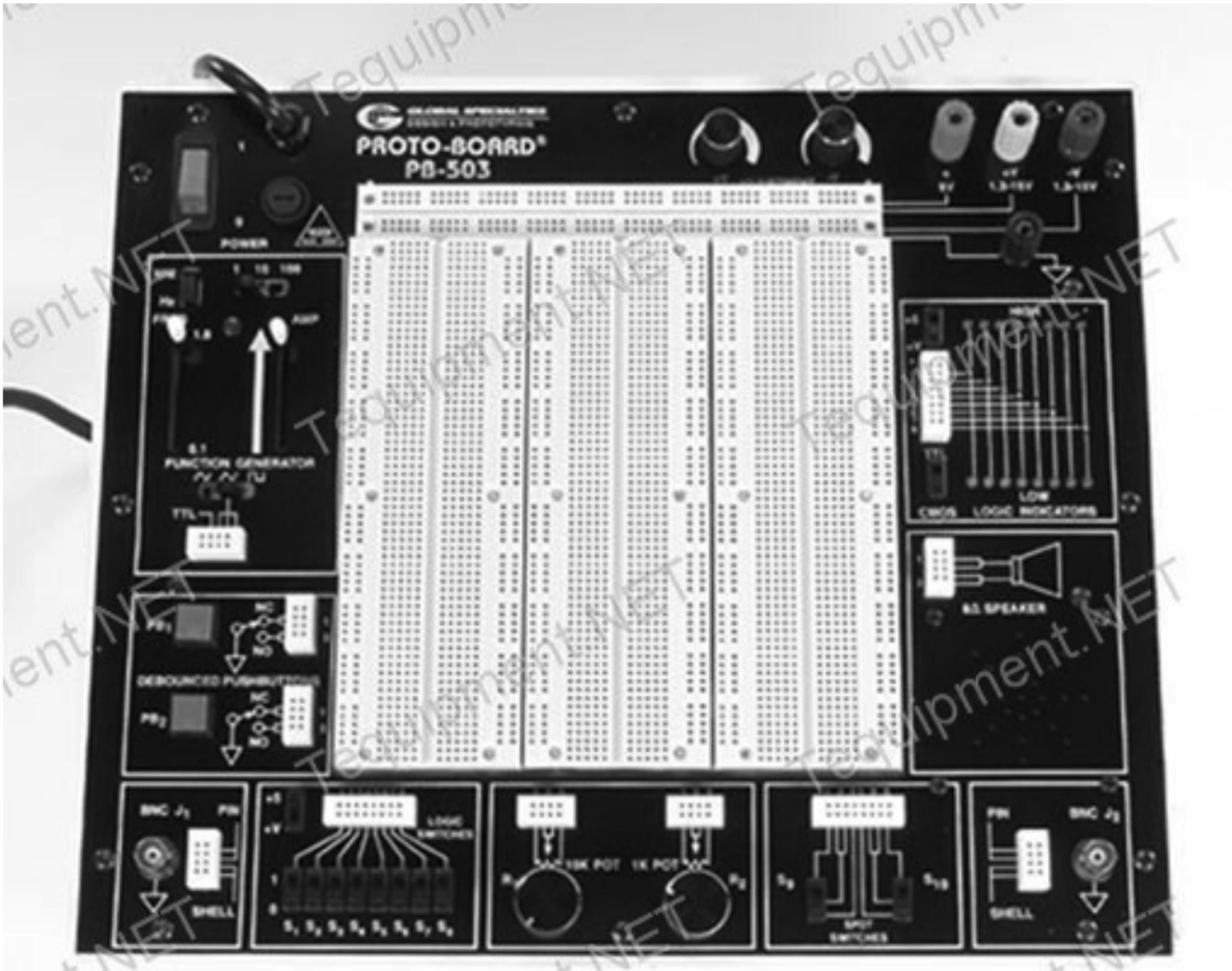
CS240 Laboratory 2
Digital Logic

- **Protoboard/Integrated Circuits review**
- **LogicWorks**
- **Circuit Equivalence**
- **Boolean Algebra/Universal Gates**
- **Exclusive OR**
- **Binary Numbers**
- **Signed Representation/Two's Complement and Overflow**

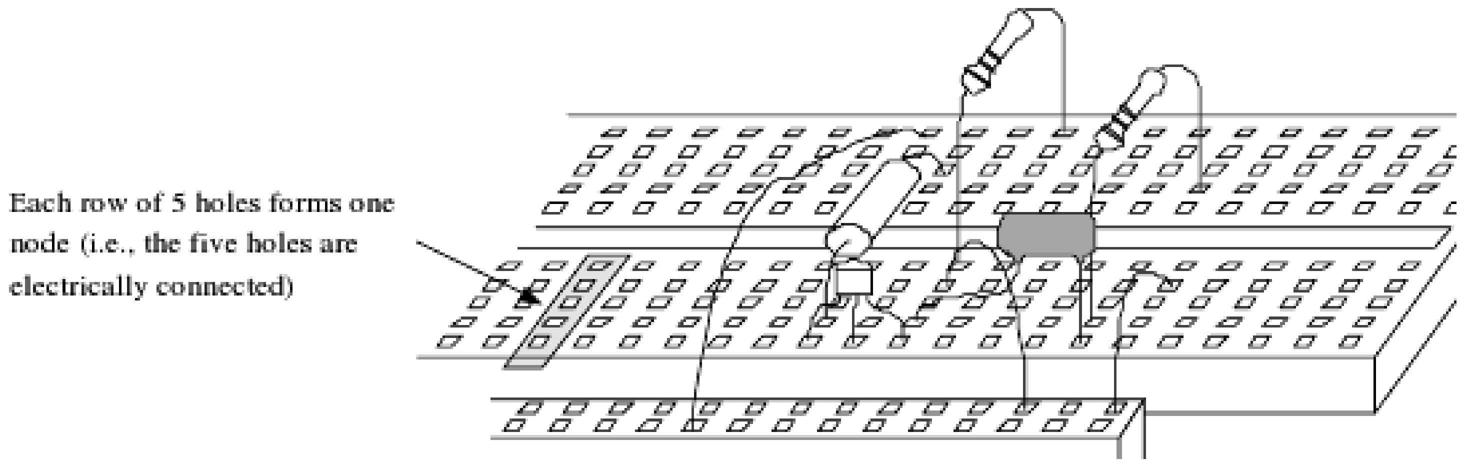
PB-503 Protoboard
Do not remove wires, resistors, or other devices already on the board.

Remove (clean up) what you have added at the end of lab!



Breadboard for wiring circuits

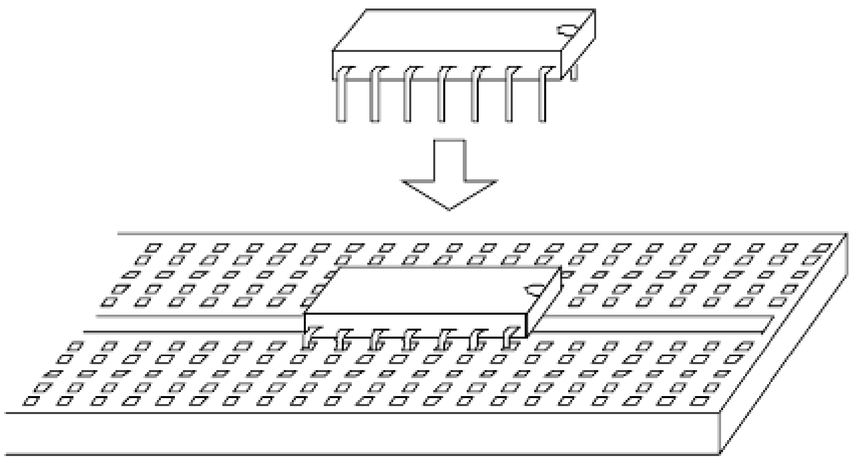
An array of holes in which wires or component leads can easily be inserted

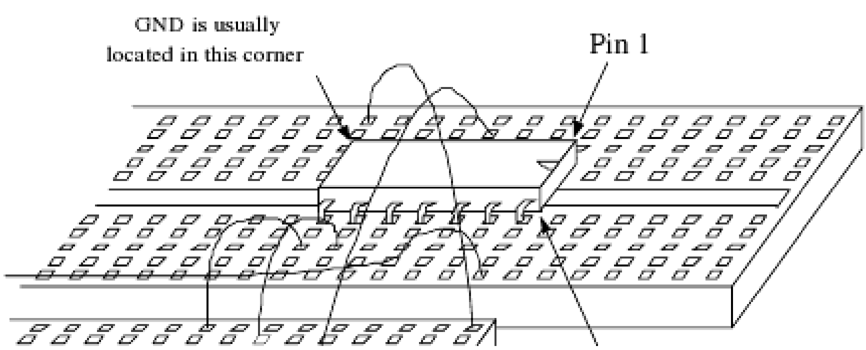


All holes in a row internally connected (use to tie one point to another in the circuit)

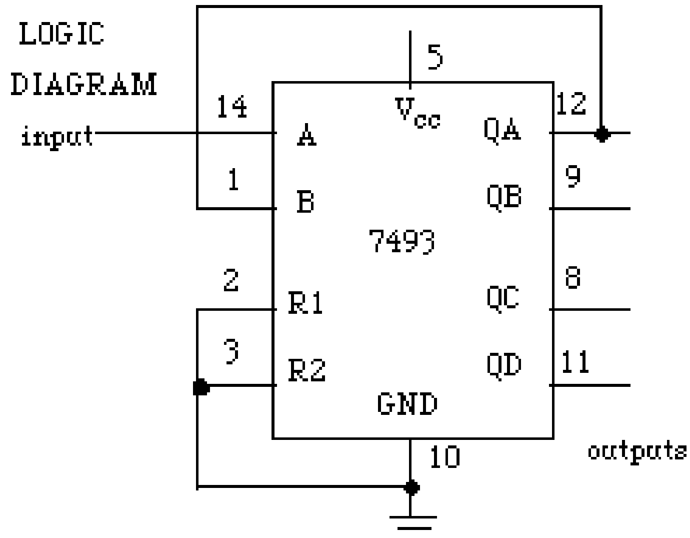
Use .22 gauge wires with 1/4” of insulation stripped from both ends

Insert chips straddling the groove





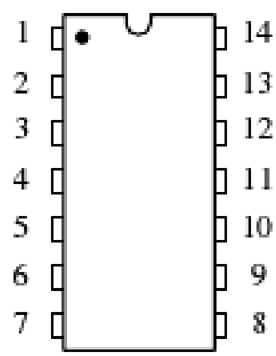
Logic diagrams are not the same as pin-outs! Show information about the logical operation of the device.



Pin-Out (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

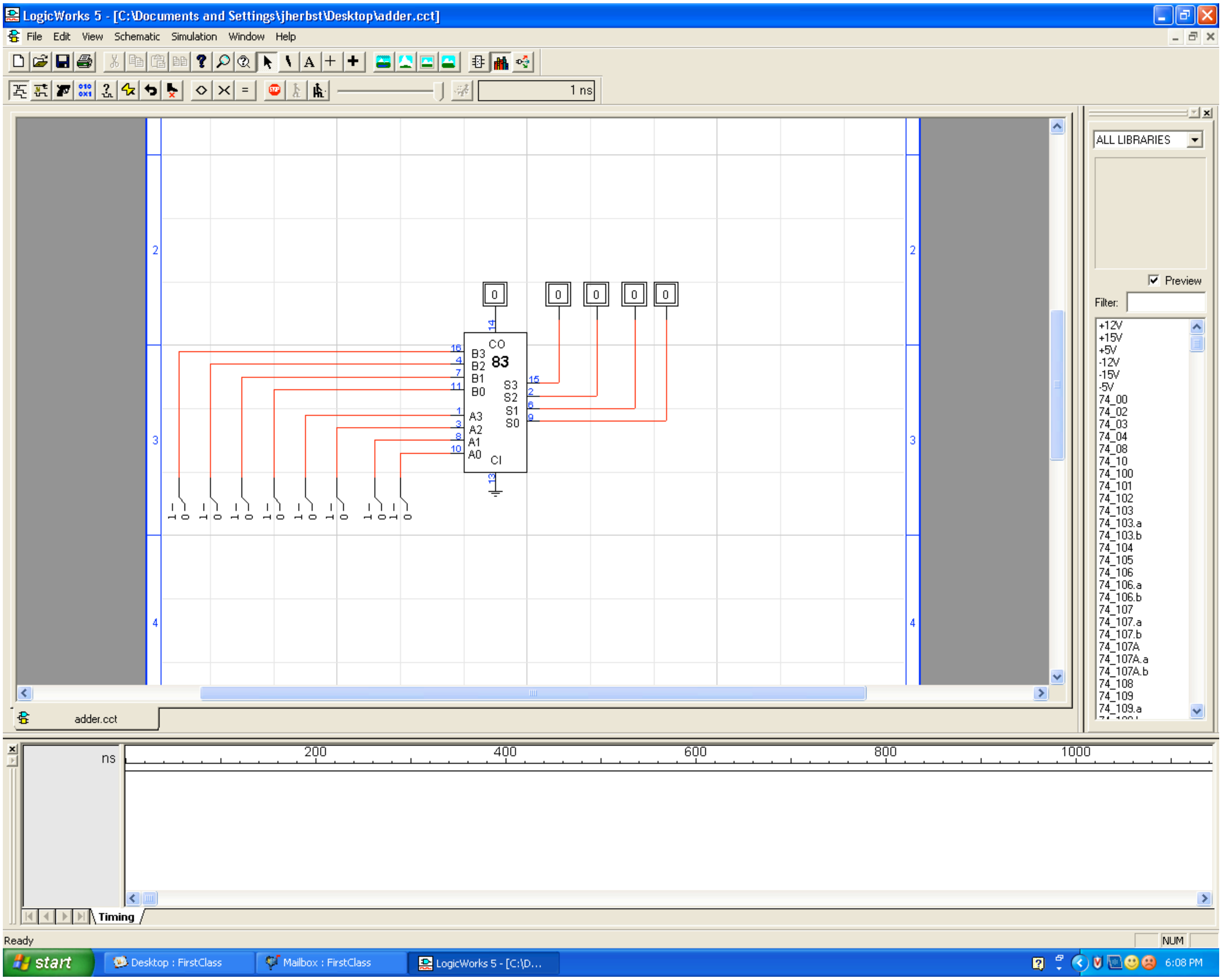


Bottom left pin is almost always connected to ground (0V)

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

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

Circuit Simulation/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'

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 $1A = A \quad 0 + A = A$
- 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 $AB = BA \qquad A + B = B + A$
- 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:

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

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

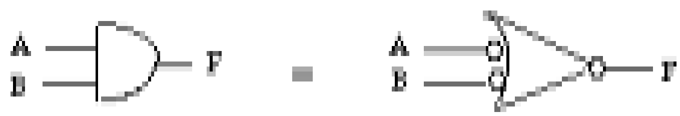
Universal Gates

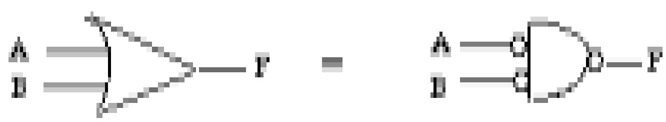
Any Boolean function can be constructed with NOT, AND, and OR gates

NAND and NOR = **universal gates**

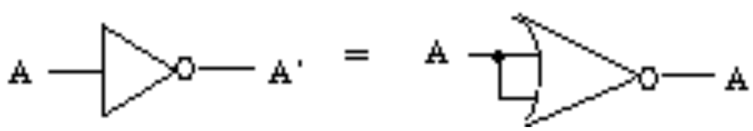
DeMorgan's Law shows how to make **AND** from NOR (and vice-versa)

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

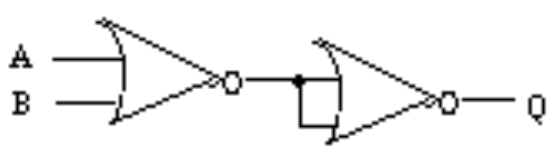




NOT from a NOR



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 $(AB)'$?

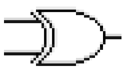
$F = (A'B)'(AB)' + A'B'$	-- can't distribute
$= (A + B')(A' + B') + A'B'$	DeMorgan's
$= AA' + AB' + A'B + B'B' + A'B'$	distributive
$= 0 + AB' + A'B + B' + A'B'$	inverse and idempotent
$= AB' + A'B + A'B'$	identity
$= B'(A+ A') + A'B$	distributive
$= B'(1) + A'B$	inverse
$= B' + A'B$	identity
$= B' + (A + B')'$	DeMorgan's
$= (B(A + B'))'$	DeMorgan's
$= (AB + BB')'$	distributive
$= (AB + 1)'$	inverse
$= (AB)'$	identity

Exclusive OR (XOR)

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

A	B	F
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 $F = A \oplus B \oplus C$

A	B	C	F
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:

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

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):

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

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

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

Signed Representation/Two's Complement and Overflow

Given n bits, range of binary values for

Unsigned representation: $0 \rightarrow 2^n - 1$
Signed representation: $-2^{n-1} \rightarrow 2^{n-1} - 1$

We use **Two's complement** to represent signed numbers. The lefttmost bit is the sign bit (0 for positive numbers, 1 for negative numbers).

Example: given a fixed number of 4 bits,

1000_2 is negative.
0111_2 is positive.

Given a fixed number of n bits, **overflow** occurs if a value cannot be represented in n bits.

Example: given 4 bits,
The largest negative value we can represent is -8_{10} (1000_2).
The largest positive value we can represent is $+7_{10}$ (0111_2).

When adding two numbers with the same sign which each can be represented with n bits, the result may cause an overflow.

An overflow occurs when either:
Two positive numbers added together yield a negative result, or
Two negative numbers added together yield a positive result.

An overflow cannot result if a positive and negative number are added.

Example: given 4 bits,

```
      0111
+     0001
      ----
      1000   overflow      NOTE:  there is not a carry-out!
```

In two's complement representation, a carry-out does not indicate an overflow, as it does in unsigned representation.

Example: given 4 bits,

```
      1001 (-7)
+     1111 (-1)
      ----
    1 1000  (-8)  no overflow, even though there is a carry-out
```