CS 240 Lab 2
Digital Logic
and
Introduction to Linux

- **Truth Tables, Sum-of-Products**
- **Boolean Identities**
- **Universal Gates**
- **Integrated circuits**
- **Binary and Hexadecimal Numbers**
- **Introduction to Linux**

Truth Tables and Sum-of-Products

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 which combination is being expressed)

A function can then be expressed as a **sum-of-products** by ORing together the input combinations which make the function true.

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

$F = A'B' + A'B$

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

F and Q are equivalent (produce the same function) when they have the same truth table.

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

Identities of Boolean Algebra

Equivalency can also be proved using the identities of Boolean algebra

- Identity law $1A = A \quad 0 + A = A$
- Null law $0A = 0 \quad 1 + A = 1$
- Idempotent law $AA = A \quad A + A = A$
- Inverse law $AA' = 0 \quad A + A' = 1$
- Commutative law $AB = BA \quad A + B = B + A$
- 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:

$F = A'B' + A'B$	$Q = A' + A'B + A'B'$
$= A'(B' + B)$ distributive	$= A' + A'B'$ absorption
$= A'(1)$ inverse	$= A'$ absorption
$= A'$ identity	

Universal Gates

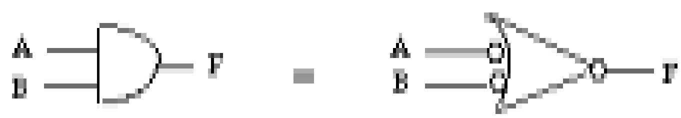
Any Boolean function can be constructed with only NOT, AND, and OR gates

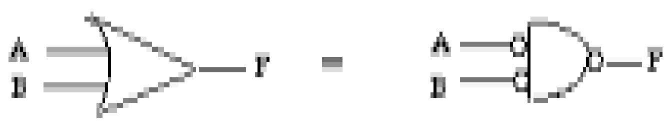
But also with either only NAND or only NOR gates = **universal gates**

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

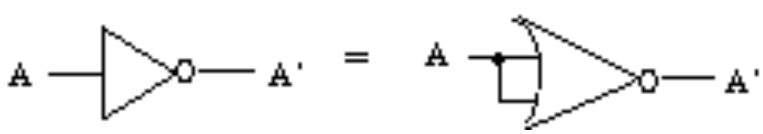
$AB = (A' + B')'$ (**AND** from NOR)

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

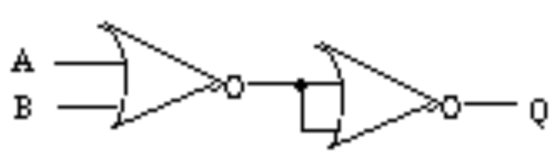




NOT from a NOR



OR from a NOR

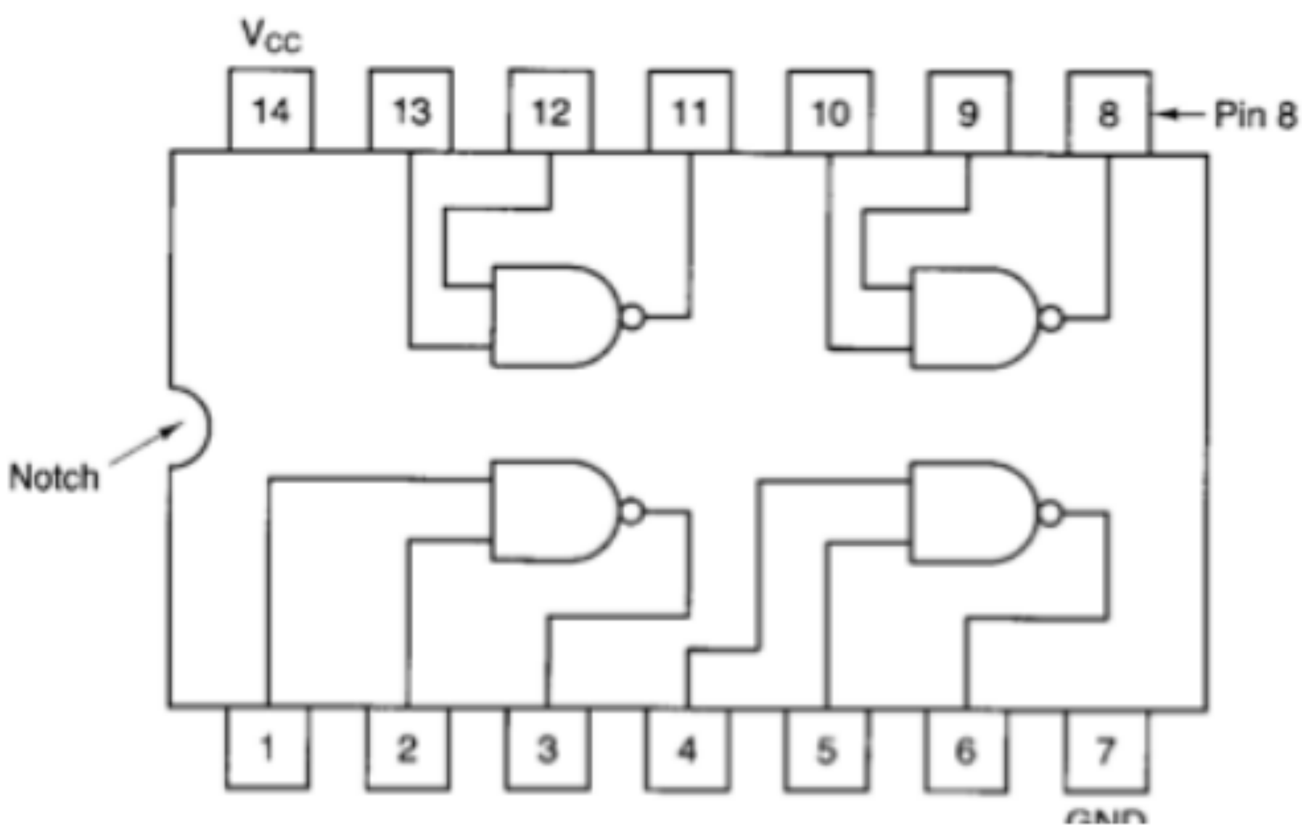


`To implement a function using only NOR gates:`

- 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.

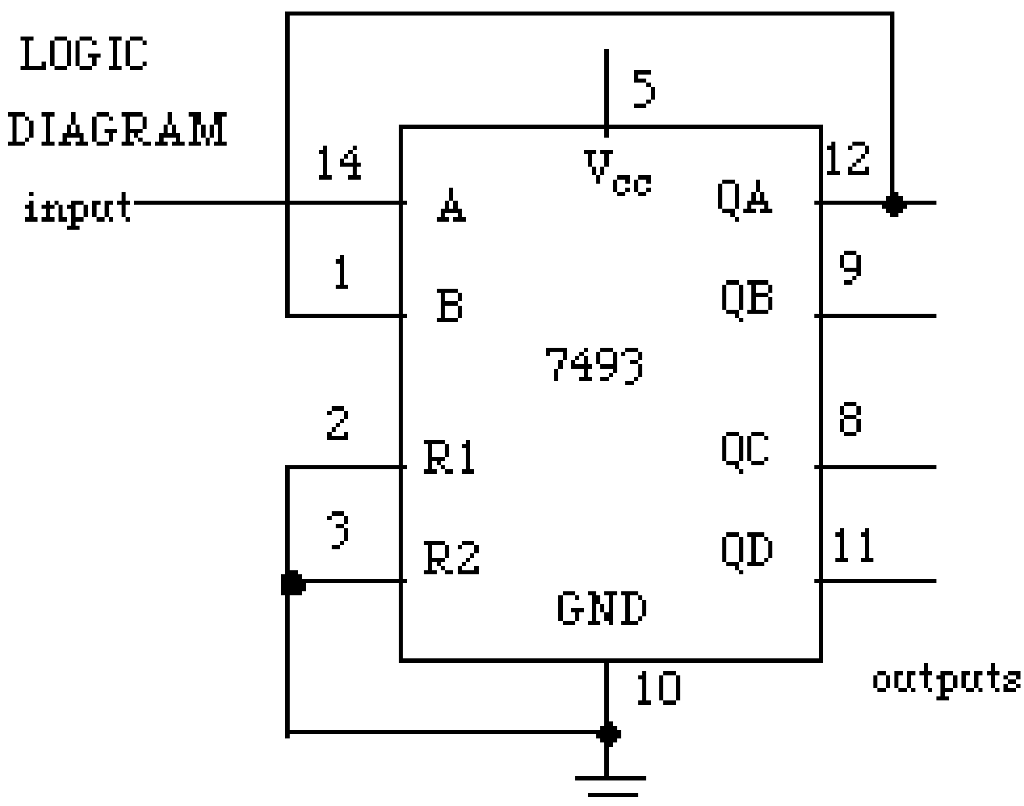
Integrated Circuits (Chips)



Logic Diagrams

Not the same as pin-outs! Show information about the logical operation of the device.

Inputs on left side of diagram
Outputs on right
Voltage shown on top
Ground shown on bottom



Binary and Hexadecimal Numbers

Hex	Binary			
	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

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

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