Basic Electronics and Digital Logic
Computer Science 240

Laboratory 1

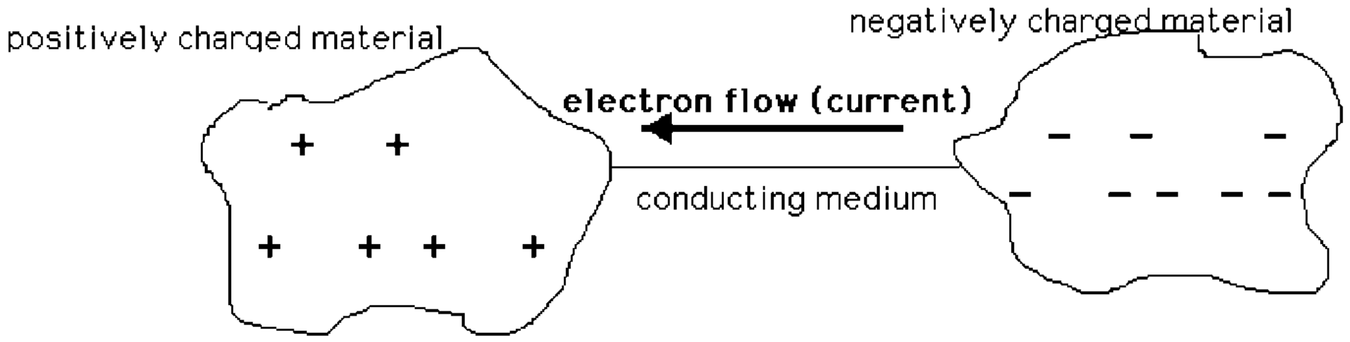
- Administrivia
- Lab Environment
- Basic Electronics (Ohm’s law, transistors, logic gates)
- Sum-of-Products and Equivalence
- Universal Gates
- Binary and Hexadecimal Numbers
- Integrated Circuits
- Protoboard (for building physical circuits)
- LogicWorks (for simulating circuits)

Lab Environment

- All lab exercises and reports will be *Google Docs*, and should be shared with lab partner and the instructor.
- You must use the machines in the room for lab (not your laptop).
- There are 2 lab machines for each pair of students. For the first several labs, the machine booted to Windows will be used for circuit simulation.
- To log in to a machine booted to Windows, use your Wellesley network username and password.
- To log in to a lab machine booted to Linux, log in using your CS server username and password.

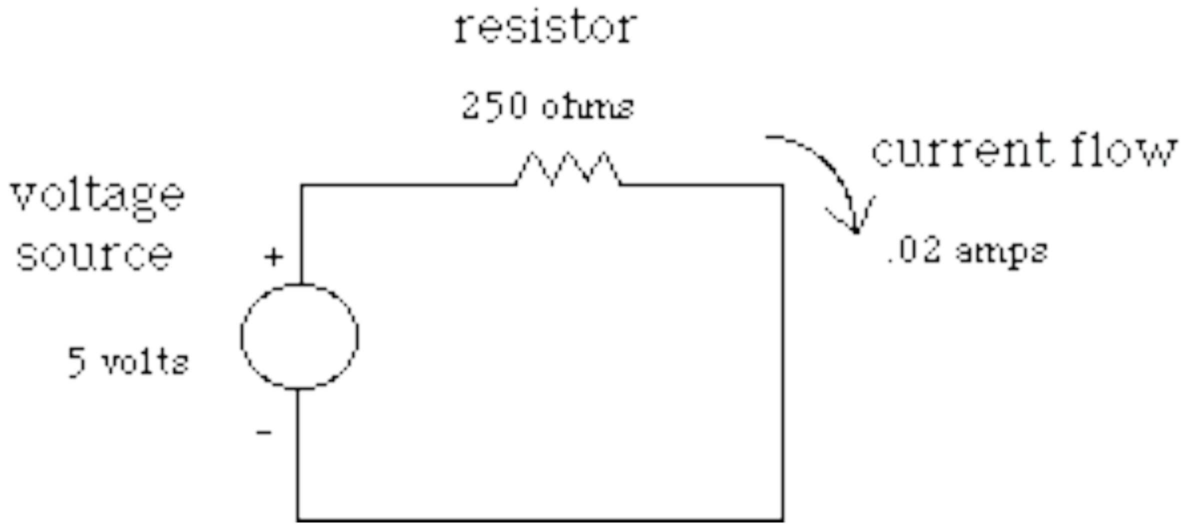
Basic Concepts of Electricity

- Electricity = **the movement of electrons** in a material
- Materials tend to have a net negative or positive charge
- Difference of charge between two points = **potential difference (V)**



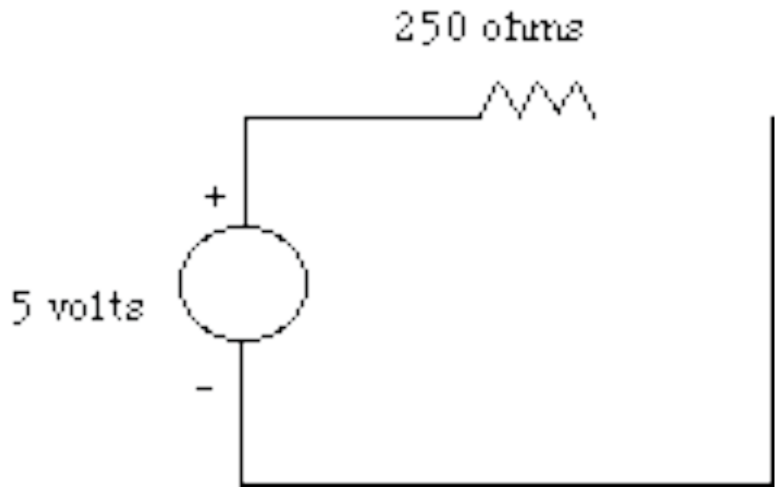
Rate at which electrons flow through = **current (A)**.

Ease of conduction, or current flow = **resistance (Ω)**

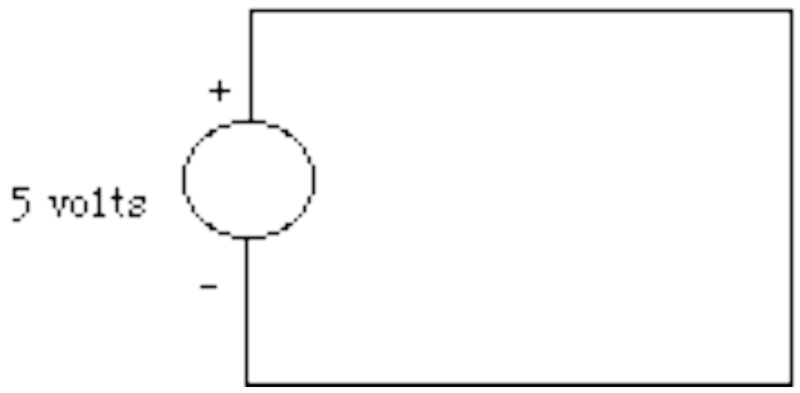


Ohm's Law, V = IR.

Open circuit = no current:



Short circuit = infinite current, since V/0 = infinite current:



Infinite current swiftly results in the destruction of the circuit!

Basic Gates and Truth Tables

Truth tables specify the output for all the given input combinations of a function.

An input combination can be expressed by ANDing together the inputs (each input or its' complement is used in the expression, depending upon whether the input is 0 or 1 for a given combination).

There are several basic logic functions which are fundamental to our study of digital electronics. They include (with given truth tables and gate symbols for each):

NOT
F = A'

A	F
0	1
1	0

NAND
F = (AB)'

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR
F = (A+B)'

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

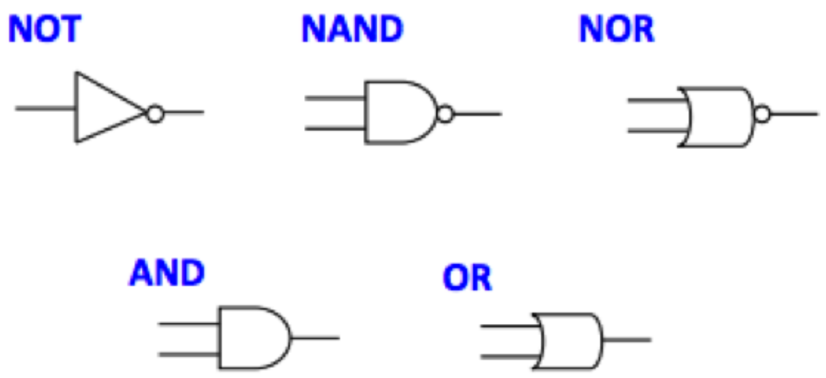
AND
F = AB

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

OR
F = A + B

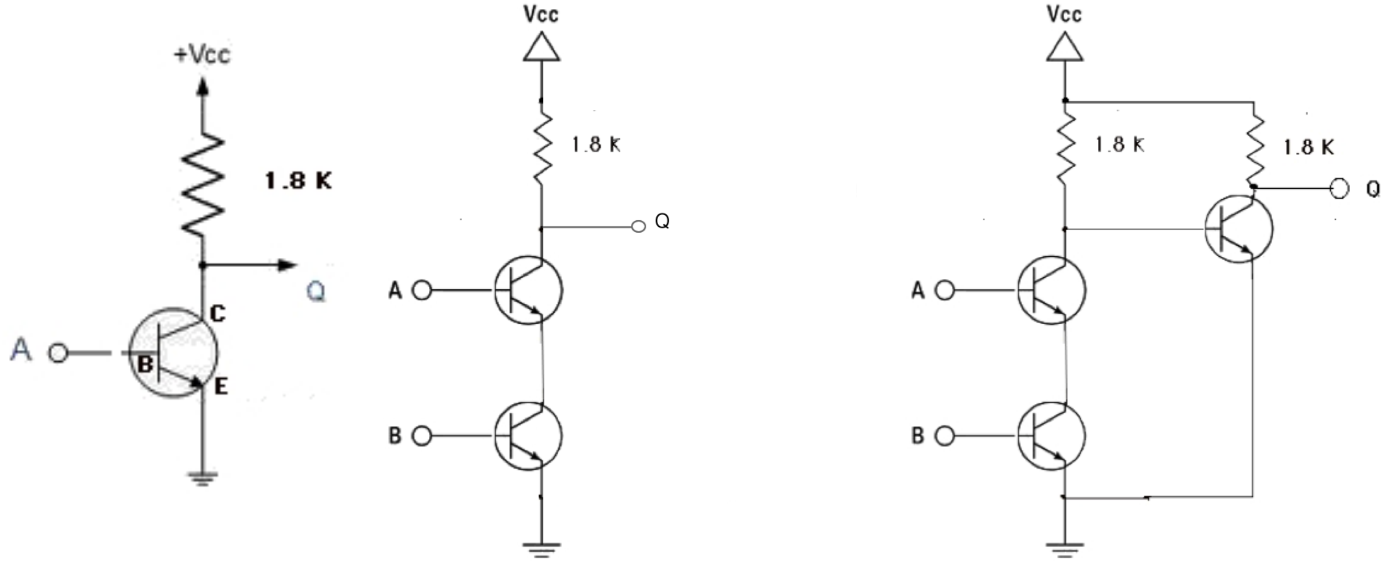
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Basic Gate Symbols



Circuits for the Basic Gates are built using **transistors.** You have seen the circuits for NOT and NAND in lecture:

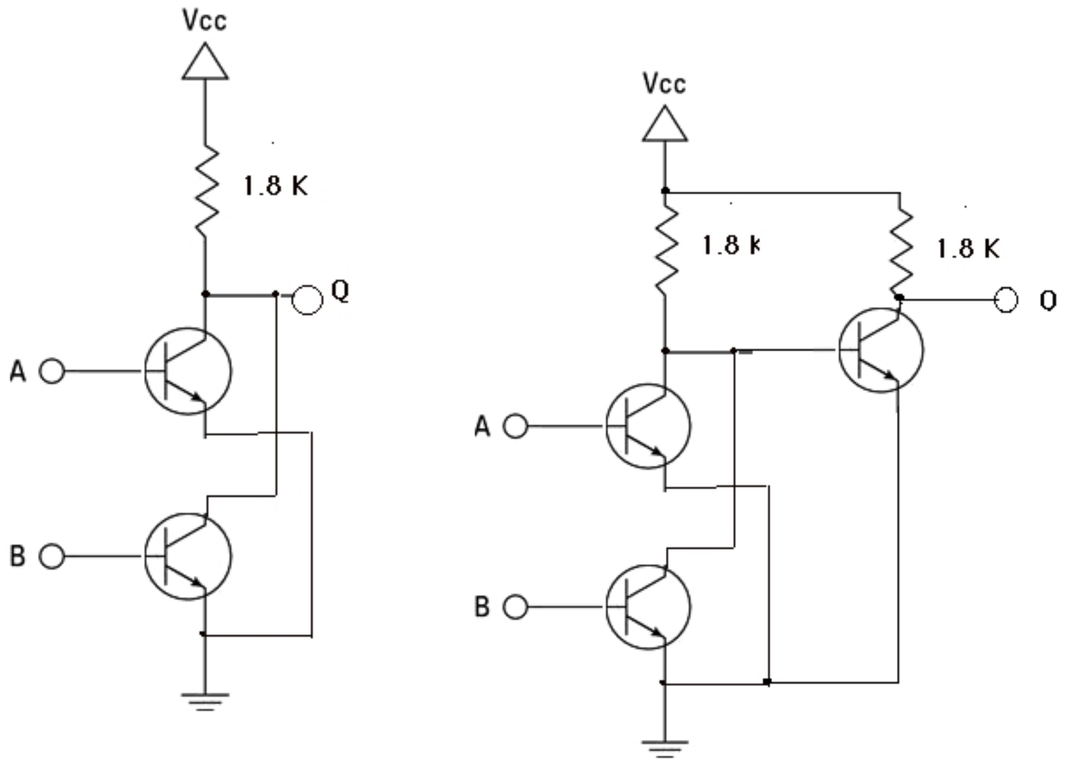
NOT – 1 transistor **NAND – 2 transistors** **AND** – 3 transistors



The **AND** gate uses 3 transistors is basically a **NOT NAND** (it sends the output of a NAND through another transistor acting as a NOT gate to complement the result):

Similarly, these are the transistor circuits for a NOR and OR gate:

NOR – 2 transistors **OR** – 3 transistors



Sum-of-products and Equivalence

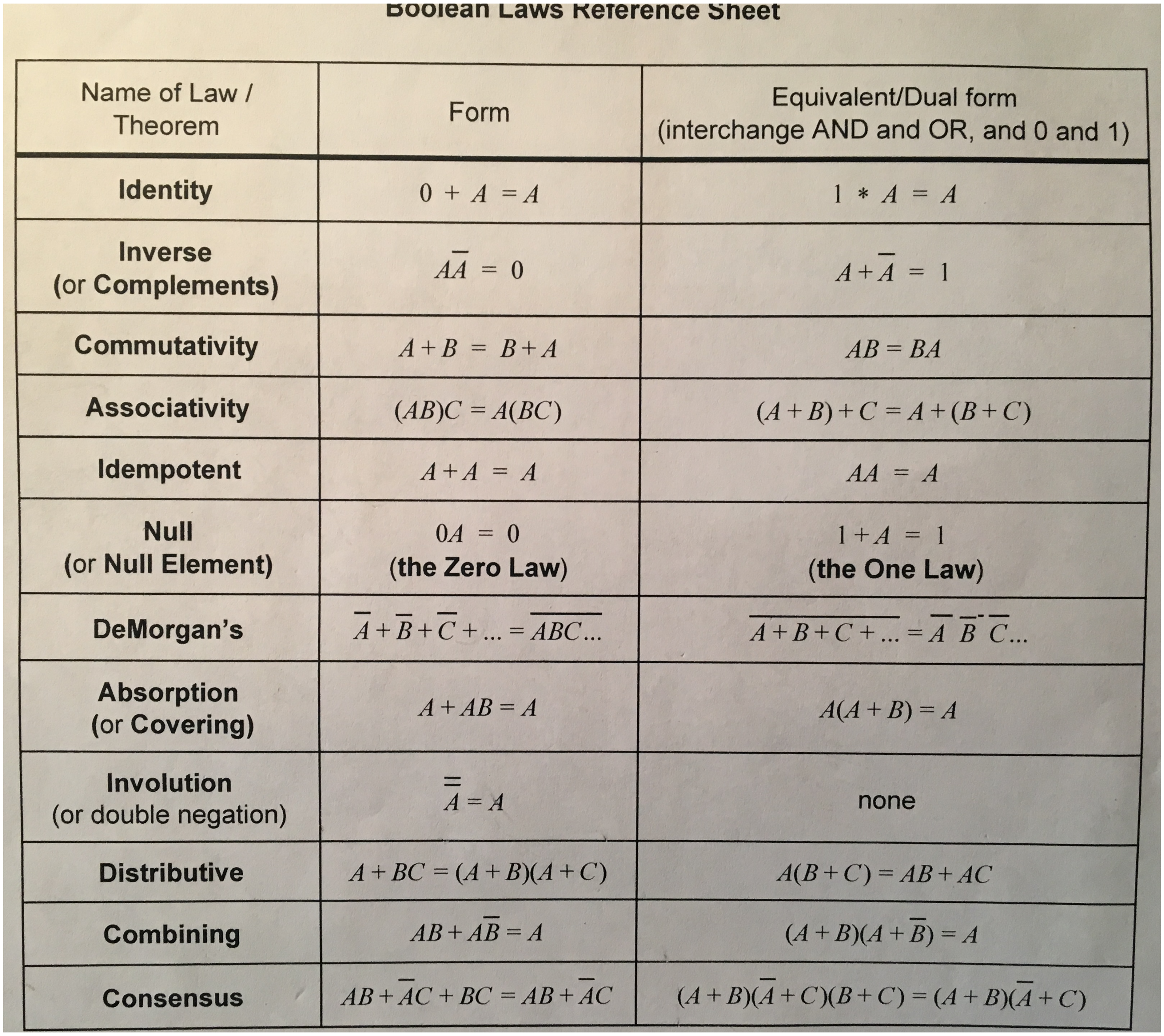
A truth tables for a function can be expressed in a **sum-of-products** form, by ORing together the input combinations which make the function true. For example, F = A'B' + A'B and Q = A' + A'B + A'B':

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

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

F and Q are **equivalent** (produce the same function) because they have the same truth table. When there is an equivalent function/circuit that uses fewer gates, transistors, or chips, it is preferable to use that circuit in a design.

Equivalence can also be proved through use of the Boolean Laws:



Boolean Laws Reference Sheet

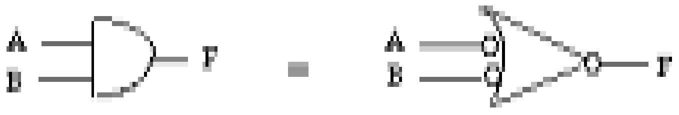
Name of Law / Theorem	Form	Equivalent/Dual form (interchange AND and OR, and 0 and 1)
Identity	$0 + A = A$	$1 * A = A$
Inverse (or **Complements**)	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutativity	$A + B = B + A$	$AB = BA$
Associativity	$(AB)C = A(BC)$	$(A+B)+C = A+(B+C)$
Idempotent	$A + A = A$	$AA = A$
Null (or **Null Element**)	$0A = 0$ **(the Zero Law)**	$1 + A = 1$ **(the One Law)**
DeMorgan's	$\overline{A}+\overline{B}+\overline{C}+\ldots = \overline{ABC\ldots}$	$\overline{A+B+C+\ldots} = \overline{A}\ \overline{B}\ \overline{C}\ldots$
Absorption (or **Covering**)	$A + AB = A$	$A(A+B) = A$
Involution (or double negation)	$\overline{\overline{A}} = A$	none
Distributive	$A + BC = (A+B)(A+C)$	$A(B+C) = AB + AC$
Combining	$AB + A\overline{B} = A$	$(A+B)(A+\overline{B}) = A$
Consensus	$AB + \overline{A}C + BC = AB + \overline{A}C$	$(A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C)$

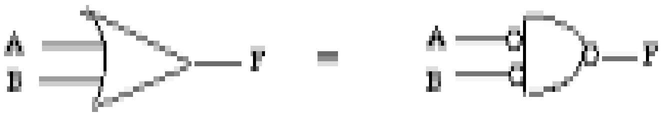
Universal Gates

Any Boolean function can be constructed with only NOT, AND, and OR gates, but also with either only NAND gates, or only NOR gates. Therefore, NAND and NOR are considered **universal gates.**

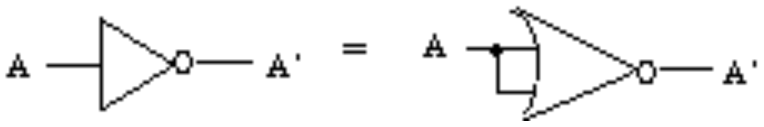
DeMorgan's Law shows how to make **AND** from NOR (and vice-versa)

AB = (A' + B')' (**AND** from NOR)
A + B = (A'B')' (**OR** from NAND)

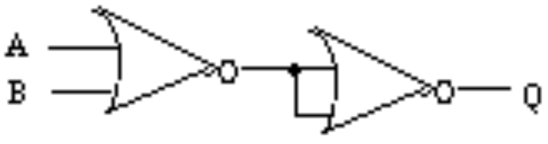




NOT from a NOR:



OR from a NOR:



To implement a function using only NOR gates

- Apply DeMorgan's to each AND until all ANDs are converted to NORs
- Use a NOR gate for any NOT gates, as well.
- Remove any redundant gates (for a NOT NOT, you may remove both)

Implementing the circuit using only NAND gates is similar.

To simplify a function or prove equivalency

- Distribute; if you can't, apply DeMorgan's Law so that you can.
- Apply other identities to remove terms, and repeat step 1.

Binary and Hexadecimal Numbers

Hex	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Note the patterns:

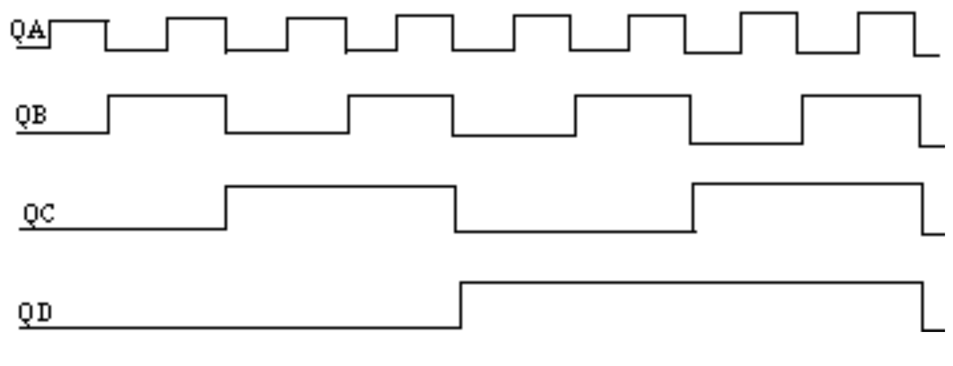
- The least significant bit QA of the 4-bit value toggles between 0 and 1 for each consecutive number.

- The next least significant bit QB toggles every two values.

- QC toggles every four values.

- QD toggles every eight values.

Graphically, the truth table may be represented by the following (assume voltage is the vertical axis and time is the horizontal axis):



You will be seeing some waveforms like this in lab today.

Conversions
Binary can be converted to decimal using positional representation of powers of 2:

$$0111_2 = 0 \text{ x } 2^3 + 1 \text{ x } 2^2 + 1 \text{ x } 2^1 + 1 \text{ x } 2^0, \quad \text{result} = 7_{10}$$

Decimal can be also be converted to binary by finding the largest power of 2 which fits, subtract, and repeat with the remainders until remainder is 0 (assigning 1 to the positions where a power of 2 is used, and 0 to the positions not represented by a power of 2):

$$6_{10} = 6 - 2^2 = 2 - 2^1 = 0, \quad \text{result} = 0110_2$$

Hex can be converted to binary and vice versa by grouping into 4 bits.

$11110101_2 = F5_{16}$ $\qquad$ $37_{16} = 00110111_2$

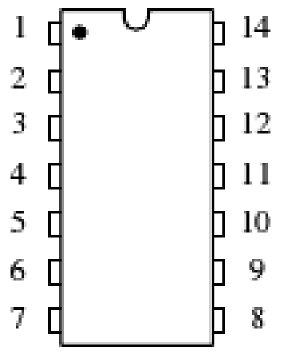
Integrated Circuits

Integrated circuits (chips) each have a specific function and physical configuration. The **pin-out** (found in TTL Data Book or online) shows the physical layout of the pins:

Pins are numbered, starting with "1" at the top left corner and incremented counter-clockwise around the device.

Top left pin is pin 1, always to left of notch in chip, and often marked with a dot

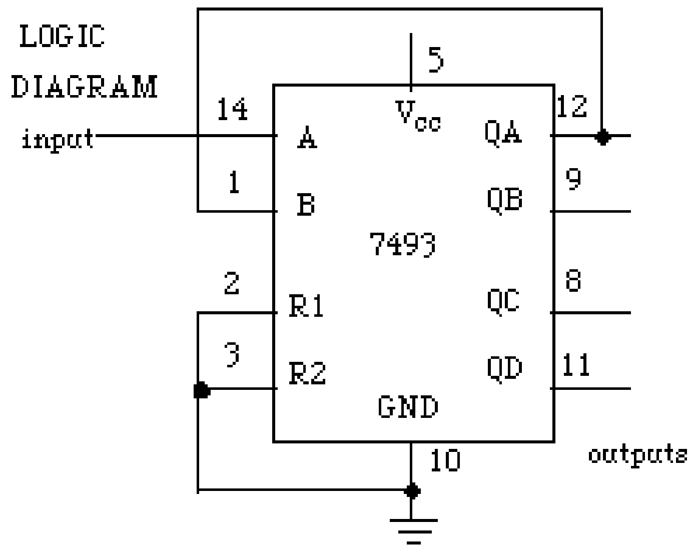
Bottom left pin is often connected to ground (0V)



Top right pin is often connected to Vcc (+5V)

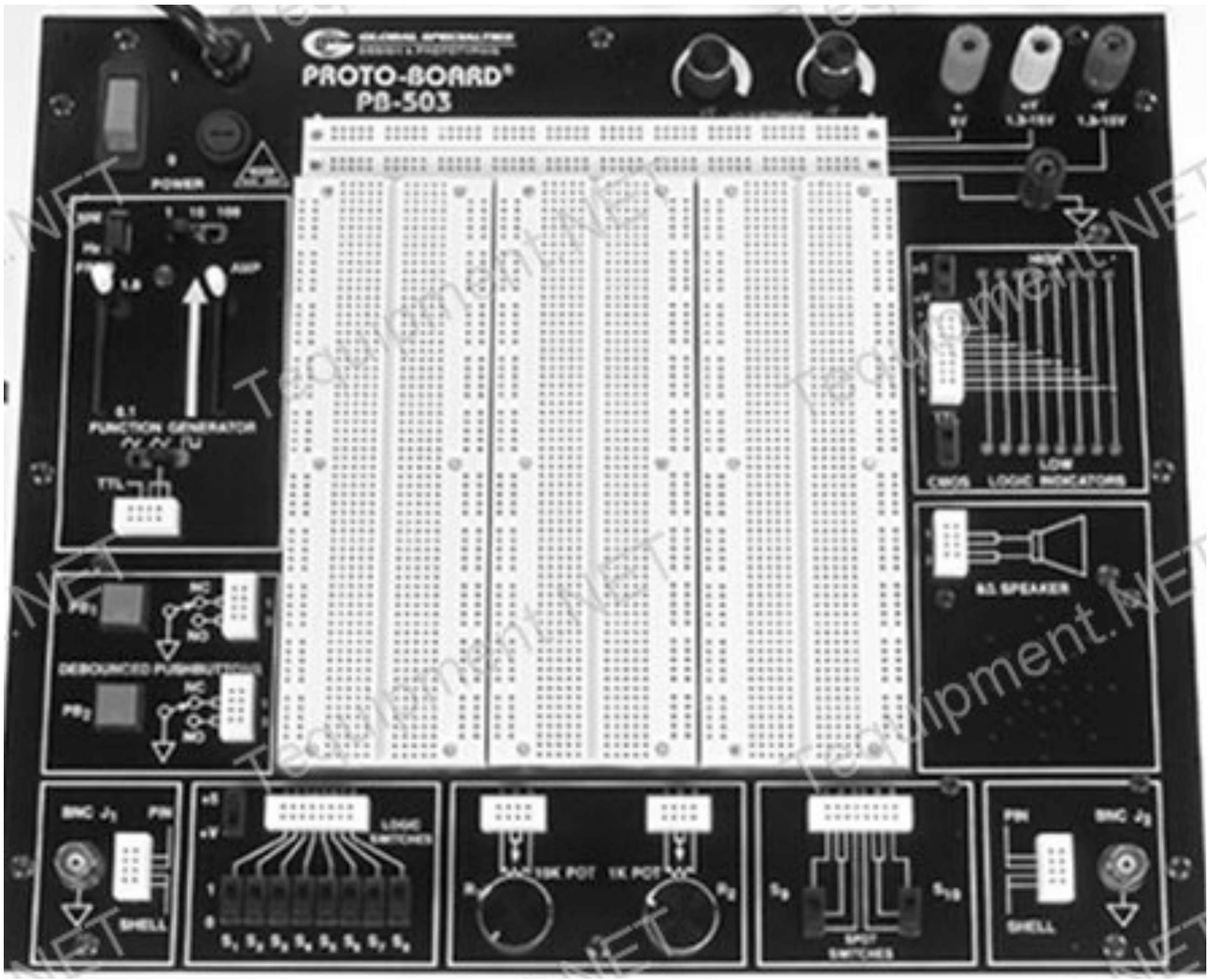
The chip will not work if it is not connected to power and ground!

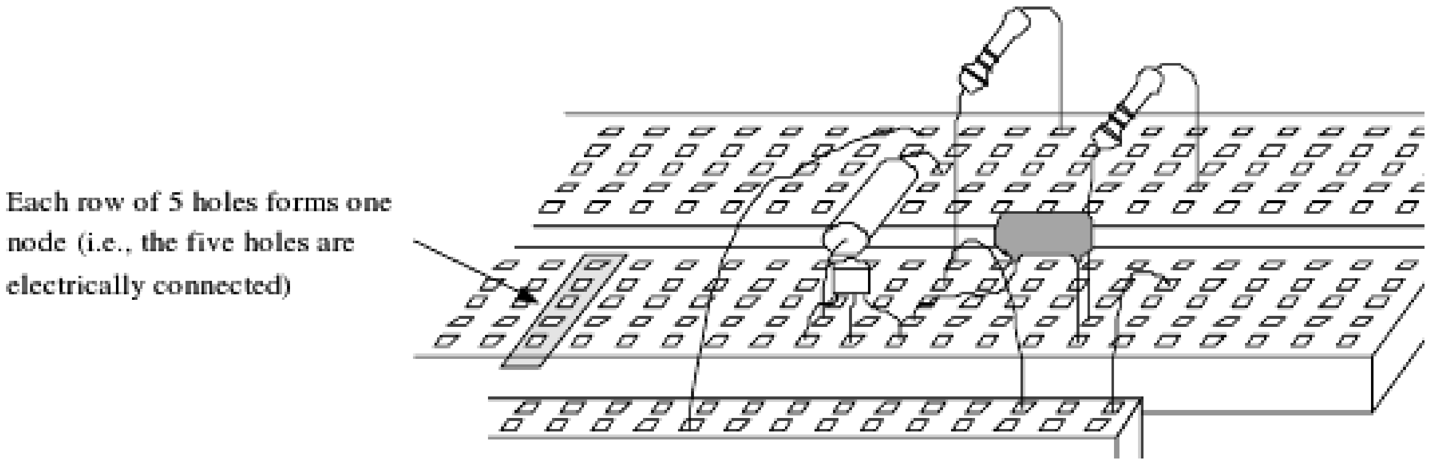
Logic diagrams are not the same as pin-outs! They show information about the logical operation of the device, and do not reflect the actual position of the pins on the chip:



Protoboard for building circuits

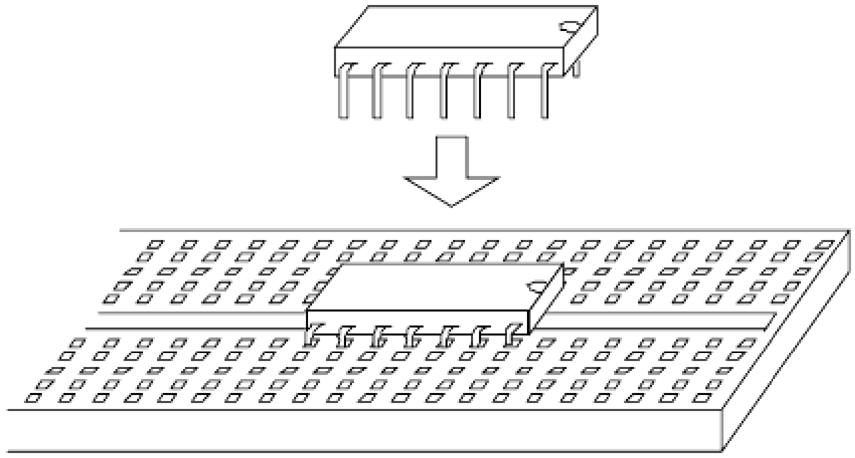
A protoboard is a tool to create prototype circuits. It contains built-in power supplys, switches to supply inputs to circuits, LEDs to display outputs of circuits, and an array of holes/tie points in which components and wires can easily be inserted to connect circuits:




Each row of 5 holes forms one node (i.e., the five holes are electrically connected)

All holes in a row internally connected (use to tie one point to another in the circuit), using .22 gauge wires with 1/4" of insulation stripped from both ends.

Insert chip straddling the groove, with pin1 on the top left



Testing and Debugging

To test that circuit is correct, check the outputs as the inputs are switched through all possible combinations.

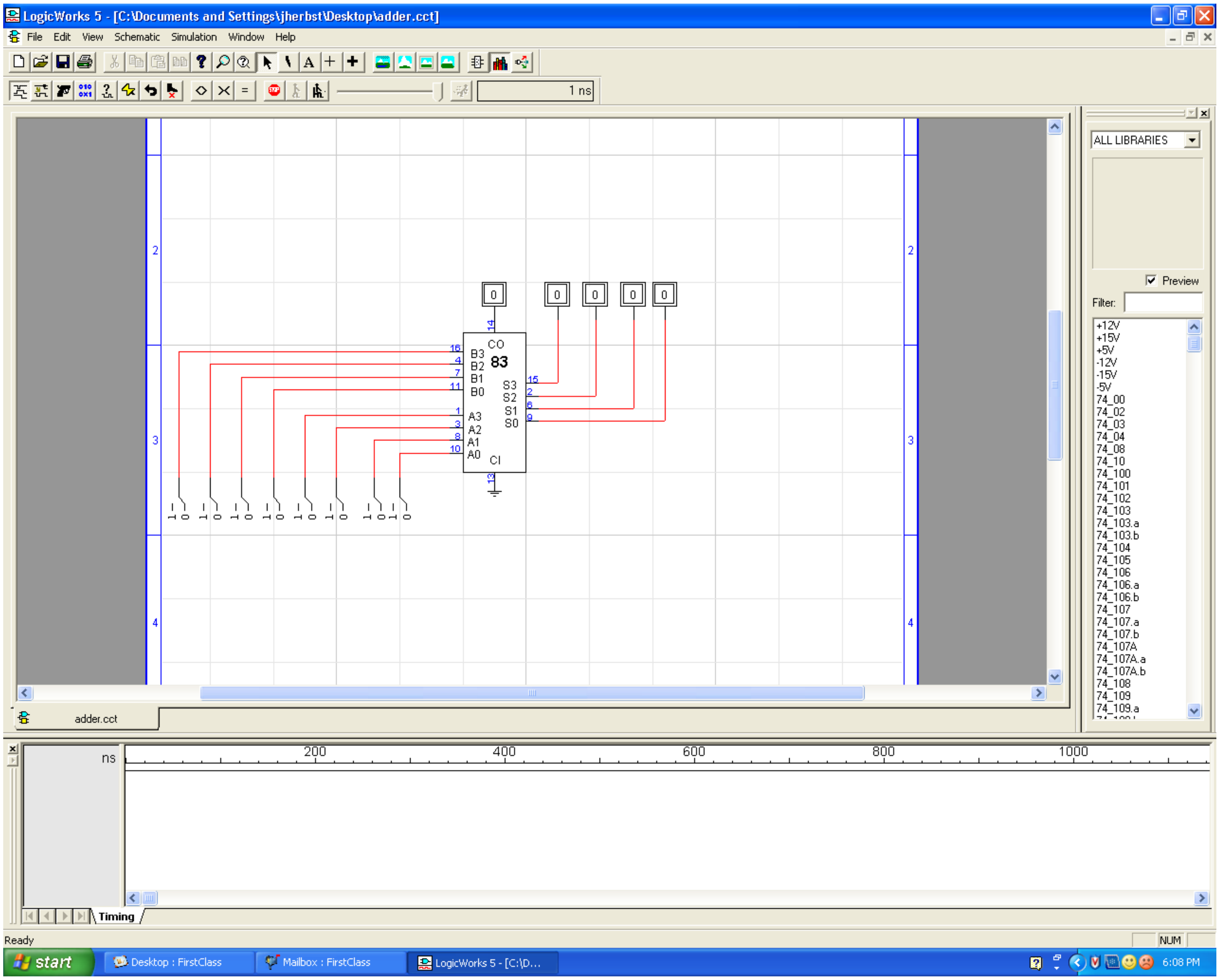
Possible causes of incorrect output (and actions to take):

1. Incorrect chip (replace with correct chip)
2. Power and/or ground not connected (connect power and ground)
3. Wrong pin connections (re-wire to connect correctly)
4. Bad gate/chip (replace with another device)

Following a systematic process to debug and correct a circuit is preferable (and often much less frustrating) than simply tearing out the circuit and starting again.

Check each of the possible problems (possibly with the use of a **logic probe** to test that inputs and outputs are valid), and then re-test the circuit when the problem has been identified.

Circuit Simulation/LogicWorks


LogicWorks 5 - [C:\Documents and Settings\jherbst\Desktop\adder.cct]
File Edit View Schematic Simulation Window Help
1 ns
ALL LIBRARIES
Preview
Filter:
+12V
+15V
+5V
-12V
-15V
-5V
74_00
74_02
74_03
74_04
74_08
74_10
74_100
74_101
74_102
74_103
74_103.a
74_103.b
74_104
74_105
74_106
74_106.a
74_106.b
74_107
74_107.a
74_107.b
74_107A
74_107A.a
74_107A.b
74_108
74_109
74_109.a
83
CO
B3
B2
B1
B0
A3
A2
A1
A0
CI
S3
S2
S1
S0
0
adder.cct
ns
200
400
600
800
1000
Timing
Ready
NUM
start
Desktop : FirstClass
Mailbox : FirstClass
LogicWorks 5 - [C:\D...
6:08 PM