

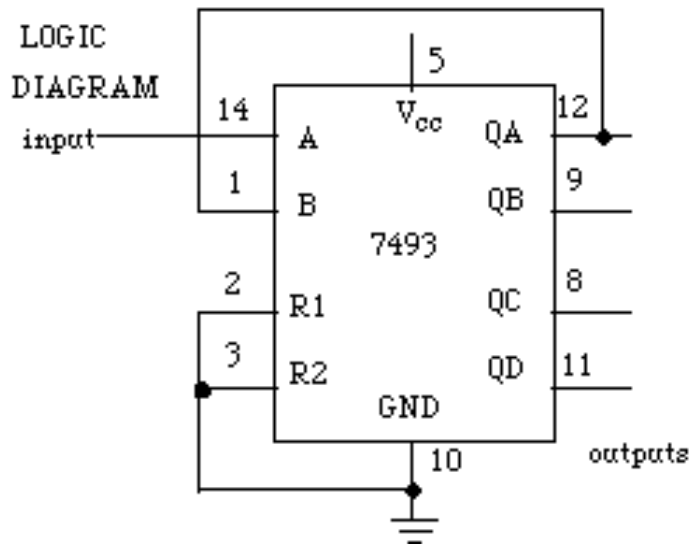
CS240 Laboratory 2

Digital Logic

- **Protoboard/Integrated circuits review**
- **Logic Diagrams vs. pin-outs**
- **LogicWorks demo**
- **Circuit equivalence**
- **Boolean Algebra/Universal gates**
- **Exclusive OR**
- **Binary Numbers**

Logic diagrams vs. pin-outs

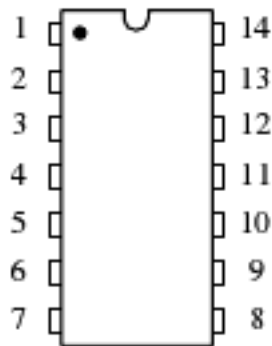
Logic diagrams are not the same as pin-outs! Logic diagrams show information about the logical operation of the device.



Pin-outs (found in **TTL Data Book** or online) show the physical layout of the pins:

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

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



Bottom left pin is almost always connected to ground (0V)

Top right pin is almost always connected to V_{cc} (+5V)

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

Circuit Simulation/LogicWorks (demo)

The screenshot displays the LogicWorks 5 interface for a circuit simulation. The main workspace shows a 4-bit adder component (labeled '83') with inputs A0-A3 and B0-B3, and outputs S0-S3 and CO. The inputs are connected to switches, and the outputs are connected to LEDs. The simulation is running at 1 ns.

The right-hand panel shows the 'ALL LIBRARIES' section with a list of components. The list includes:

- +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

The bottom of the window shows the Windows taskbar with the Start button, several open applications (Desktop: FirstClass, Mailbox: FirstClass, LogicWorks 5 - [C:]D...), and the system tray showing the time as 6:08 PM.

Circuit Equivalence

Two boolean functions with same truth table = **equivalent**

When there is an equivalent circuit which uses fewer gates, transistors, or chips, it is preferable to use that circuit in the design

Example:

Given: $F = A'B' + A'B$

$Q = A' + A'B + A'B'$

A	B	A'B'	A'B	F
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'	Q
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 because they have the same truth table.

Identities of Boolean Algebra

- Identity law $1A = A$ $0 + A = A$
- Null law $0A = 0$ $1 + A = 1$
- Idempotent law $AA = A$ $A + A = A$
- Inverse law $AA' = 0$ $A + A' = 1$
- Commutative law $AB = BA$ $A + B = B + A$
- Associative law $(AB)C = A(BC)$
 $(A + B) + C = A + (B + C)$
- Distributive law $A + BC = (A + B)(A + C)$
 $A(B + C) = AB + AC$
- Absorption law $A(A + B) = A$
 $A + AB = A$
- De Morgan's law $(AB)' = A' + B'$
 $(A + B)' = A'B'$

Example:

$$\begin{aligned} F &= A'B' + A'B \\ &= A'(B' + B) \text{ distributive} \\ &= A'(1) \text{ inverse} \\ &= A' \text{ identity} \end{aligned} \qquad \begin{aligned} Q &= A' + A'B + A'B' \\ &= A' + A'B' \text{ absorption} \\ &= A' \text{ absorption} \end{aligned}$$

Universal Gates

Any Boolean function can be constructed with NOT, AND, and OR gates

NAND and NOR = **universal gates**

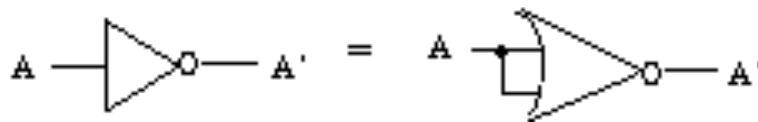
DeMorgan's Law shows how to make **AND** from NOR (and vice-versa)

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

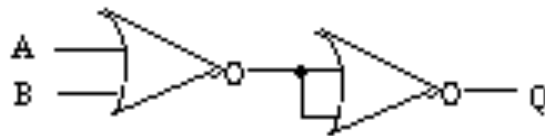
$$A + B = (A'B)'$$
 (**OR** from NAND)



NOT from a NOR



OR from a NOR



To implement a function using only NOR gates:

- apply DeMorgan's Law to each AND in the expression until all ANDs are converted to NORs
- use a NOR gate for any NOT gates, as well.
- remove any redundant gates (NOT NOT, may remove both)

Implementing the circuit using only NAND gates is similar.

Example: $Q = (AB)'B'$

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

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

(NOR gates only, since NOR can be used as a NOT gate)

Simplifying Circuits or Proving Equivalency

General rule to simplify circuits or prove equivalency:

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

EXAMPLE: Is $(A'B)'(AB)' + A'B'$ equivalent to $(AB)'$?

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

-- can't distribute

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

DeMorgan's

$$= AA' + AB' + A'B + B'B' + A'B'$$

distributive

$$= 0 + AB' + A'B + B' + A'B'$$

inverse and idempotent

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

identity

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

distributive

$$= B'(1) + A'B$$

inverse

$$= B' + A'B$$

identity

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

DeMorgan's

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

DeMorgan's

$$= (AB + BB)'$$

distributive

$$= (AB + 1)'$$

inverse

$$= (AB)'$$

identity

Exclusive OR (XOR)

$$F = AB' + A'B = A \oplus B$$

<u>A</u>	<u>B</u>	<u>F</u>
0	0	0
0	1	1
1	0	1
1	1	0

Available on IC as a gate, useful for comparison problems



Example: Even parity $F = A \oplus B \oplus C$

<u>A</u>	<u>B</u>	<u>C</u>	<u>F</u>
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

Binary Numbers

Hex	Binary			
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

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

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

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):

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

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

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