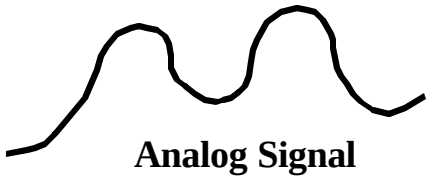
Introduction to Digital Systems

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



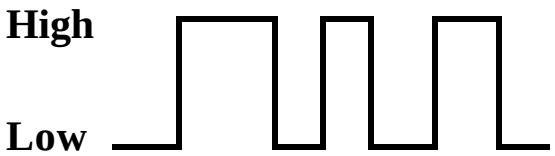
Analog Signal

- **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)

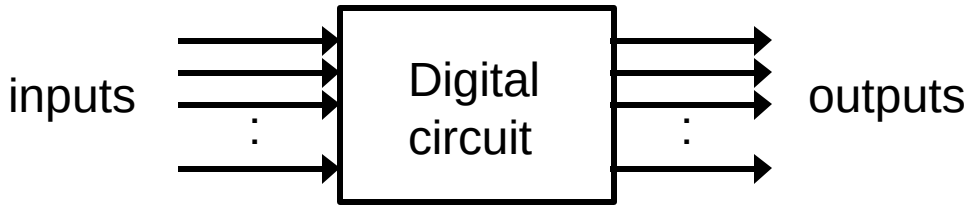
  or LOW and HIGH

  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….**



Advantages of Digital Systems Over Analog Systems

- **Reproducibility of the results and accuracy.**
- **More reliable than analog systems due to better immunity to noise.**
- **Ease of design: No special math skills needed to visualize the behavior of small digital (logic) circuits.**
- **Flexibility and functionality.**
- **Programmability.**
- **Speed: A digital logic element can produce an output in less than 10 nanoseconds (10^{-8} seconds).**
- **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**

Boolean Algebra

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

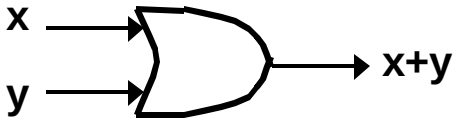
Sometimes denoted by ’, for example a’

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

AND

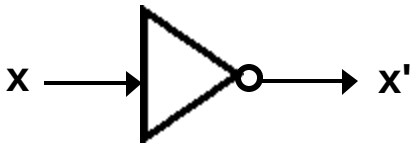
x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

OR



x	¬x
0	1
1	0

NOT

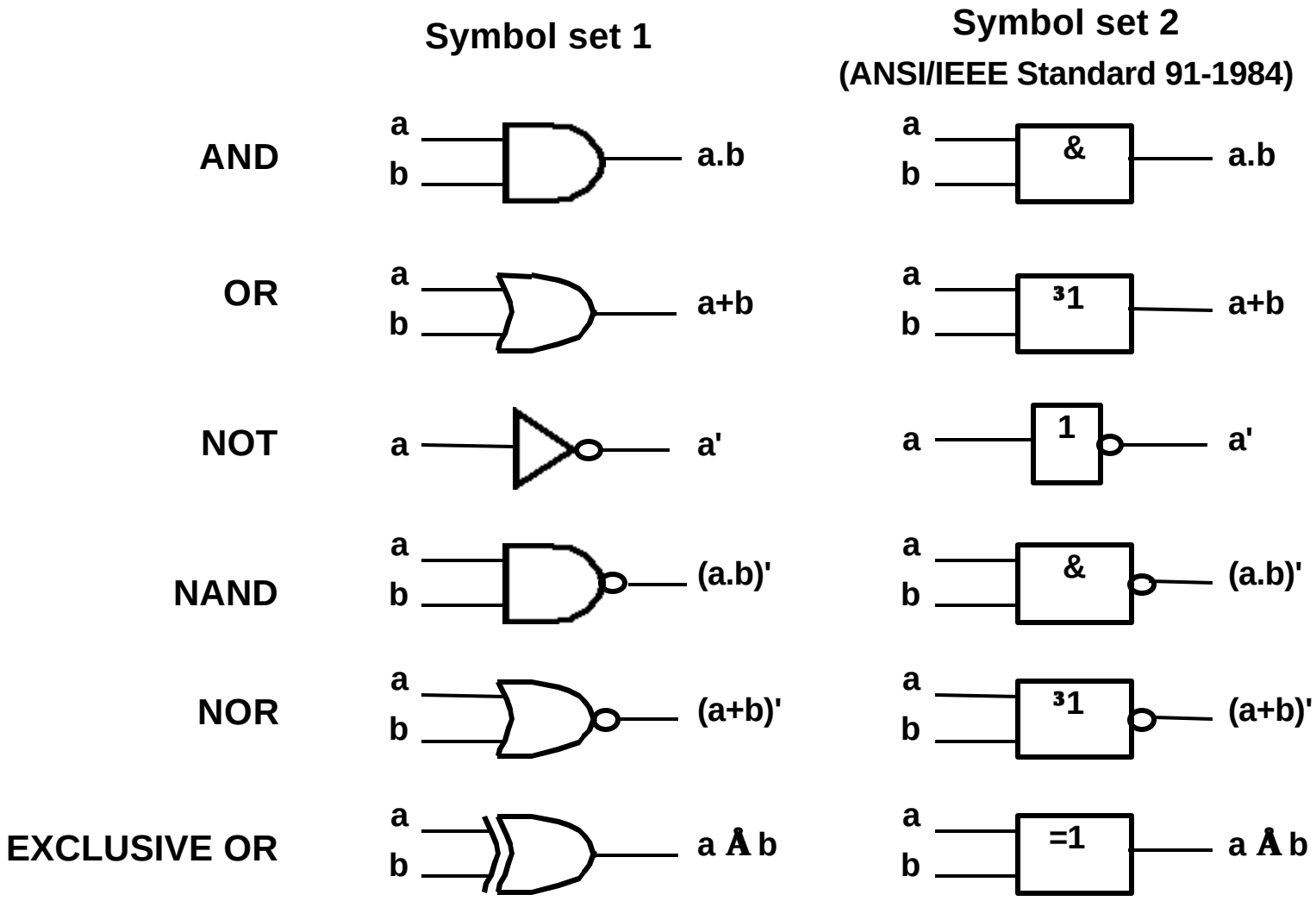


Signals: High = 5V = 1; Low = 0V = 0

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.**

Logic Gates


Symbol set 1
Symbol set 2
(ANSI/IEEE Standard 91-1984)
AND
a
b
a.b
&
OR
a+b
³1
NOT
a'
1
NAND
(a.b)'
NOR
(a+b)'
EXCLUSIVE OR
a Å b
=1

Truth Tables

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

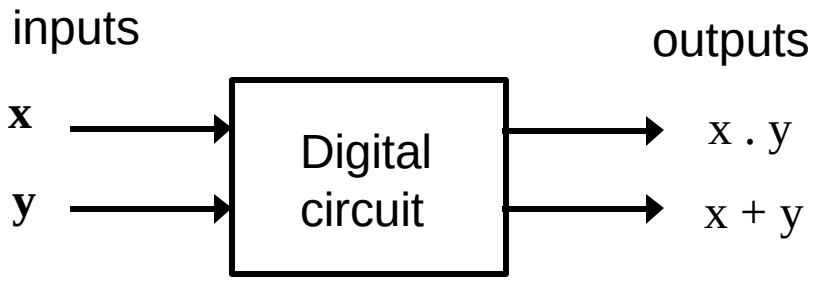
INPUTS	OUTPUTS
…	…
…	…

- **Example (2 inputs, 2 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

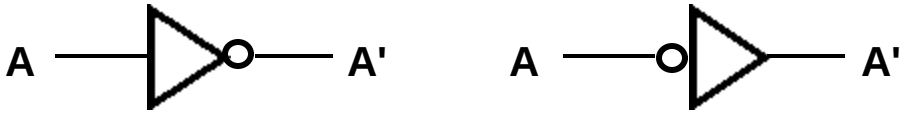


Realizing Logic in Hardware

- **Boolean Algebra and truth tables are essential important tools to express logical relationships.**
- **To use these tools in the real world , we must have some physical way to represent TRUE and FALSE (T and F).**
- **In, digital electronic circuits, T and F are represented by voltage levels:**
  - **The transistor-transistor logic (TTL) 74LS family of digital integrated circuits produces two voltage levels:**
    - **< .5V which represents low voltage L (0) and,**
    - **> 2.7V which represents high voltage H (1) for the digital device.**

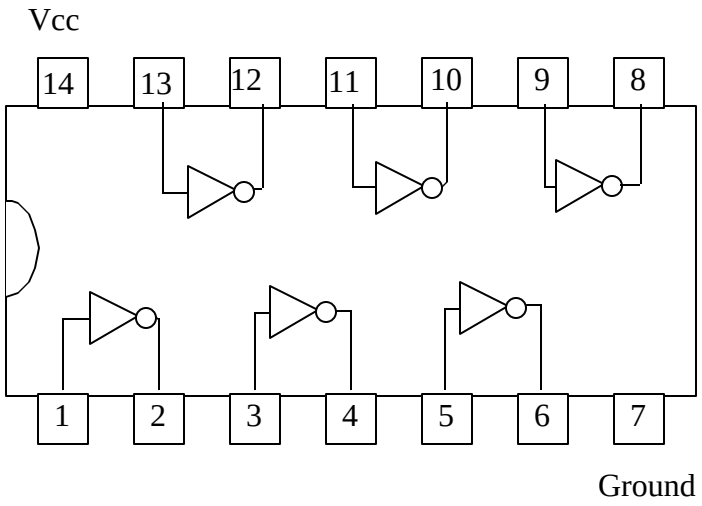
Logic Gates: The Inverter

- **The Inverter**



A	A'
0	1
1	0

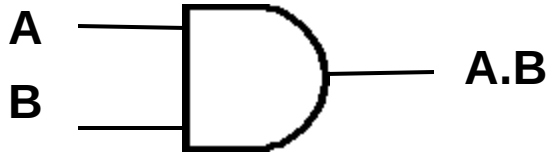
Truth table



Top View of a TTL 74LS family 74LS04 Hex Inverter IC Package

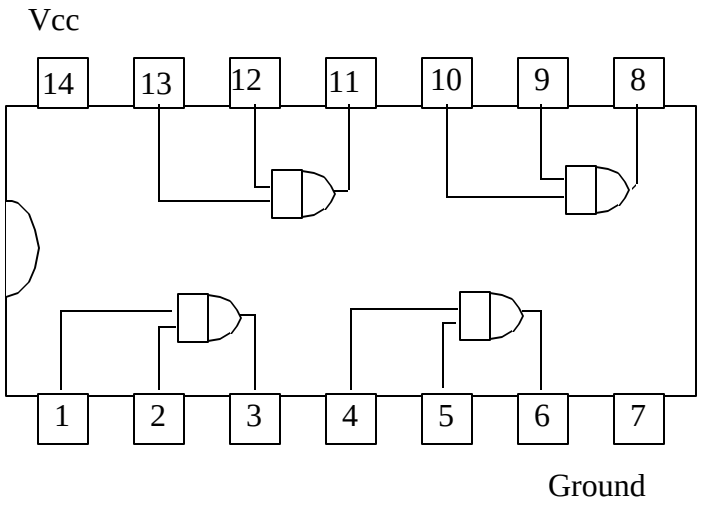
Logic Gates: The AND Gate

- **The AND Gate**



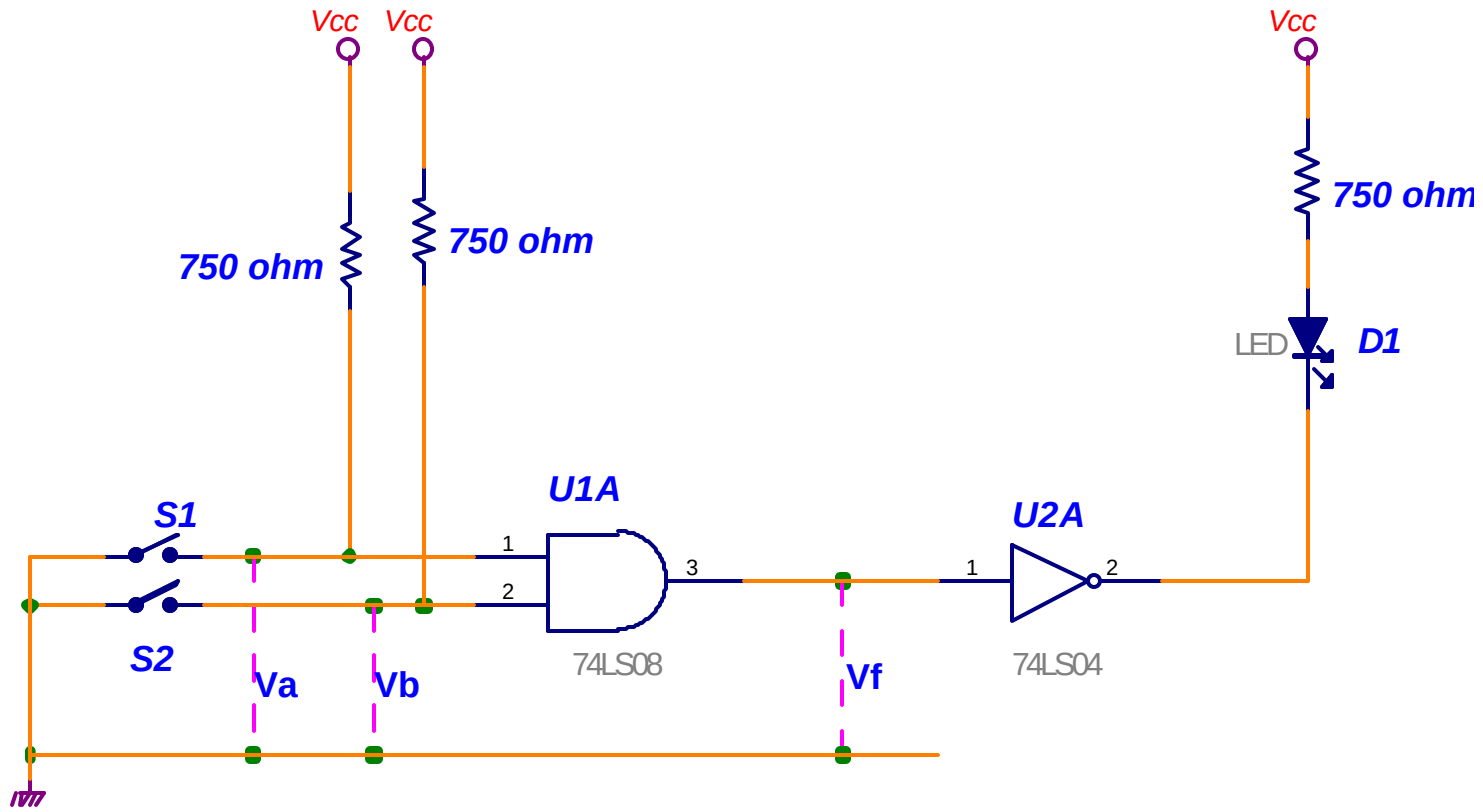
A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

Truth table



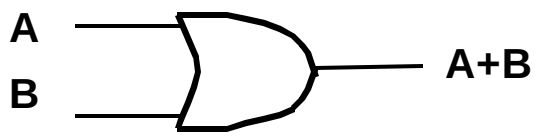
Top View of a TTL 74LS family 74LS08 Quad 2-input AND Gate IC Package

Circuit to determine AND Gate Truth Table (From Lab 1)



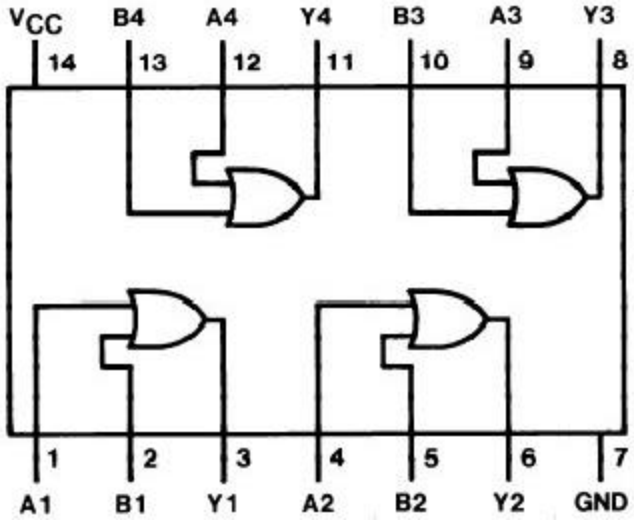
Logic Gates: The OR Gate

- **The OR Gate**



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

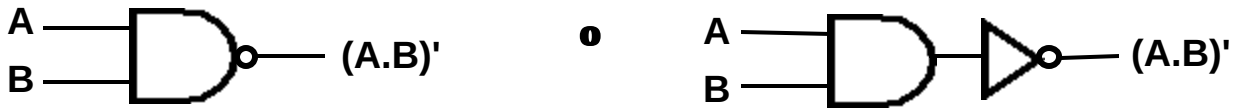
Truth table



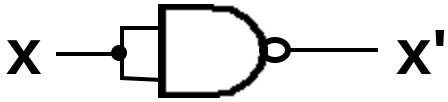
Top View of a TTL 74LS family 74LS08 Quad 2-input OR Gate IC Package

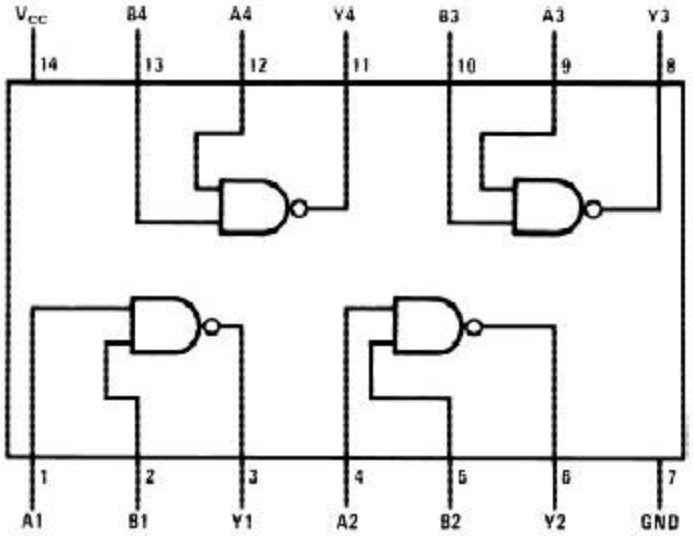
Logic Gates: The NAND Gate

- 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:





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

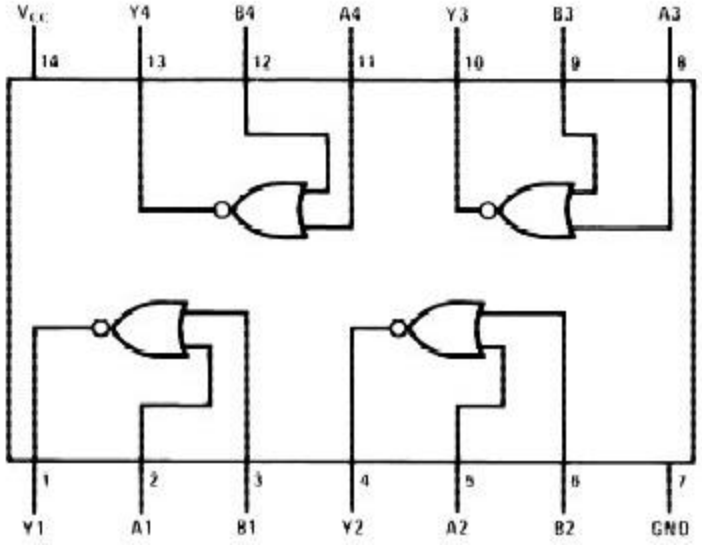
Truth table

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

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:



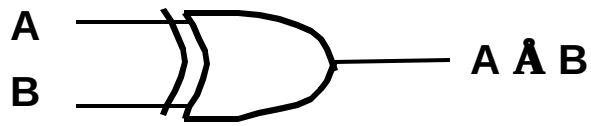
A	B	(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

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

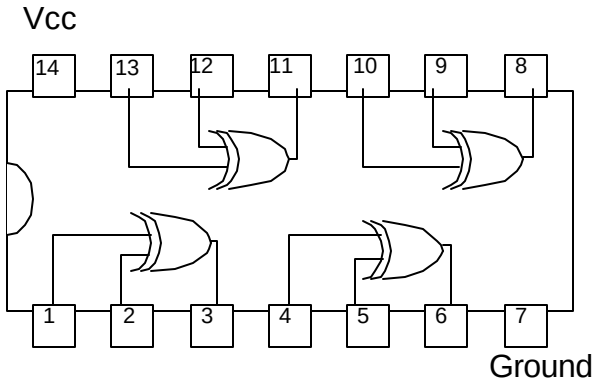
Logic Gates: The XOR Gate

- **The XOR Gate**



A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Truth table



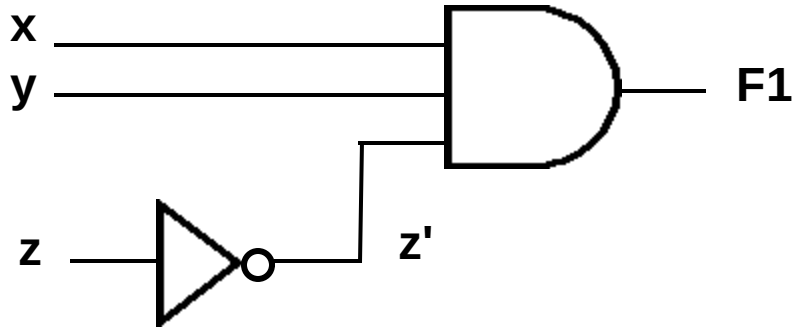
Top View of a TTL 74LS family 74LS86 Quad 2-input XOR Gate IC Package

Drawing Logic Circuits

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

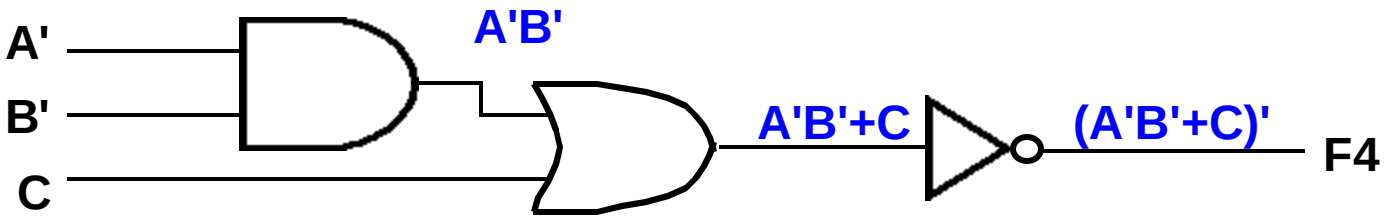
$$F1 = xyz'$$

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



Analysing Logic Circuits

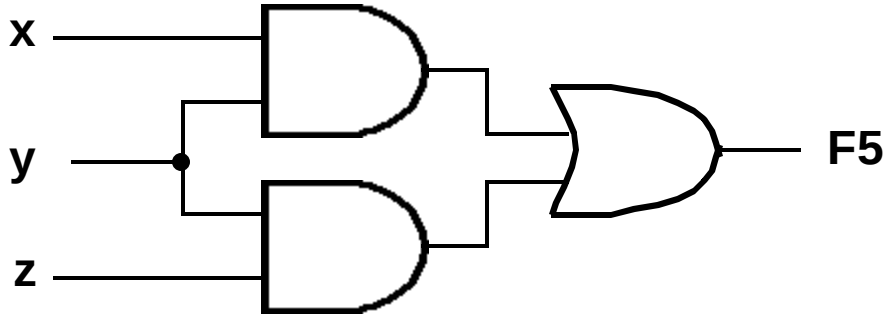
- When a logic circuit is provided, we can analyse the circuit to obtain the logic expression.
- Example: What is the Boolean expression of F4?



F4 = (A'B'+C)'

Analysing Logic Circuit

- **Example: What is Boolean expression of F5?**



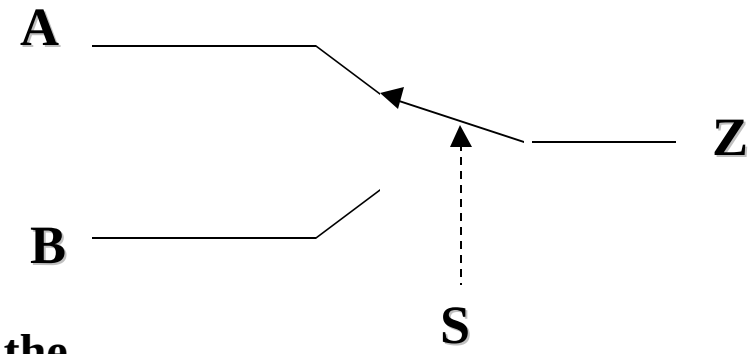
F5 =

Simple Circuit Design: Two-input Multiplexer

- **Multiplexer with two input bits, A, B and a control input bit S and output Z. Depending on the value of S, the circuit is to transfer either the the value of A or B to the output Z**

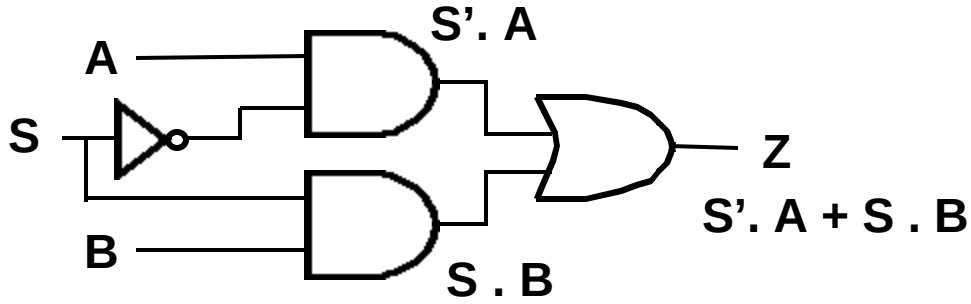
Truth table from circuit description

S	A	B	Z
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



Using logic design methods (to be studied later) we get the optimal logic function for Z

$$Z = S'.A + S.B$$



Integrated Circuits

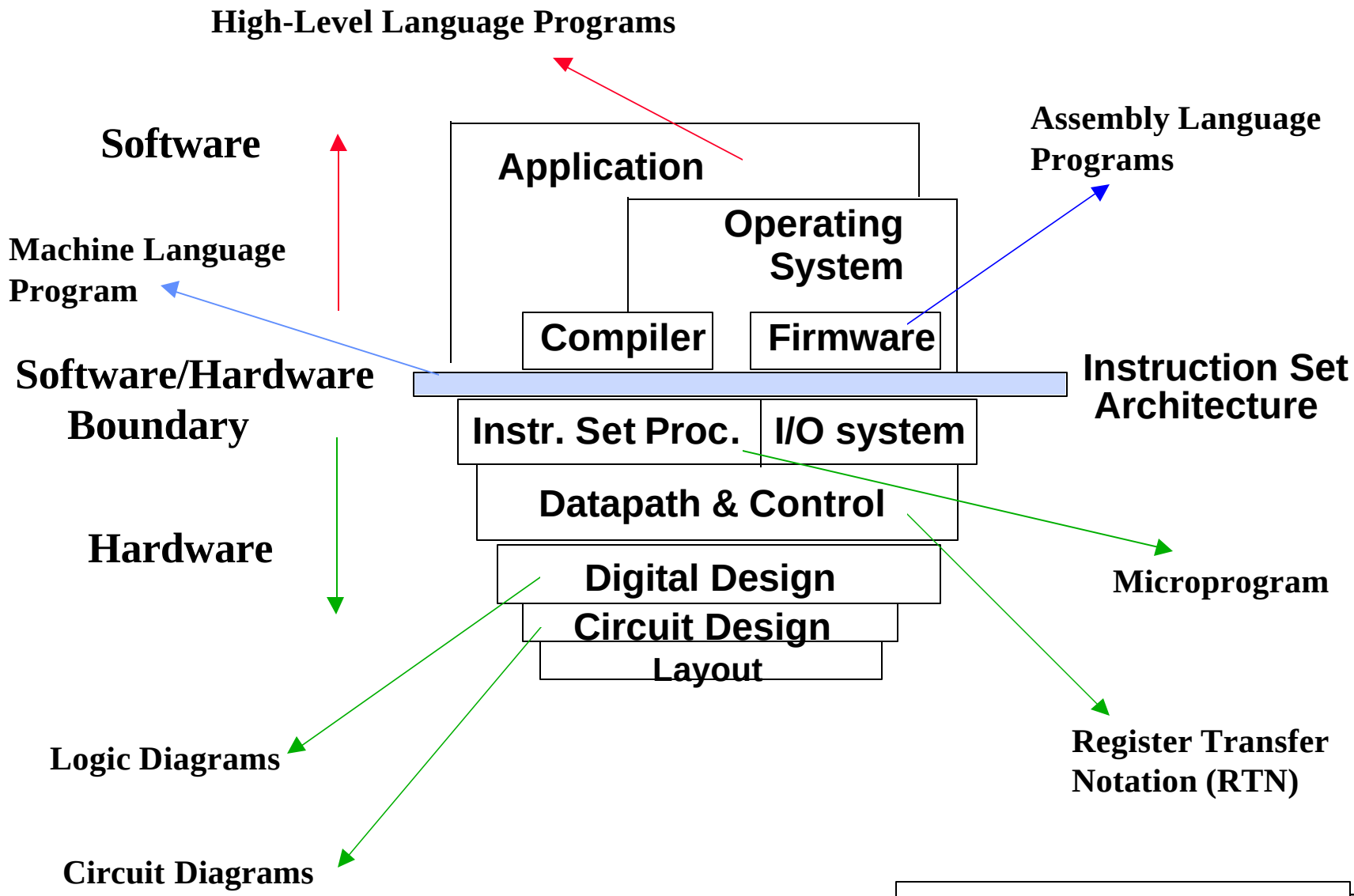
- **An Integrated circuit (IC) is a number of logic gated fabricated on a single silicon chip.**
- **ICs can be classified according to how many gates they contain as follows:**
  - **Small-Scale Integration (SSI):** Contain 1 to 20 gates.
  - **Medium-Scale Integration (MSI):** Contain 20 to 200 gates. 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.**

Hierarchy of Computer Architecture



A Hierarchy of Computer Design

Level	Name	Modules	Primitives	Descriptive Media
1	Electronics	Gates, FF's	Transistors, Resistors, etc.	Circuit Diagrams
2	Logic	Registers, ALU's ...	Gates, FF's ….	Logic Diagrams
3	Organization	Processors, Memories	Registers, ALU's …	Register Transfer Notation (RTN)
4	Microprogramming	Assembly Language	Microinstructions	Microprogram
5	Assembly language programming	OS Routines	Assembly language Instructions	Assembly Language Programs
6	Procedural Programming	Applications Drivers ..	OS Routines High-level Languages	High-level Language Programs
7	Application	Systems	Procedural Constructs	Problem-Oriented Programs

Low Level - Hardware

Firmware

High Level - Software