High True versus Low True

- Binary Boolean Constants are MODELS for Digital Circuit Values
- Can Let 0 Represent False and 1 Represent True – Positive Logic or High True
- Can Let 0 Represent True and 1 Represent False – Negative Logic or Low True
- Assignment is Arbitrary but Must Know When Building Circuit (ENCODING)
- "Bubble Notation" is an Aid to Determine Positive versus Negative Logic

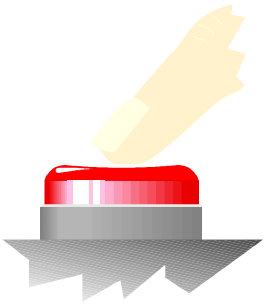
High True vs. Low True Logic

- Different ways to say that a signal is **high** true
  - Is high if signal is TRUE, is low if signal is FALSE
  - Is high if signal is 1, is low if signal is 0
  - Is high if signal is *asserted* , is low if signal is *negated* or *deasserted*
- Different ways to say that a signal is **low** true
  - Is low if signal is TRUE, is high if signal is FALSE
  - Is low if signal is 1, is high if signal is 0
  - Is low if signal is *asserted* , is high if signal is *negated* or *deasserted*

Asserted vs. Negated

- Asserted ALWAYS means that a signal is TRUE
  - Logic 1 could be represented by a HIGH voltage (high true)
  - Logic 1 could be represented by LOW voltage (low true)
- Negated (or Deasserted) ALWAYS means that a signal is FALSE
  - Logic 0 could be represented by a LOW voltage (low false)
  - Logic 0 could be represented by a HIGH voltage (high false)

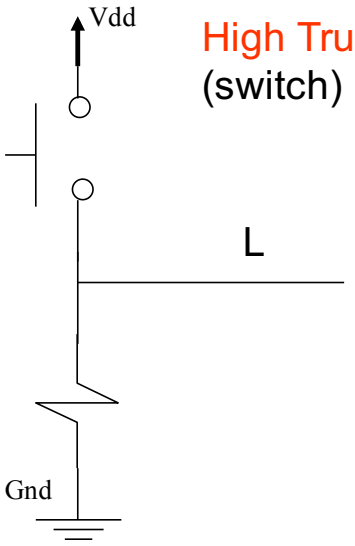
The Physical World



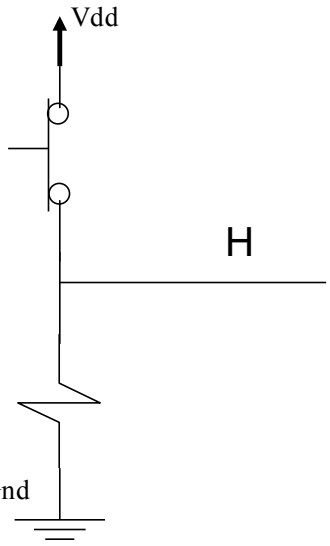
When a button is pressed, it is asserted (true). However, its physical construction may output this as a LOW VOLTAGE (low true)

- Person pressing button doesn't know or care that a low voltage is output
- They just know they have ASSERTED the button

Examples of high, low signals



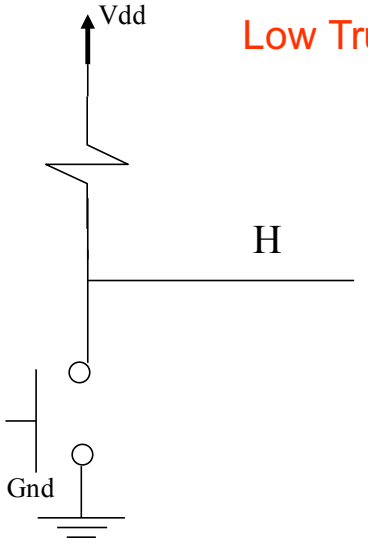
High True button (switch)



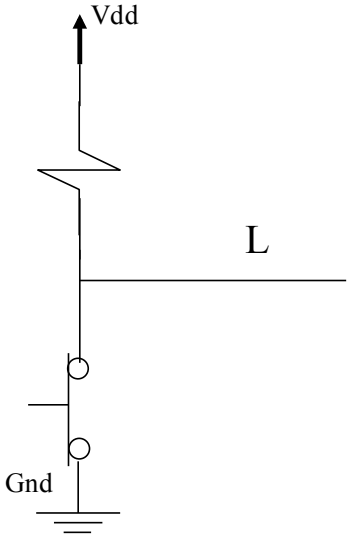
Switch open (negated), output is L

Switch closed (asserted), output is H

Examples of high, low signals



Low True switch



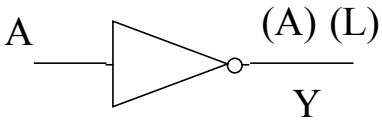
Switch open (negated), output is H

Switch closed (asserted), output is L

7404 Logic Gate

Fixed Logic

A	Y
L	H
H	L

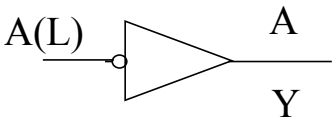


Mixed Logic

A	A (L)
0	0
1	1

Buffer that converts high true input to low true output

A(L)	Y
L	H
H	L



A(L)	A
1	1
0	0

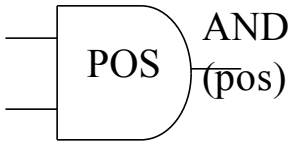
Buffer that converts low true input to high true output

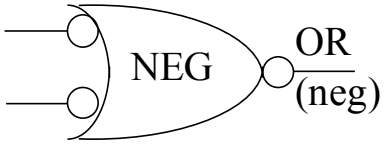
Electronically Identical Circuits – Matter of Interpretation

TTL Device

A B	Y
L L	L
L H	L
H L	L
H H	H

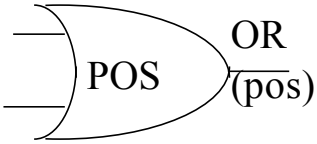
7408

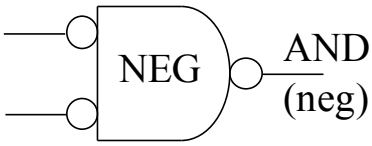




A B	Y
L L	L
L H	H
H L	H
H H	H

7432

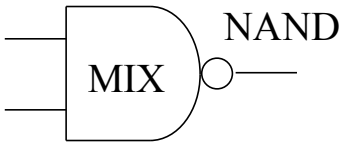


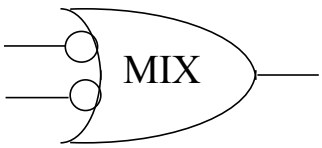


DeMorgan's Theorem

A B	Y
L L	H
L H	H
H L	H
H H	L

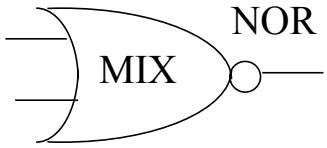
7400

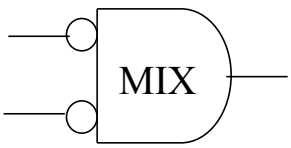




A B	Y
L L	H
L H	L
H L	L
H H	L

7402

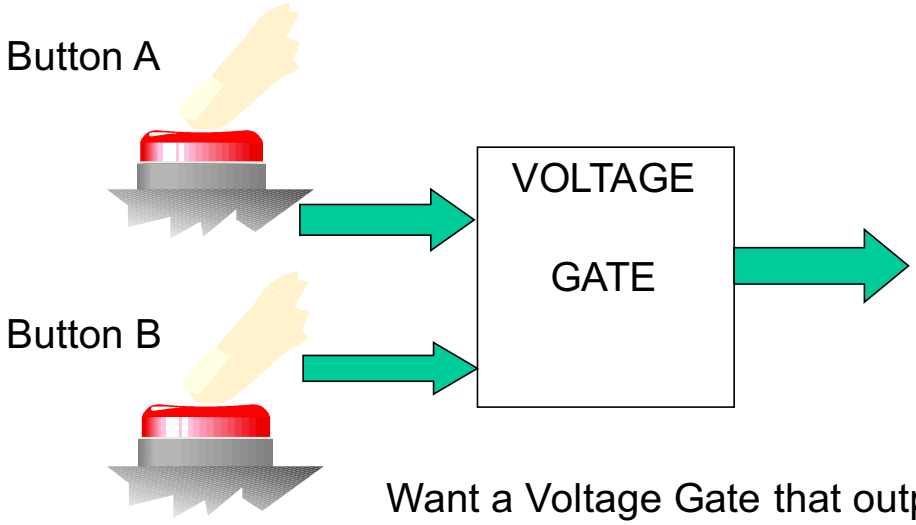




Gate Summaries

THE Problem

Have two buttons, each button outputs a low voltage (L) when pressed.

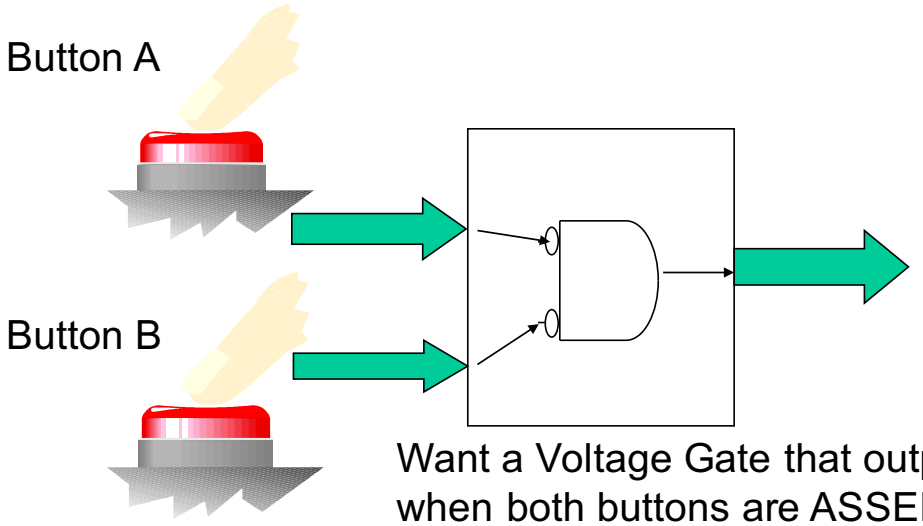


Want a Voltage Gate that outputs a 'H' when both buttons are ASSERTED.

What does the word "both" mean?

THE Problem (solution)

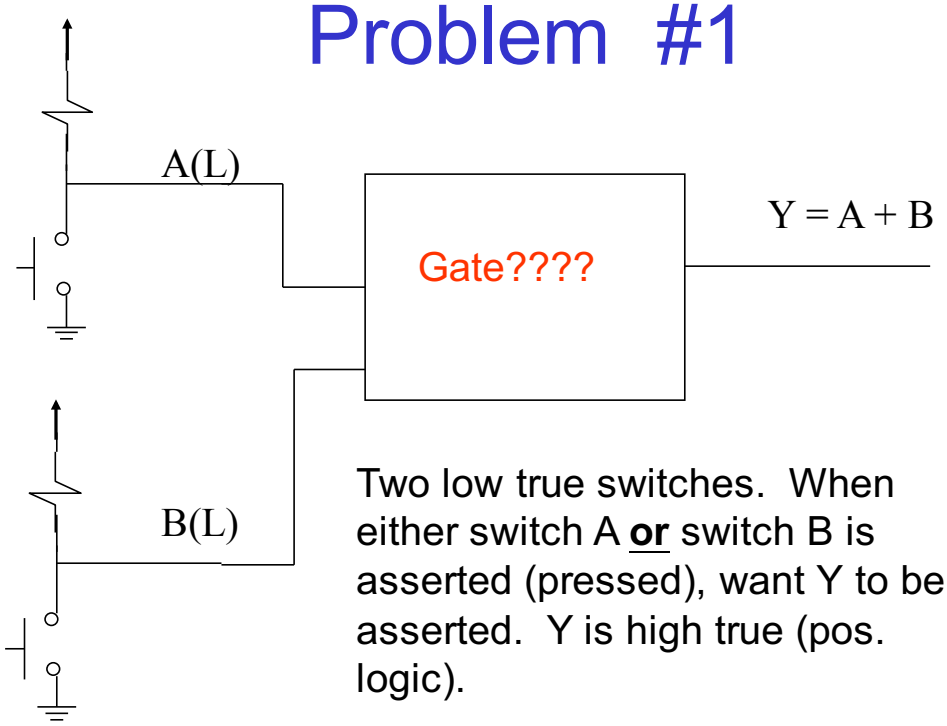
Have two buttons, each button outputs a low voltage (L) when pressed.



Want a Voltage Gate that outputs a 'H' when both buttons are ASSERTED.

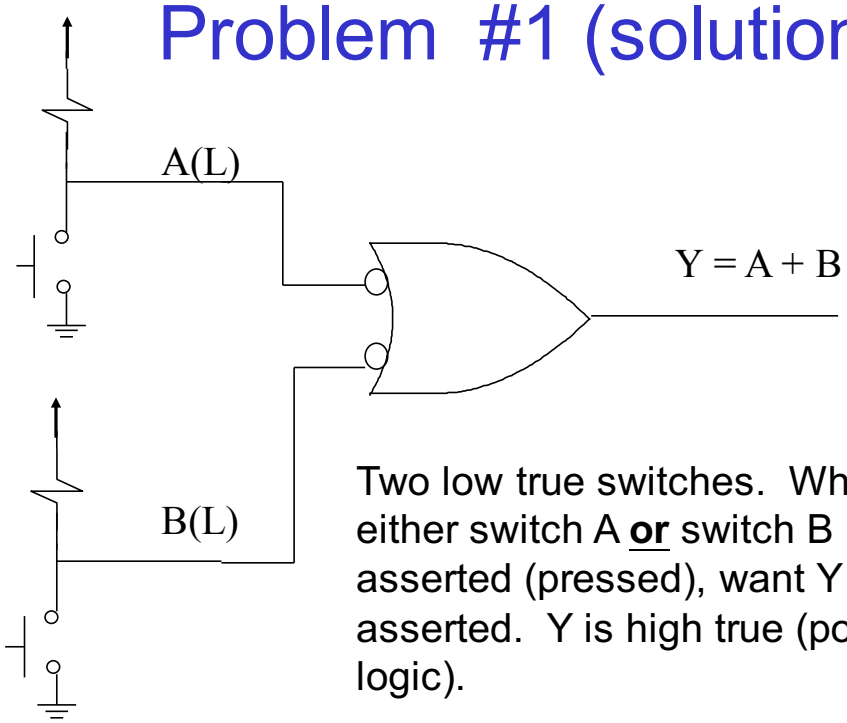
When both A **and** B asserted (low true), output is asserted (high true)

Problem #1



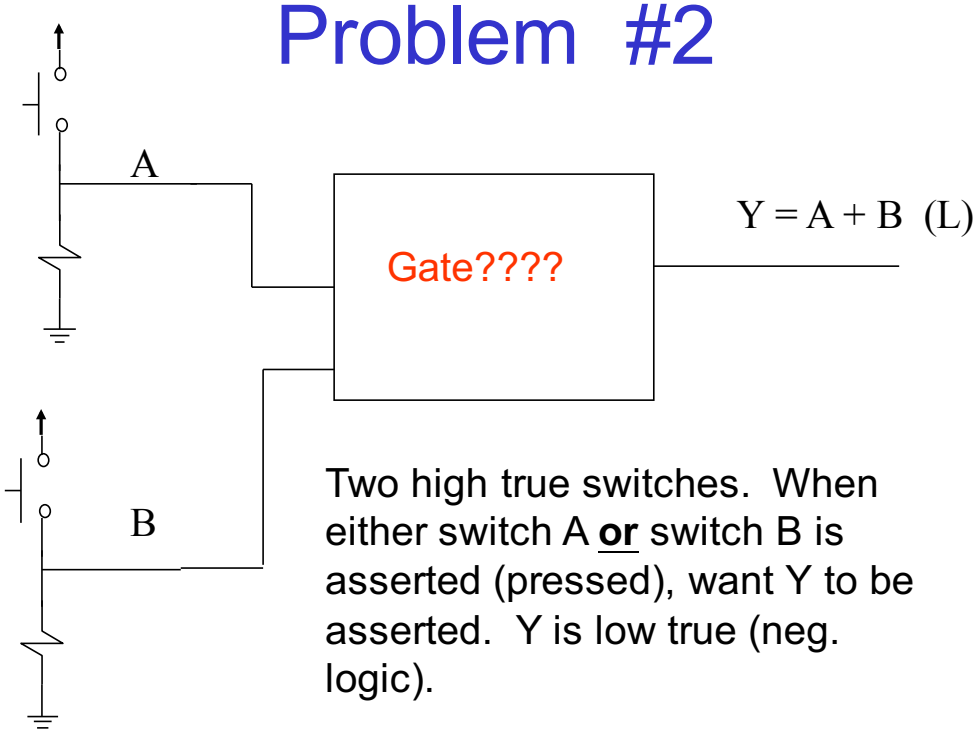
Two low true switches. When either switch A **or** switch B is asserted (pressed), want Y to be asserted. Y is high true (pos. logic).

Problem #1 (solution)



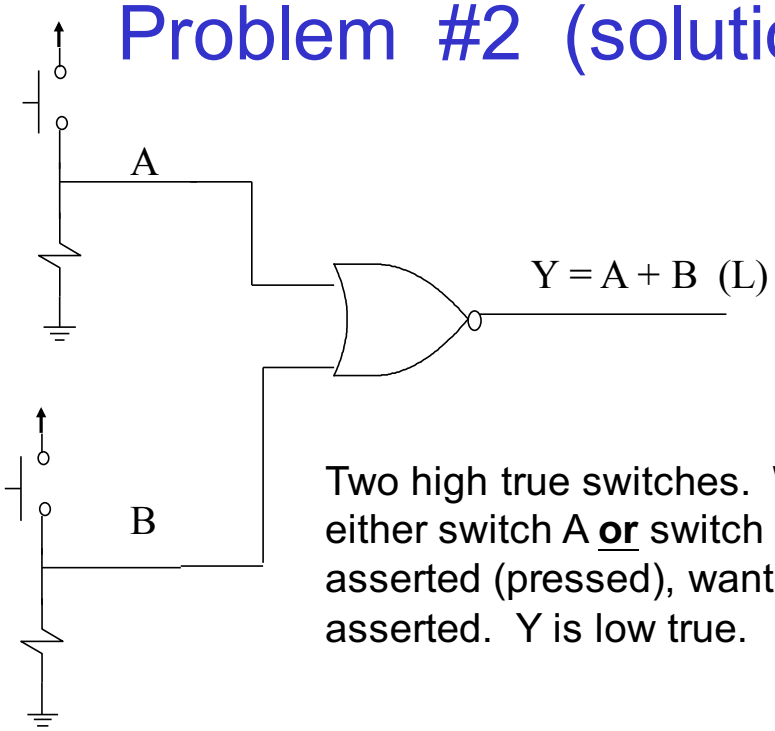
Two low true switches. When either switch A **or** switch B is asserted (pressed), want Y to be asserted. Y is high true (pos. logic).

Problem #2



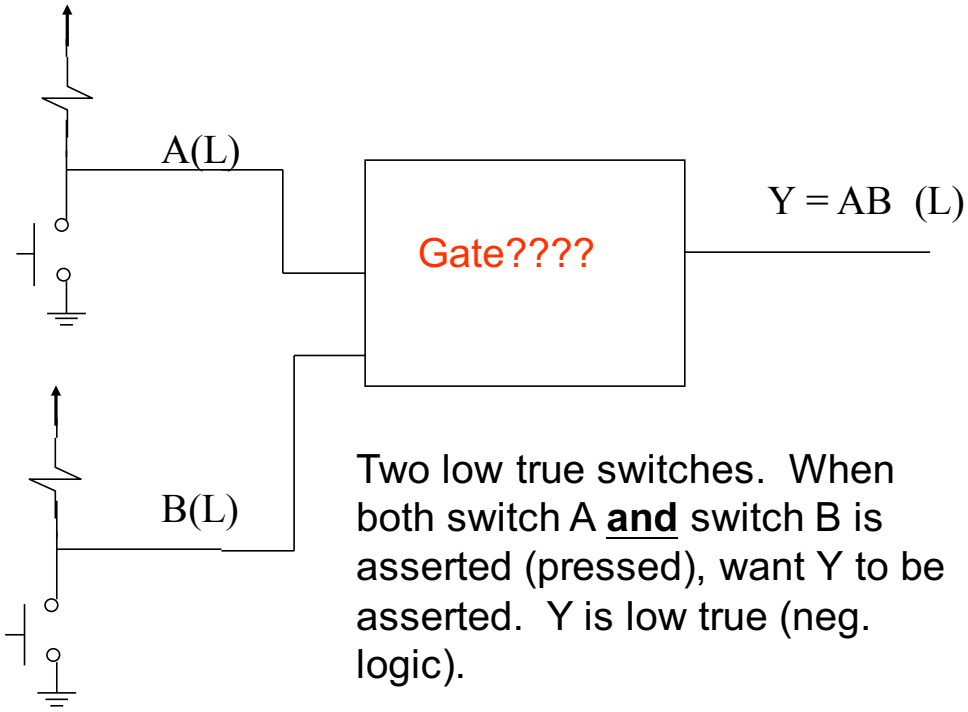
Two high true switches. When either switch A **or** switch B is asserted (pressed), want Y to be asserted. Y is low true (neg. logic).

Problem #2 (solution)



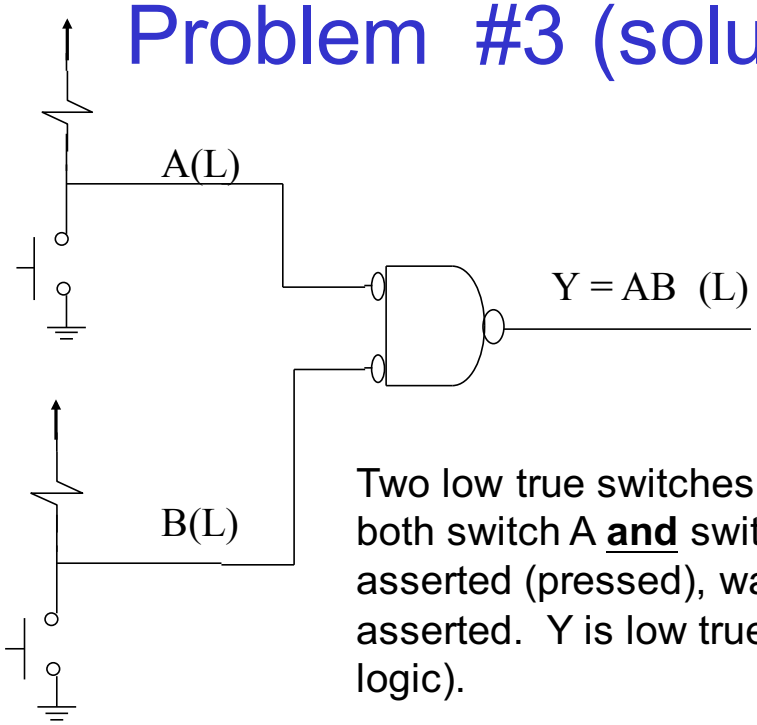
Two high true switches. When either switch A **or** switch B is asserted (pressed), want Y to be asserted. Y is low true.

Problem #3



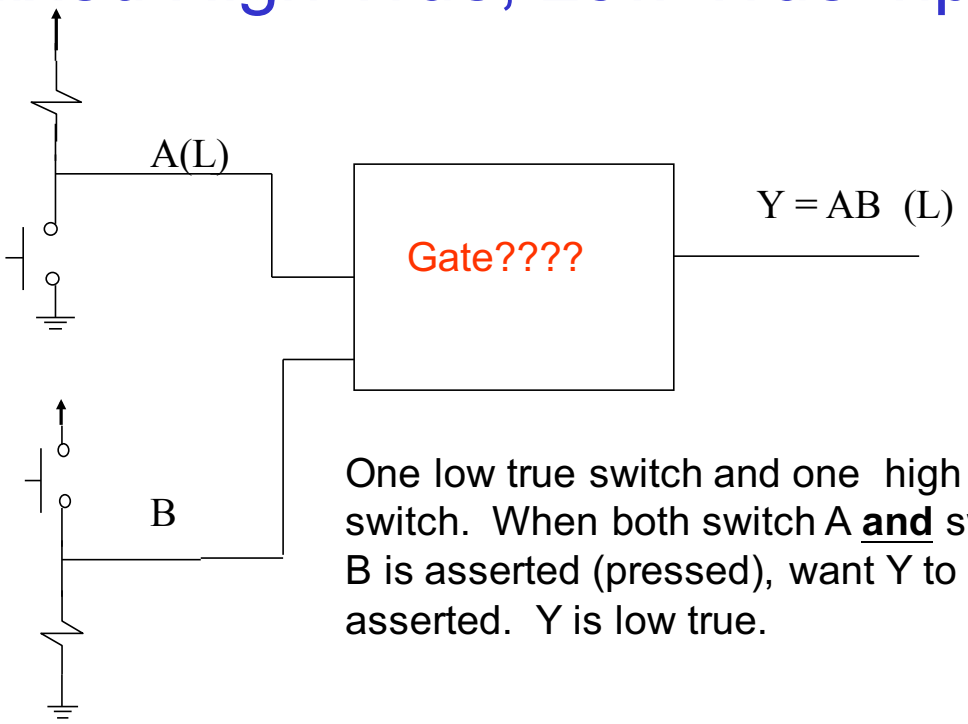
Two low true switches. When both switch A **and** switch B is asserted (pressed), want Y to be asserted. Y is low true (neg. logic).

Problem #3 (solution)



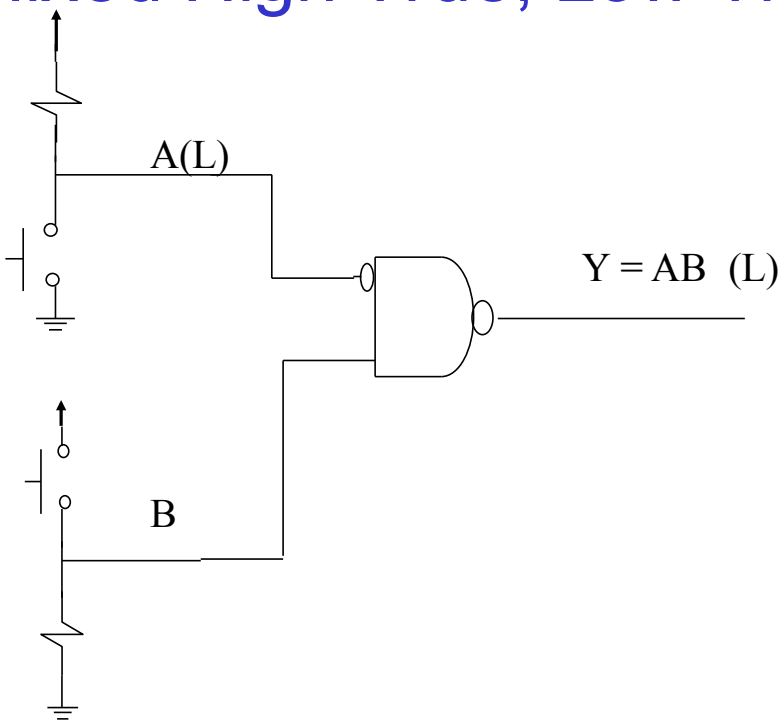
Two low true switches. When both switch A **and** switch B is asserted (pressed), want Y to be asserted. Y is low true (neg. logic).

Mixed High True, Low True Inputs



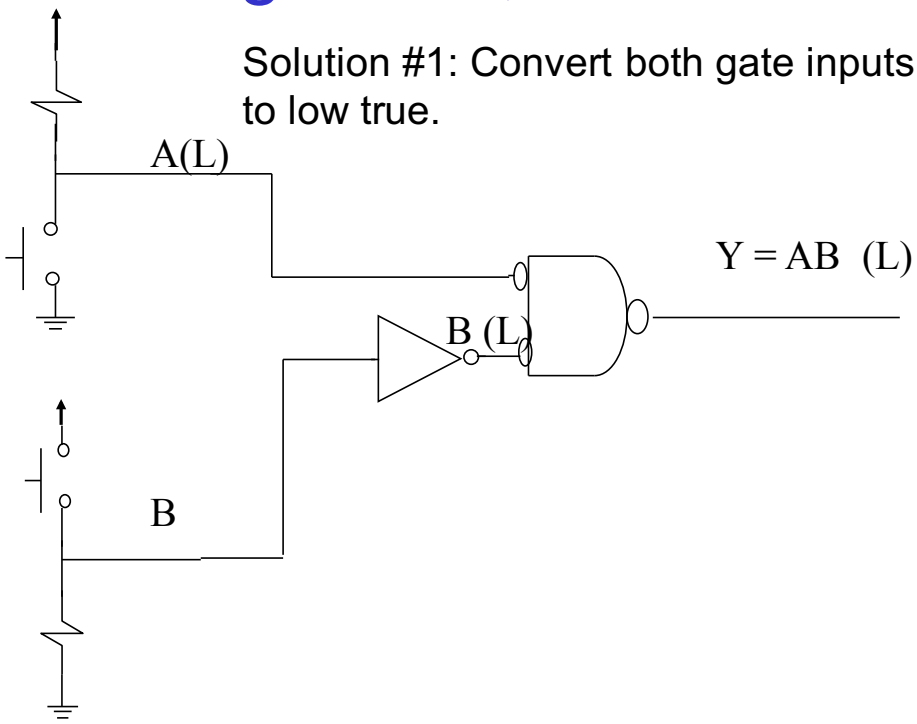
One low true switch and one high true switch. When both switch A **and** switch B is asserted (pressed), want Y to be asserted. Y is low true.

Mixed High True, Low True Inputs



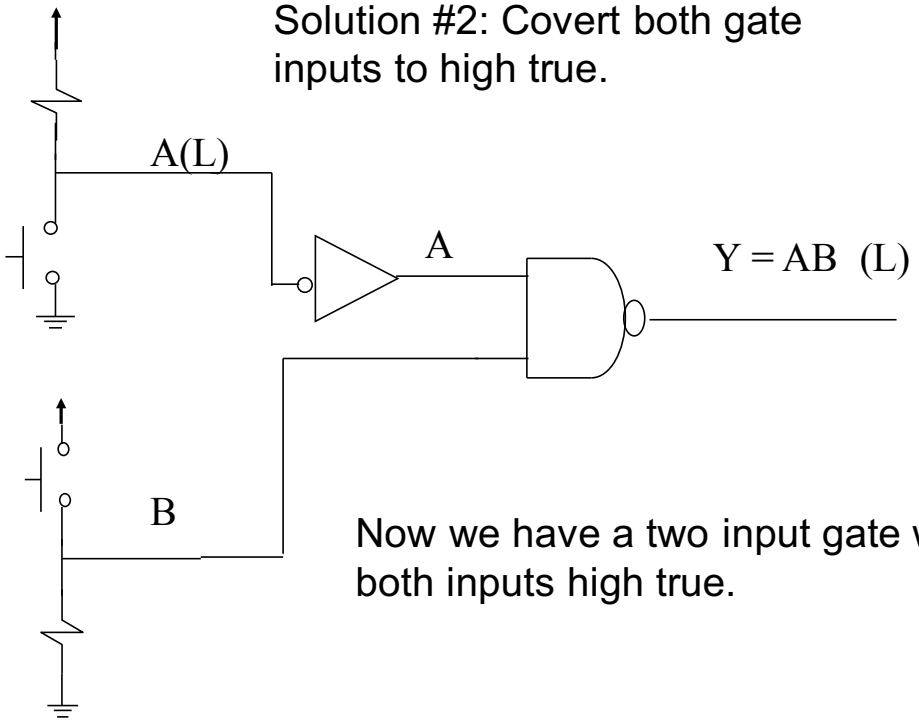
Mixed High True, Low True Inputs

Solution #1: Convert both gate inputs to low true.



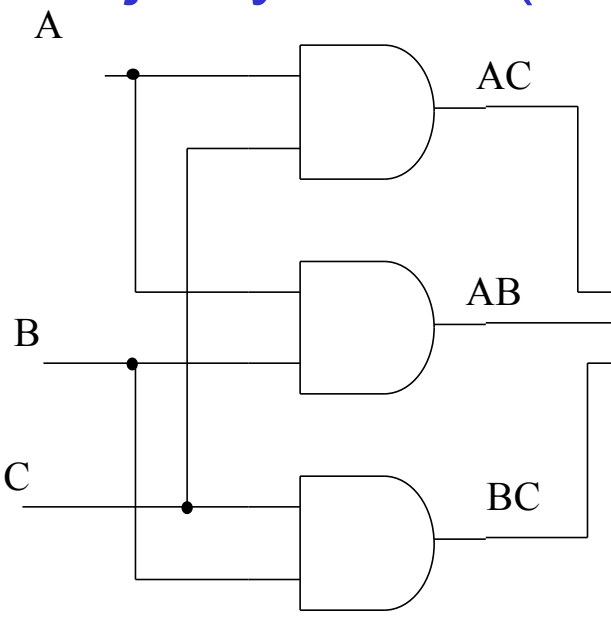
Mixed High True, Low True Inputs

Solution #2: Covert both gate inputs to high true.



Now we have a two input gate with both inputs high true.

Majority Gate (and-or) form



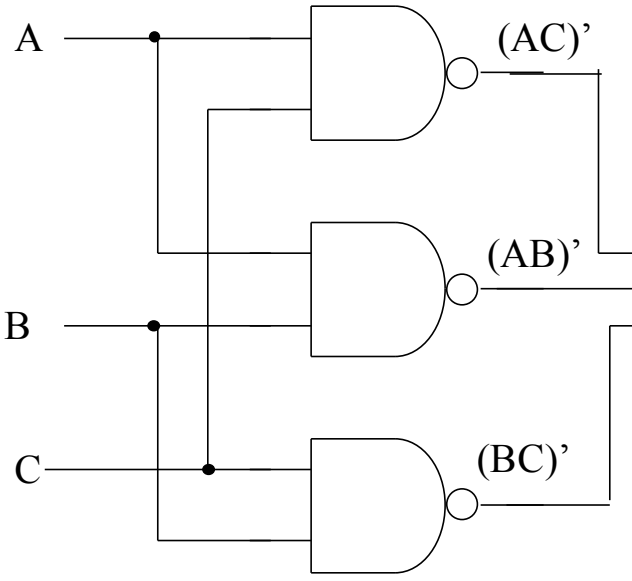
$F = AB + AC + BC$

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

Majority Gate (nand-nand) form

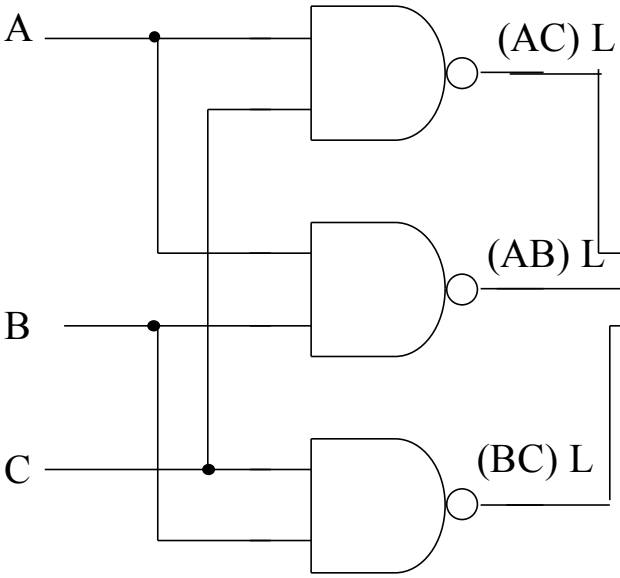
$$F = \overline{(\overline{AC})(\overline{AB})(\overline{BC})}$$
$$= \overline{\overline{AC} + \overline{AB} + \overline{BC}}$$
$$= AC + AB + BC$$

by DeMorgan's theorem



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

Majority Gate (mixed logic) form



$$F = \overline{AB} + \overline{AC} + \overline{BC}$$
$$= AB + AC + BC$$

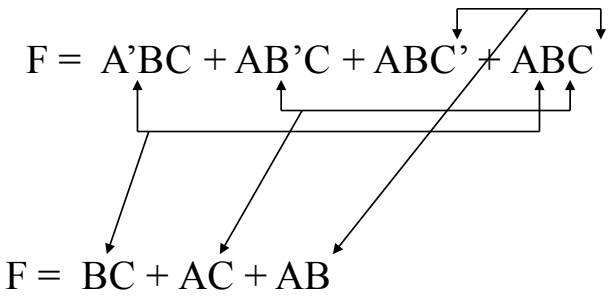
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Boolean Minimization

Reduce a Boolean equation to fewer terms in SOP form - hopefully, this will result in using less gates to implement the Boolean equation.

Pencil-Paper: Algebraic techniques, K-maps
Automated: Many powerful algorithms exist

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



Find Boolean adjacencies to minimize equation; eliminate redundant term

Map Method

Graphical Aid for minimization - used to visualize Boolean adjacencies

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

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

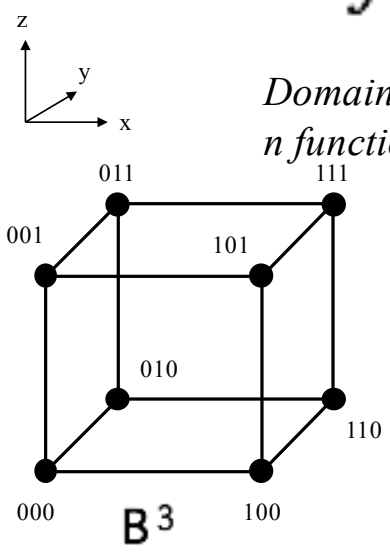
$F = BC + AC + AB$

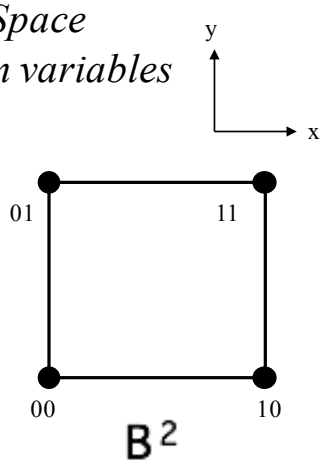
Boolean Functions to Represent Logic Circuit Behavior

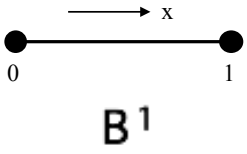
$$f : B^n \rightarrow B^m$$

Domain Space
n function variables

Range Space
m function outputs

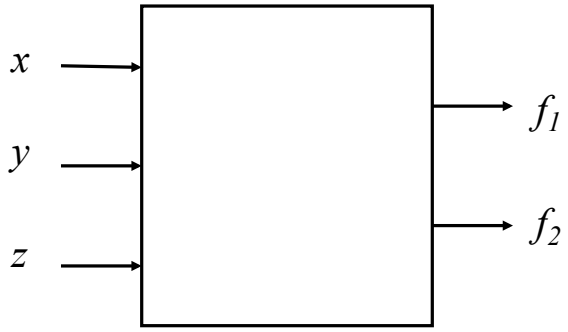






Example Boolean Function

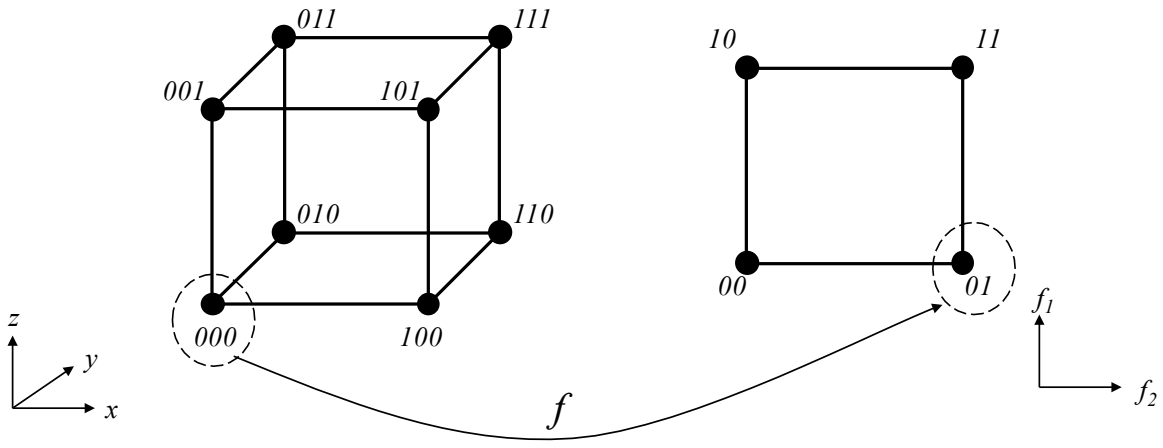
$$f : B^3 \rightarrow B^2$$



x	y	z	f_1	f_2
0	0	0	0	1
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	0	1

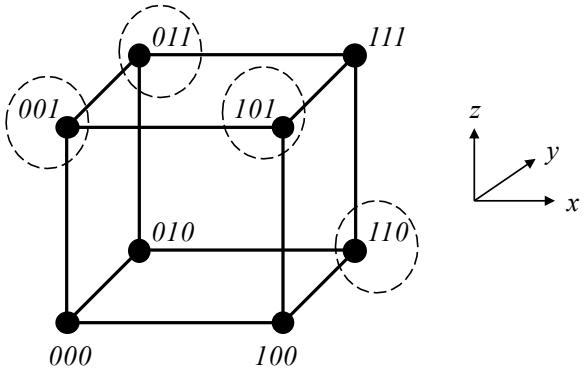
Example Boolean Function

$$f : B^3 \rightarrow B^2$$



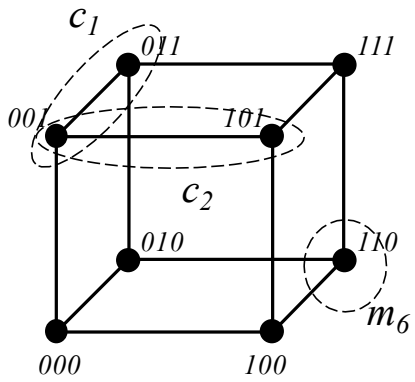
Example Boolean Function

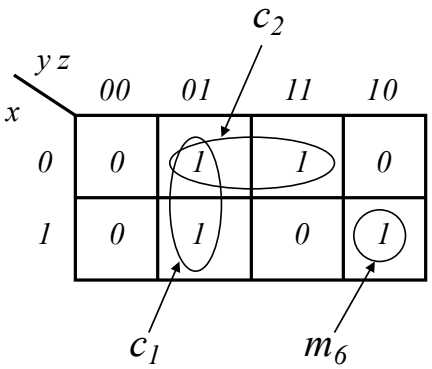
(*single output function*)



- Represent Function Range by Circling Points in Domain Space
- Each Point is a Cube, B^0
- Each Point is a *minterm* of f_1

Combining Cubes





- Adjacent Cubes may be Combined
- Combined Cube has Higher Dimension
- Combined Cube is a *product term*
- This is Same as K-Map Operation