest number of d digits in base R is

$R^d - 1$

Examples:

3 digits of base 10: $10^3 - 1 = 999$

2 digits of base 16: $16^2 - 1 = 255$

Decimal-to-Binary Conversion

- Separate the decimal number into whole and fraction portions.
- To convert the whole number portion to binary, use successive division by 2 until the quotient is 0. The remainders form the answer, with the first remainder as the *least significant bit (LSB)* and the last as the *most significant bit (MSB)*.
- Example: Convert 179_{10} to binary:

179 / 2 = 89 remainder 1 (LSB)
/ 2 = 44 remainder 1
/ 2 = 22 remainder 0
/ 2 = 11 remainder 0
/ 2 = 5 remainder 1
/ 2 = 2 remainder 1
/ 2 = 1 remainder 0
/ 2 = 0 remainder 1 (MSB)

$$179_{10} = 10110011_2$$

Decimal-to-Binary examples

108/2 = 54
54 * 2 = 108, remainder 0
54 /2 = 27
27 * 2 = 54, remainder 0
27/2 = 13.5
13 * 2 = 26, remainder 1
13 /2 = 6.5
6 * 2 = 12, remainder 1
6/2 = 3
3 * 2 = 6, remainder 0
3/2 = 1
1 * 2 = 2, remainder 1
1101100_2

90/2 = 45
45 * 2 = 90, remainder 0
45/2 = 22.5
22 * 2 = 44, remainder 1
22 * 2 = 44, remainder 0
22/2 = 11
11 * 2 = 22, remainder 0
11/2 = 5.5
5 * 2 = 10, remainder 1
5/2 = 2.5
2 * 2 = 4, remainder 1
2/2 = 1
1 * 2 = 2, remainder 0
1 / 2 = 0
0 * 2 = 0, remainder 1
1011010_2

7/2 = 3.5
3 * 2 = 6, remainder 1
3/2 = 1
1 * 2 = 2, remainder 1
1/2 = 0
0 * 2 = 0, remainder 1
111_2

11/2 = 5.5
5 * 2 = 10, remainder 1
5/2 = 2.5
2 * 2 = 4, remainder 1
2/2 = 1
1 * 2 = 2, remainder 0
1 / 2 = 0
0 * 2 = 0, remainder 1
1011_2

Decimal-to-Hex examples

108/16 = 6.75
6 * 16 = 96, remainder 12
6 /16 = 0
0 * 16 = 0, remainder 6
$6C_{16}$

160/16 = 10
10 * 16 = 160, remainder 0
10/16 = 0
0 * 16 = 0, remainder 10
$A0_{16}$

20/16 = 1
1 * 16 = 16, remainder 4
1/16 = 0
0 * 16 = 0, remainder 1
14_{16}

32/16 = 2
2 * 16 = 32, remainder 0
2 /16 = 0
0 * 16 = 0, remainder 2
20_{16}

90/16 = 5.625
5 * 16 = 80, remainder 10
5 / 16 = 0
0 * 16 = 0, remainder 5
$5A_{16}$

Decimal-to-Octal example

108/8 = 13.5
13 * 8 = 104, remainder 4
13/8 = 1
1 * 8 = 8, remainder 5
1 / 8 = 0
0 * 8 = 0, remainder 1
154_8

24/8 = 3
3 * 8 = 24, remainder 0
3/8 = 0
0 * 8 = 0, remainder 3
30_8

10/8 = 1
1 * 8 = 8, remainder 2
1/8 = 0
0 * 8 = 0, remainder 1
12_8

16/8 = 2
2 * 8 = 16, remainder 0
2/8 = 0
0 * 8 = 0, remainder 2
20_8

Decimal-to-Binary Conversion

- To convert decimal fractions to binary, repeated multiplication by 2 is used, until the fractional product is 0 (or until the desired number of binary places). The whole digits of the multiplication results produce the answer, with the first as the MSB, and the last as the LSB.
- Example: Convert 0.3125_{10} to binary

			Result Digit	
$.3125 \times 2$	=	0.625	0	(MSB)
$.625 \times 2$	=	1.25	1	
$.25 \times 2$	=	0.50	0	
$.5 \times 2$	=	1.0	1	(LSB)

$0.3125_{10} = .0101_2$

Binary Arithmetic Operations - Addition

- Similar to decimal number addition, two binary numbers are added by adding each pair of bits together with carry propagation.
- Addition Example:

			1	0	1	1	1	1	0	0	0	Carry
X	190			1	0	1	1	1	1	1	0	
Y	+ 141	+		1	0	0	0	1	1	0	1	
X + Y	331		1	0	1	0	0	1	0	1	1	

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 with a carry of 1

Binary Arithmetic- subtraction

95 = 1011111
-16 = 0010000
79 = 1001111

0 – 0 = 0
1 – 0 = 1
1 – 1 = 0
0 – 1 = 1 with a borrow of 1

Binary Arithmetic Operations: Subtraction

- Two binary numbers are subtracted by subtracting each pair of bits together with borrowing, where needed.
- Subtraction Example:

			0	0	1	1	1	1	1	0	0	Borrow
X	229			1	1	1	0	0	1	0	1	
Y -	46	-		0	0	1	0	1	1	1	0	
	183			1	0	1	1	0	1	1	1	

Binary Arithmetic - Multiplication

```
   1011
   *101
  ------
   1011
  0000
 1011
 ------
 110111
```

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

Negative Binary Number Representations

- Signed-Magnitude Representation:
  - **For an *n-bit* binary number:**

    Use the first bit (most significant bit, MSB) position to represent the sign where 0 is positive and 1 is negative.

    Ex. $\underset{\text{Sign}}{1}\ \underset{\text{Magnitude}}{1\ 1\ 1\ 1\ 1\ 1\ 1}_2 = -127_{10}$
  - **Remaining n-1 bits represent the magnitude which may range from:**

$$-2^{(n-1)} + 1 \quad \text{to} \quad 2^{(n-1)} - 1$$

  - **This scheme has two representations for 0; i.e., both positive and negative 0: for 8 bits: 00000000, 10000000**
  - **Arithmetic under this scheme uses the sign bit to indicate the nature of the operation and the sign of the result, but the sign bit is not used as part of the arithmetic.**

Parity bit

- Pad an extra bit to MSB side to make the number of 1's to be even or odd.
- Sender and receiver of messages make sure that even/odd transmission patterns match

Gray codes

- In binary codes, number of bit changes are not constant,

000→001→010→011→100→101→110→111→1000…

- bit changes in gray codes are constant

•000→001→011→010→110→111→000…

Alphanumeric Binary Codes: ASCII

Seven bit codes are used to represent all upper and lower case letters, numbers, punctuation and control characters

	MSBs							
LSBs	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC_1	!	1	A	Q	a	q
0010	STX	DC_2	“	2	B	R	b	r
0011	ETX	DC_3	#	3	C	S	c	s
0100	EOT	DC_4	$	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	O	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL