

# **CS240 Laboratory 2**

## **Digital Logic**

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

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

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

| <u>A</u> | <u>B</u> | <u>A'B'</u> | <u>A'B</u> | <u>F</u> |
|----------|----------|-------------|------------|----------|
| 0        | 0        | 1           | 0          | 1        |
| 0        | 1        | 0           | 1          | 1        |
| 1        | 0        | 0           | 0          | 0        |
| 1        | 1        | 0           | 0          | 0        |

| <u>A</u> | <u>B</u> | <u>A'</u> | <u>A'B</u> | <u>A'B'</u> | <u>Q</u> |
|----------|----------|-----------|------------|-------------|----------|
| 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**

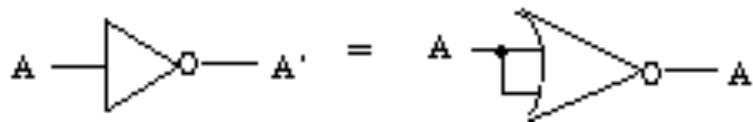
**DeMorgan's Law** shows how to make **AND** from **NOR** (and vice-versa)

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

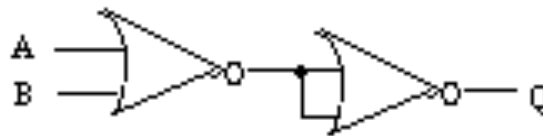
$$A + B = (A'B)'$$
 (**OR** from **NAND**)



**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):

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

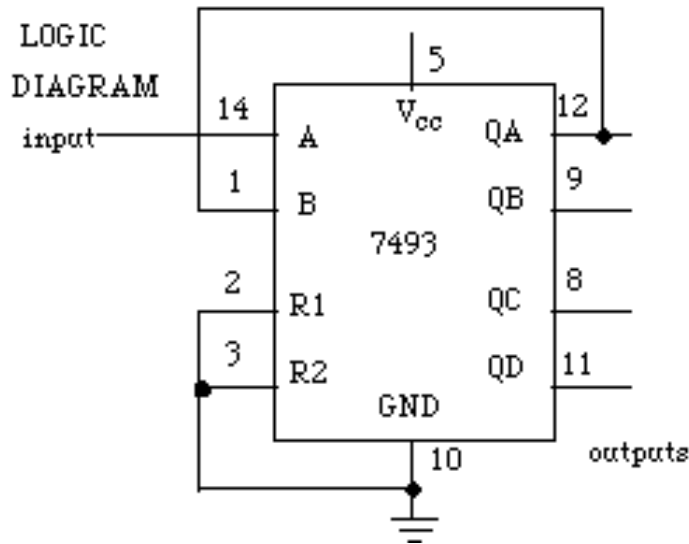
Hex can be converted to binary and vice versa by grouping into 4 bits.

$$11110101_2 = F5_{16}$$

$$37_{16} = 00110111$$

# Logic diagrams vs. pin-outs

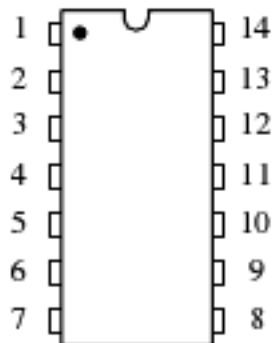
**Logic diagrams** are not the same as pin-outs! Logic diagrams show information about the logical operation of the device.



**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<sub>cc</sub> (+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 schematic of a 4-bit ripple carry adder (component 83). The adder has inputs A0-A3, B0-B3, and a carry-in CI connected to ground. The carry-out CO is connected to a logic 0. The sum outputs S0-S3 are connected to logic 0s. The timing diagram below shows a 1 ns scale.

The right-hand panel shows the component library with a list of components including power supplies (+12V, +15V, +5V, -12V, -15V, -5V) and various logic components (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 status bar shows the Windows taskbar with the Start button, taskbar buttons for Desktop: FirstClass, Mailbox: FirstClass, and LogicWorks 5 - [C:\D...], and a system tray with the time 6:08 PM.

