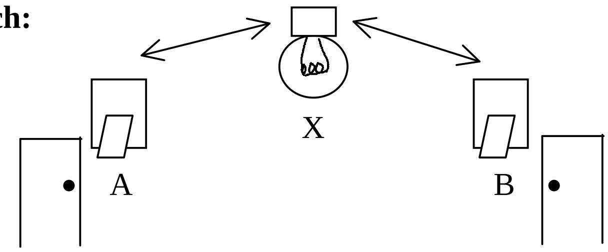
- **3-way light switch:**



  - want either switch to be able to turn light ON/OFF independent of position of other switch

    X = { 1, light ON ; 0, OFF } , A = { 1, switch UP ; 0, switch DOWN } , B = { 1, UP ; 0, DOWN }

  - assume when first installed X = 0, A = 0, and B = 0
  - want:
    - $A = 0$ and $B = 0 \Rightarrow X = 0$
    - $A = 0 \rightarrow 1$ and $B = 0 \Rightarrow X = 1$
    - $A = 1$ and $B = 0 \rightarrow 1 \Rightarrow X = 0$
    - $A = 1 \rightarrow 0$ and $B = 1 \Rightarrow X = 1$
    - $A = 0$ and $B = 1 \rightarrow 0 \Rightarrow X = 0$

    $\Rightarrow$

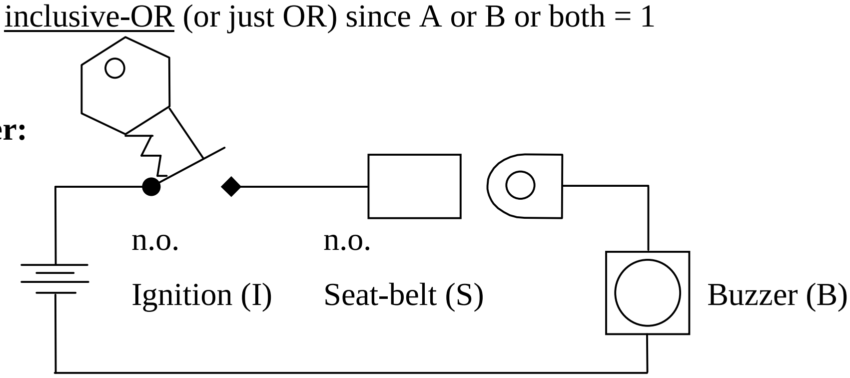
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$\Rightarrow$ circuit (6)

  - used double pole switches to realize (6) : $X = \overline{A} \bullet B + A \bullet \overline{B}$
  - (6) termed <u>exclusive-OR</u> (XOR) since A or B = 1 but not both

    (7) termed <u>inclusive-OR</u> (or just OR) since A or B or both = 1

- **Seat-belt buzzer:**



  - want:

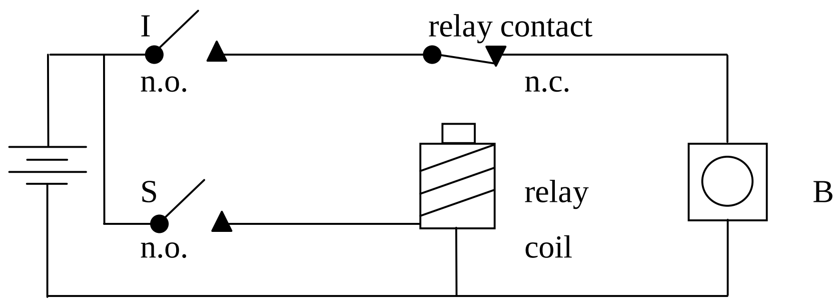
    I = { 1, ignition ON ; 0, OFF } , S = { 1, fastened ; 0, unfastened } , B = { 1, sound ; 0, quiet }

I	S	B
0	0	0
0	1	0
1	0	1
1	1	0

⇒ circuit (2) ⇒ $B = I \bullet \overline{S}$ ⇒ seat-belt needs to be n.c. switch

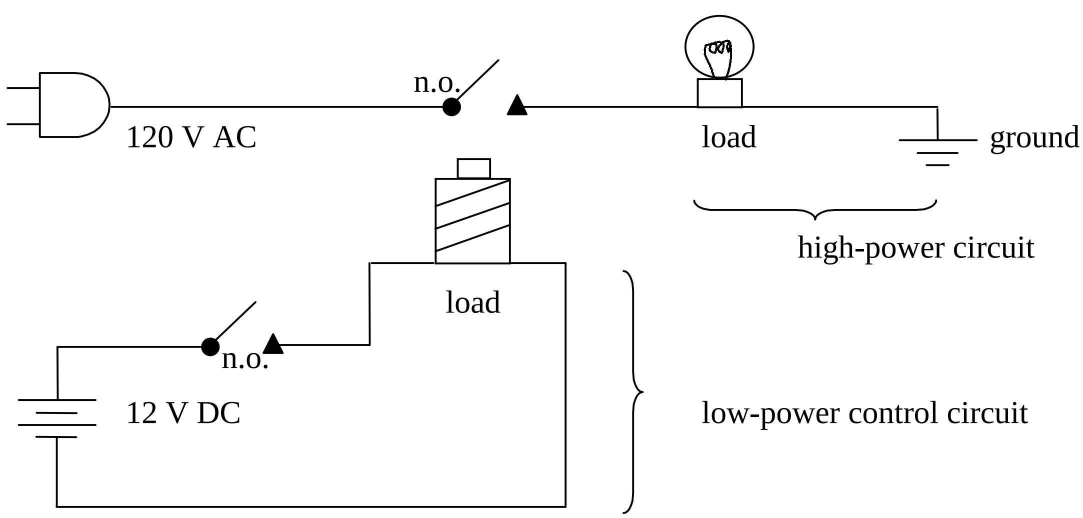
- physically, S best as n.o., but want it to be n.c. in control circuit

  ⇒ can use relay:



- both B and relay coil are loads
- when S actuated, current flows through relay coil, which becomes a magnet that actuates relay contact

- **Relays:**
  - up to now, all inputs have been mechanical switches that directly made or broke the circuit controlling the load
  - relays can be used to indirectly control higher power circuit:

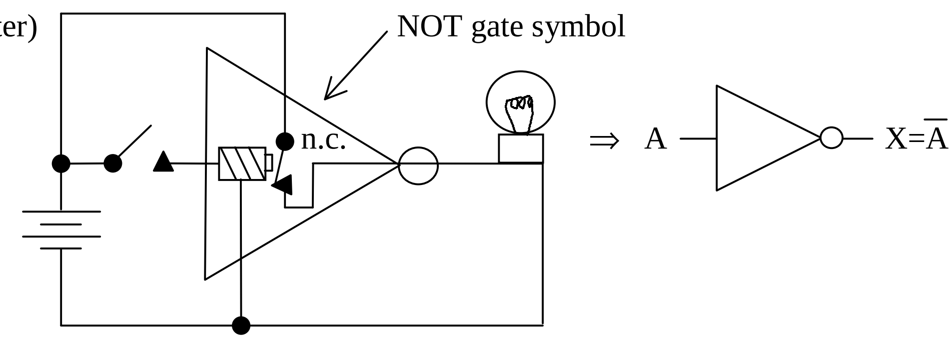


- relays can be normally open (n.o.) or normally closed (n.c.)
- use of relays eliminates need for complex multiple pole switches and wiring

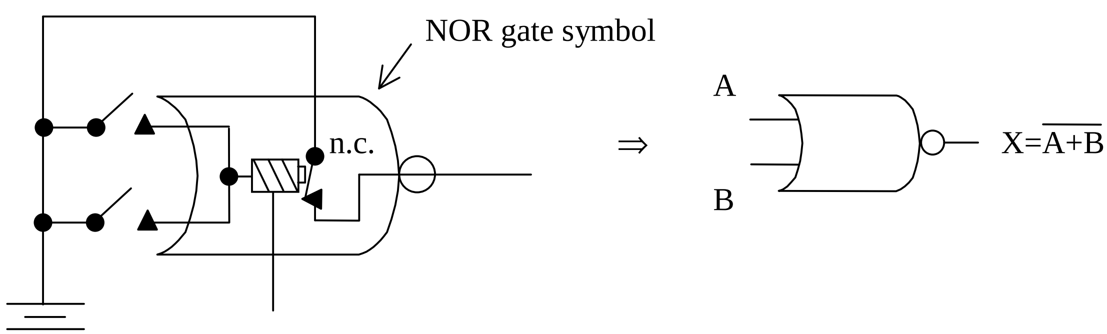
- **Logic Gates:** (see handout)
  - in seat-belt control circuit, relay used to convert n.o. mechanical switch to n.c. switch

    ⇒ performing Logical NOT ⇒ inverting signal ⇒ <u>inverter</u>
  - **NOT gate**:

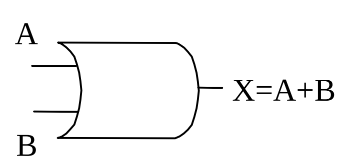
    (inverter)



  - **NOR gate**: when two (or more) inputs connected in parallel to single relay of NOT gate, get NOR gate



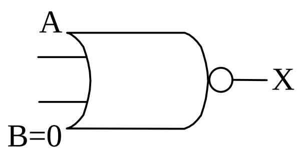
  - if n.o. relay contact used in NOR gate ⇒ **OR gate** =



  - but, unlike OR gate, NOR gate (and NAND gate) are <u>universal gates</u>

    ⇒ any Boolean logic circuit can be realized using just NOR gates (or just NAND gates)

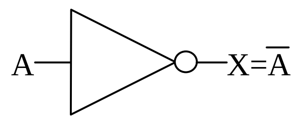
- NOT gate via NOR gate:



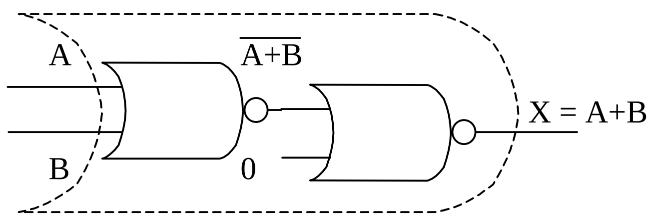
$\Rightarrow$

A	B	X
0	0	1
1	0	0

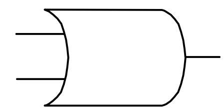
$\Rightarrow$



- OR gate via NOR gates:



$\Leftrightarrow$



- AND gate via NOR gates: not immediately obvious how to construct

  $\Rightarrow$ use <u>Theorems of Boolean Algebra</u>

- **Theorems of Boolean Algebra** (see handout)
  - used to manipulate Boolean expressions
  - developed by George Boole in 1850s (Laws of Thought)
  - in 1938, Claude Shannon saw one-to-one correspondence between Boolean expressions and switching circuits

    $\Rightarrow$ Boolean algebra can be used to simplify logic control circuits
  - AND gate via NOR gates + Boolean algebra:

    Want: $X = A \bullet B$

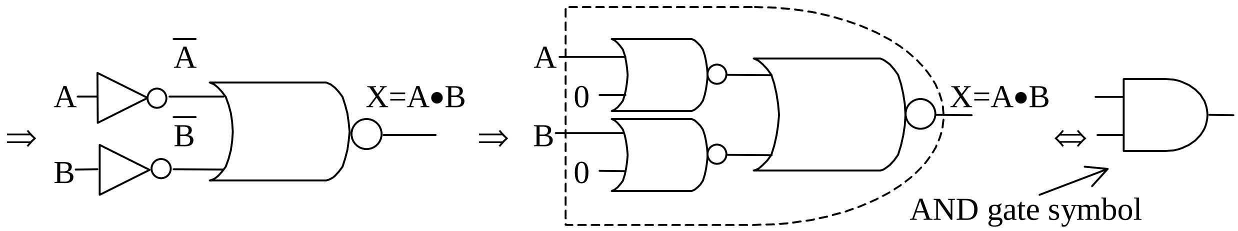
    $= \overline{\overline{A}} \bullet \overline{\overline{B}}$ (by Negation Th.)

    Let $A' = \overline{A}$ and $B' = \overline{B}$

    $X = \overline{A'} \bullet \overline{B'}$

    $= \overline{A' + B'}$ (by DeMorgan 2 — left side of DeMorgan 2 is NOR)

    $= \overline{\overline{A} + \overline{B}}$



- Theorems also give NOT and OR gates via NOR gates:

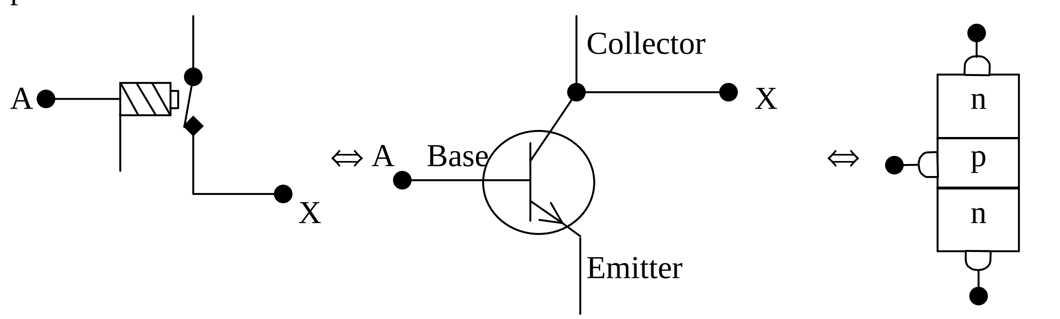
NOT:

$$X = \overline{A}$$
$$= \overline{A} \bullet 1 \quad \text{(by Char. 2)}$$
$$= \overline{A} \bullet \overline{0} \quad \text{(1=0 \& 0=1)}$$
$$= \overline{A + 0} \quad \text{(by DeMorgan 2)}$$

OR:

$$X = A + B$$
$$= \overline{\overline{A}} + \overline{\overline{B}} \quad \text{(by Neg.)}$$
$$= \overline{\overline{A} \bullet \overline{B}} \quad \text{(by DeMorgan 1)}$$
$$= \overline{\overline{A} \bullet \overline{B} \bullet 1} \quad \text{(by Char. 2)}$$
$$= \overline{\overline{A} \bullet \overline{B} \bullet \overline{0}} \quad \text{(1=0)}$$
$$= \overline{\overline{A} \bullet \overline{B} + 0} \quad \text{(by DeMorgan 2)}$$
$$= \overline{\overline{(A + B)} + 0} \quad \text{(by DeMorgan 2)}$$

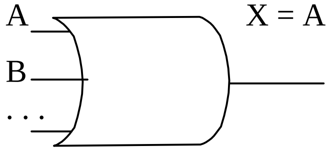
- **Transistor Logic:**
  - Why are 3 NOR gates used instead of 1 AND gate?
  - NOR gate easy to make using transistor
  - transistors used instead of relays in all control applications except to switch high power circuits



n.c. relay n-p-n transistor semiconductor

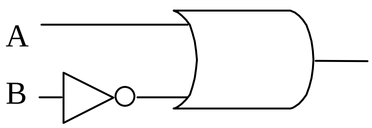
  - for control system design, can think in terms of relays

- **Multi-input gates:** logic gates can have more than two inputs

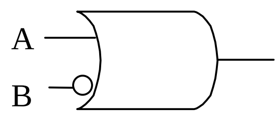


$X = A + B + \ldots$ — can all be realized using multi-input NOR or NAND gates

- **Input negation:**

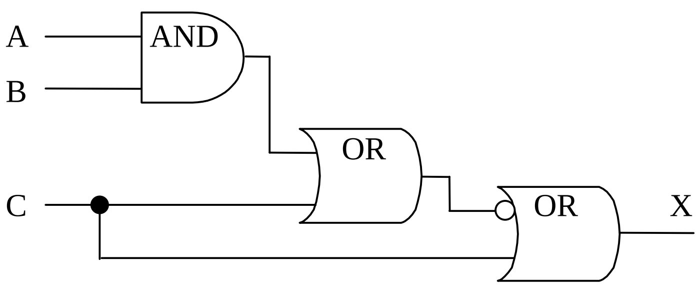


⇔

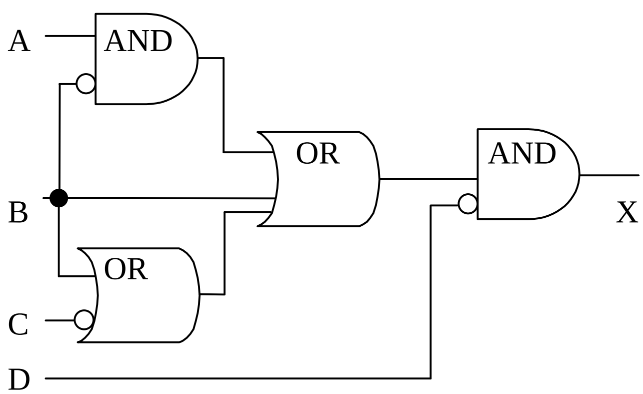


⇔ $X = A + \overline{B}$

- **Nesting:** when output from gate used as input to next, equivalent to parenthesis in Boolean expression (AND higher precedence than OR)
- **Logic gate network** ⇒ Boolean expression:



⇔ $X = (\overline{A \bullet B + C}) + C$



⇔ $X = [(A \bullet \overline{B}) + B + (B + \overline{C})] \bullet \overline{D}$

$= (A \bullet \overline{B} + B + B + \overline{C}) \bullet \overline{D}$

Theorems and Laws of Boolean Algebra

CHARACTERISTIC THEOREMS

1. $X \bullet 0 = 0$
2. $X \bullet 1 = X$
3. $X + 0 = X$
4. $X + 1 = 1$

COMMUTATIVE LAW

1. $X + Y = Y + X$
2. $X \bullet Y = Y \bullet X$

ASSOCIATIVE LAW

1. $X + Y + Z = X + (Y + Z)$
   $= (X + Y) + Z$
2. $X \bullet Y \bullet Z = X \bullet (Y \bullet Z)$
   $= (X \bullet Y) \bullet Z$

DISTRIBUTIVE LAW

1. $X \bullet Y + X \bullet Z = X(Y + Z)$
2. $(X + Y)(W + Z) = XW + XZ + YW + YZ$

IDEMPOTENT THEOREMS

1. $X \bullet X = X$
2. $X + X = X$

NEGATION THEOREM

$\overline{(\overline{X})} = X$

INCLUSION THEOREMS

1. $X \bullet \overline{X} = 0$
2. $X + \overline{X} = 1$

ABSORPTIVE LAWS

1. $X + XY = X$
2. $X(X + Y) = X$

REFLECTIVE THEOREMS

1. $X + \overline{X}Y = X + Y$
2. $X(\overline{X} + Y) = XY$
3. $XY + \overline{X}YZ = XY + YZ$

CONSISTENCY THEOREM

1. $XY + X\overline{Y} = X$
2. $(X + Y)(X + \overline{Y}) = X$

DEMORGAN'S LAWS

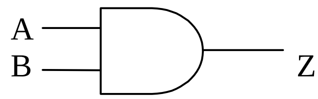
1. $\overline{XY} = \overline{X} + \overline{Y}$
2. $\overline{X + Y} = \overline{X}\,\overline{Y}$

											≡				⊃		
Logic:			$\wedge$		A		B		$\vee$	↓	↔	$\overline{B}$	←	$\overline{A}$	→	↑	
Boolean:			•					⊕	+		⊙						
Name:			AND					XOR	OR	NOR	XNOR	NOT		NOT		NAND	
A	B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		short circuit	$A \bullet B$	$A \bullet \overline{B}$	A	$\overline{A} \bullet B$	B	$A \bullet \overline{B} + \overline{A} \bullet B$	$A+B$	$\overline{A} \bullet \overline{B}$, $\overline{A+B}$	$A \bullet B + \overline{A} \bullet \overline{B}$	$\overline{B}$	$A+\overline{B}$	$\overline{A}$	$\overline{A}+B$	$\overline{A}+\overline{B}$, $\overline{A \bullet B}$	no control

DIGITAL LOGIC GATES AND ASSOCIATED LOGICAL OPERATIONS FOR BINARY VARIABLES

Name | **Symbol** | **Logical operation** | **Truth table**

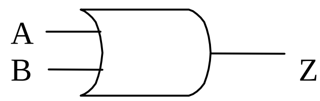
AND



$Z = A \bullet B$

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

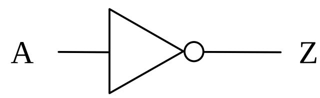
OR



$Z = A + B$

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

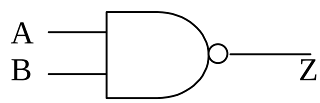
NOT



$Z = \overline{A}$

A	Z
0	1
1	0

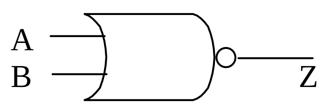
NAND



$Z = \overline{A \bullet B}$

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

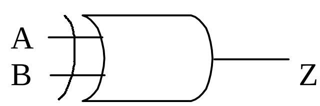
NOR



$Z = (\overline{A + B})$

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

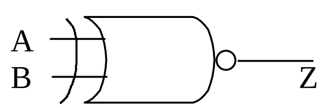
XOR



$Z = A \oplus B$

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

XNOR



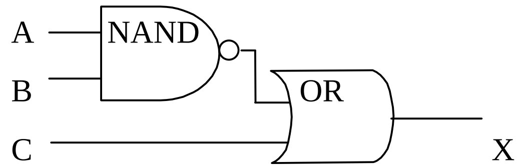
$Z = A \odot B$

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

- Simplifying Boolean expression $\Leftrightarrow$ reducing number of logic gates

$$X = (\overline{A \bullet B + C}) + C$$
$$= (\overline{A \bullet B} \bullet \overline{C}) + C \quad \text{(by DeMorgan 2)}$$
$$= \overline{A \bullet B} + C \quad \text{(by Reflective 1)}$$

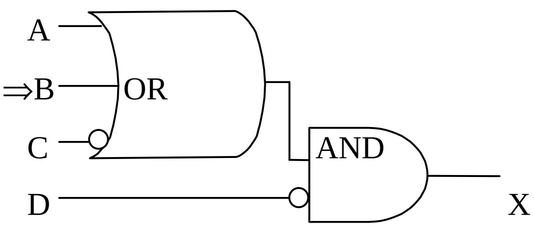
$\Rightarrow$



($\downarrow$ from 3 to 2 gates)

$$X = (A \bullet \overline{B} + \underbrace{B + B} + \overline{C}) \bullet \overline{D}$$
$$= (A \bullet \overline{B} + B + \overline{C}) \bullet \overline{D} \quad \text{(by Idempotent 2)}$$
$$= (A + B + \overline{C}) \bullet \overline{D} \quad \text{(by Reflective 1)}$$

$\Rightarrow$



($\downarrow$ from 4 to 2)

- **Canonical Sum-of-Products Form**
  - Used to realize any logic control circuit from its truth table

1. For each circuit, construct its truth table to relate all possible inputs to desired output of circuit:

Row	A	B	X
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

} desired output

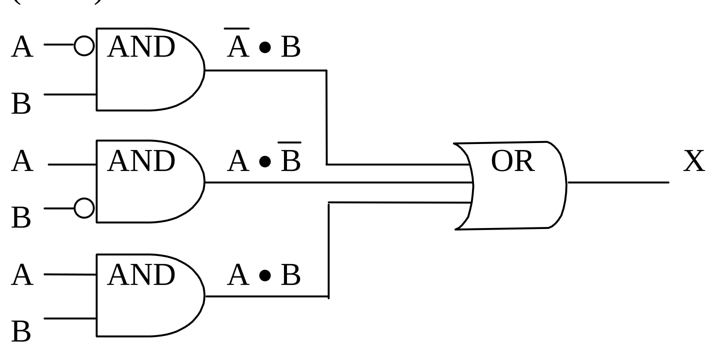
2. For each row where X = 1, AND together inputs, where 0 inputs are NOTed

row 0: $(X \neq 1)$

row 1:

row 2:

row 3:



3. OR together AND gates from step 2

   $X = \overline{A} \bullet B + A \bullet \overline{B} + A \bullet B$

4. Use Theorems of Boolean Algebra to simplify if possible

$= \overline{A} \bullet B + A \bullet (\overline{B} + B)$	(by Distrib. 1)
$= \overline{A} \bullet B + A \bullet (1)$	(by Inclusion 2)
$= \overline{A} \bullet B + A$	(by Char. 2)
$= B + A$	(by Reflect. 1)
$= A + B \Rightarrow$ OR	(by Commut. 1)

- termed "Sum-of-Products" since OR-ing together (summing) AND-ed inputs (products)
- Example: XOR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$\Rightarrow X = \overline{A} \bullet B + A \bullet \overline{B}$, can't simplify

- Example: Seat belt and Door-Open Buzzer

$I = \begin{cases} 1, \text{ ignition ON} \\ 0, \text{ OFF} \end{cases}$, $S = \begin{cases} 1, \text{ seat belt fastened} \\ 0, \text{ unfastened} \end{cases}$

$D = \begin{cases} 1, \text{ door closed} \\ 0, \text{ door open} \end{cases}$, $B = \begin{cases} 1, \text{ sound} \\ 0, \text{ quiet} \end{cases}$

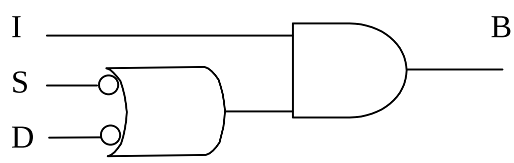
I	S	D	B	
0	0	0	0	quiet when I = 0
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	0	

$B = I \bullet \overline{S} \bullet \overline{D} + I \bullet \overline{S} \bullet D + I \bullet S \bullet \overline{D}$

$= I \bullet (\overline{S}\bullet\overline{D} + \overline{S}\bullet D + S\bullet\overline{D})$ (by Distrib. 1)

$= I \bullet (\overline{S} + S\bullet\overline{D})$ (by Consist. 1)

$= I \bullet (\overline{S} + \overline{D})$ (by Reflect. 1)



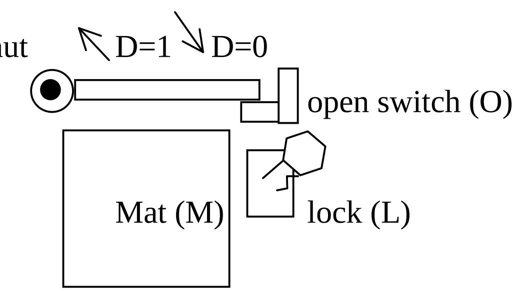
- Example: Automatic Door and Lock

D = 1, door opening; 0, door closing

O = 1, door not shut; 0, door shut

M = 1, someone on mat; 0, no one on mat

L = 1, unlocked; 0, locked



Operation: Want door to open if someone on mat and door unlocked.
If locked, want it to stay open or shut independent of mat

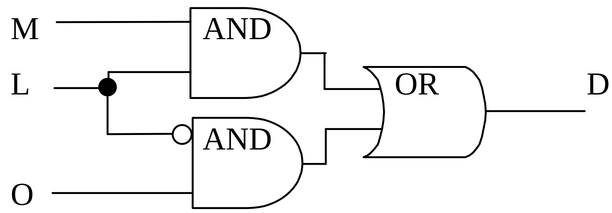
	M	L	O	D	
Stay shut	0	0	0	0	start 0
	0	0	1	1	unlock 1
	0	1	0	0	
stay open	0	1	1	0	closing 5
	1	0	0	0	open 2
	1	0	1	1	
	1	1	0	1	opening 3
	1	1	1	1	shut 4

$D = \overline{M}\bullet\overline{L}\bullet O + M\bullet\overline{L}\bullet O + M\bullet L\bullet\overline{O} + M\bullet L\bullet O$

$= (\overline{M}+M)\bullet\overline{L}\bullet O + M\bullet L\bullet(\overline{O}+O)$ (by Distrib. 1)

$= (1)\bullet\overline{L}\bullet O + M\bullet L\bullet(1)$ (by Inclus. 2)

$= \overline{L}\bullet O + M\bullet L$


M
AND
L
OR
D
AND
O