Digital Logic Circuits

- Logic Gates
- Truth Tables
- Analysis of logic circuits

Logic Gates

Logical operations can be physically implemented using semiconductor switches (transistors). They are represented symbolically by logic gates. These are the building blocks of digital logic circuits.

NOT

X	$\overline{X}$
0	1
1	0

AND

X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

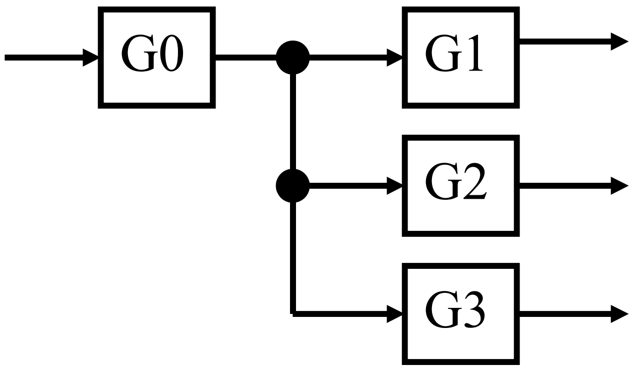
OR

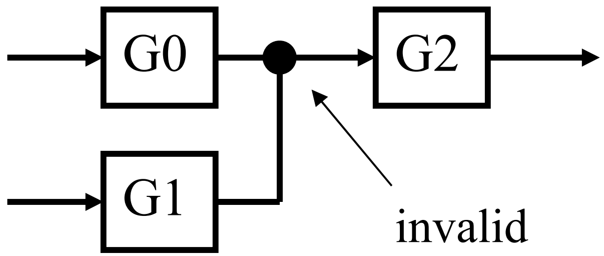
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

A truth table can be used to tabulate the output of the gate for every possible combination of its input values.

Logic Gate Networks

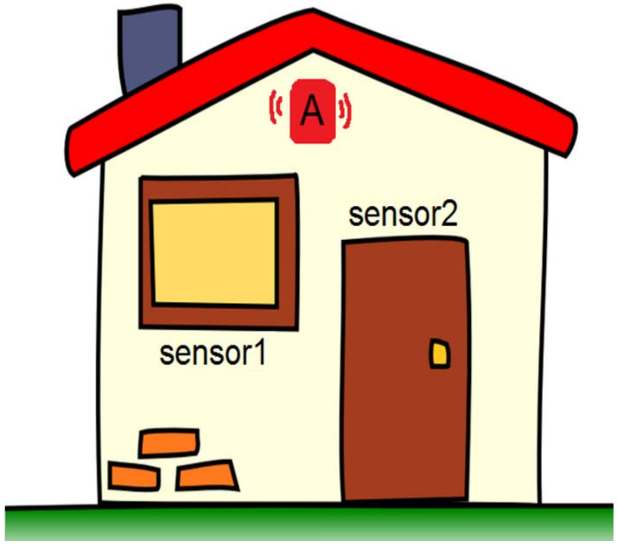
- Logic gates can be connected together to implement a logic circuit.
- A gate output can be connected to (**drive**) more than one input.
- Gate outputs must not be connected together.
  (the exception is tri-state gates)





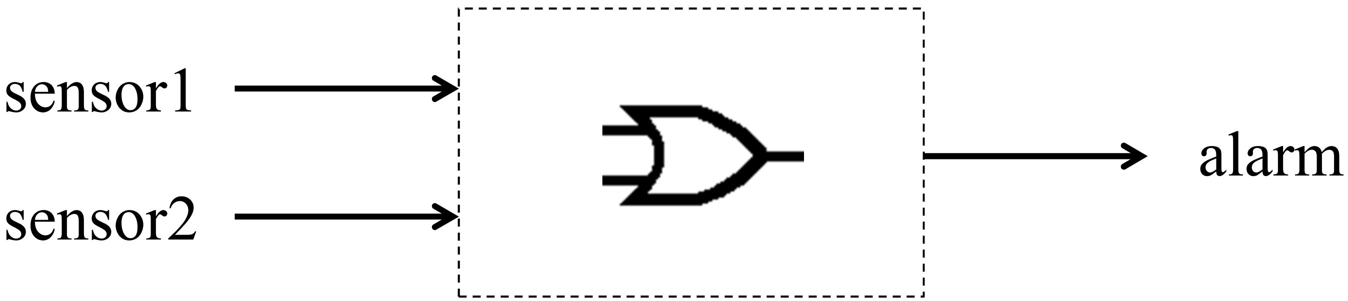
Gates can have more than two inputs e.g. three input AND

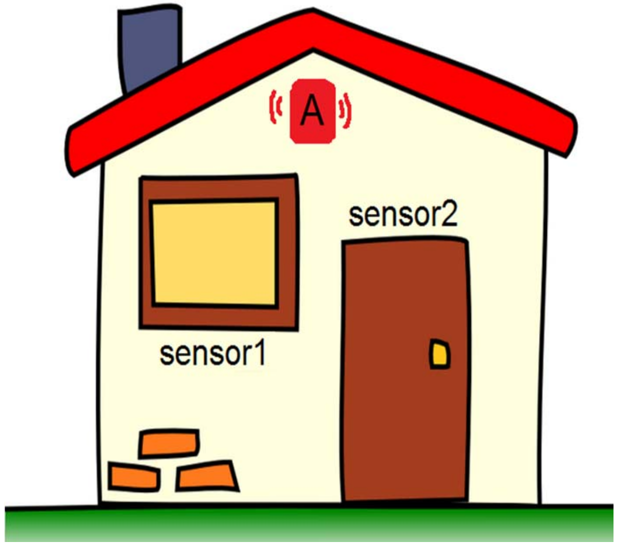
Burglar Alarm



When the window is open sensor1 outputs a **1**
When the door is open sensor2 outputs a **1**
The alarm will sound when its input is a **1**

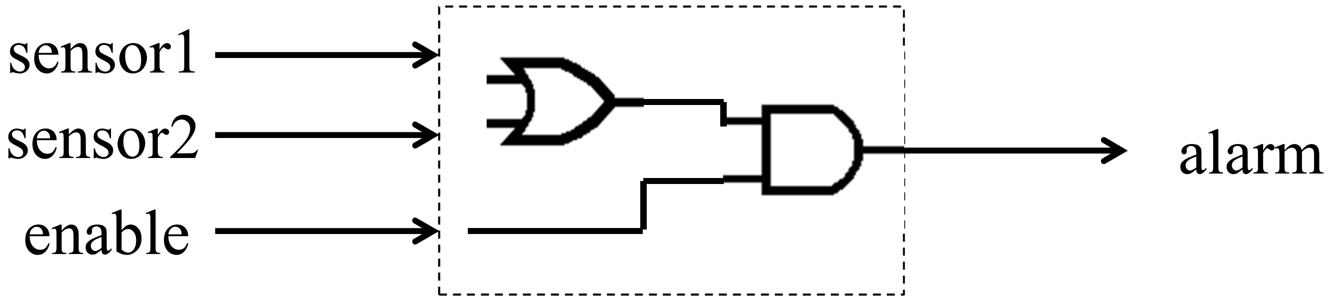
What is the required logic to sound the alarm if there is a break-in ?





When the window is open sensor1 outputs a **1**
When the door is open sensor2 outputs a **1**
The alarm will sound when its input is a **1**

The alarm should not sound if the owner opens the window or door so a control input '*enable*' is added. When *enable* is **1** the alarm should be activated. What is the required logic now ?



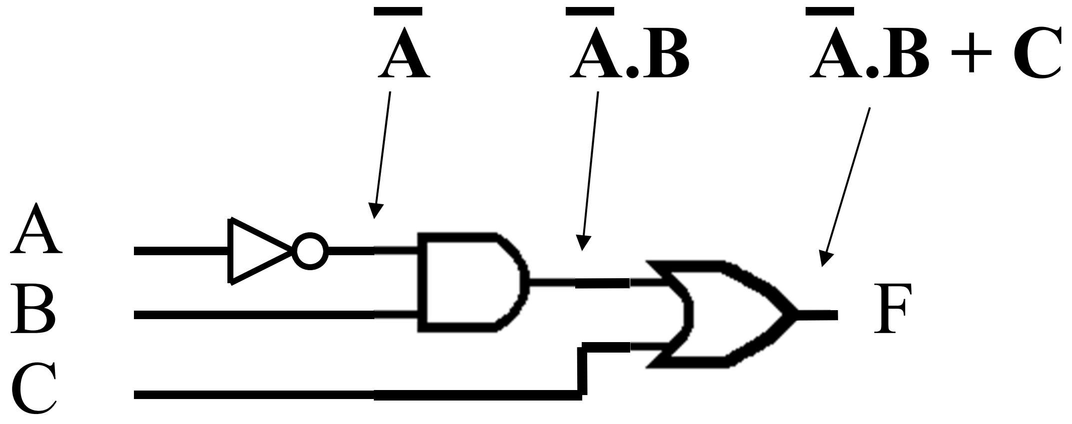
Logic Expressions

Logic expressions can be formed from operators and variables.
The order of precedence is NOT, AND, OR.

$$\mathbf{F = \overline{A}.B + C}$$

This means that we evaluate NOT terms first , then we AND the variables, then finally we perform the OR operation.

The function can be represented schematically:



Truth Tables

The function can also be represented as a truth table. The rows for the input values must follow the pattern shown.

$$F = \overline{A}.B + C$$

A B C	**$\overline{A}$**	**$\overline{A}$.B**	**$\overline{A}$.B + C**
0 0 0			
0 0 1			
0 1 0			
0 1 1			
1 0 0			
1 0 1			
1 1 0			
1 1 1			

Since a truth table covers all possible input combinations, it has 2^n rows, where n is the number of input variables.

A function with 4 variables W,X,Y,Z would have $2^4 = 16$ rows.

Truth Tables

The function can also be represented as a truth table. The rows for the input values must follow the pattern shown.

$F = \overline{A}.B + C$

A B C	$\overline{A}$	$\overline{A}.B$	$\overline{A}.B + C$
0 0 0	1		
0 0 1	1		
0 1 0	1		
0 1 1	1		
1 0 0	0		
1 0 1	0		
1 1 0	0		
1 1 1	0		

Since a truth table covers all possible input combinations, it has 2^n rows, where n is the number of input variables.

A function with 4 variables W,X,Y,Z would have $2^4 = 16$ rows.

Truth Tables

The function can also be represented as a truth table. The rows for the input values must follow the pattern shown.

$$F = \bar{A}.B + C$$

A B C	$\bar{A}$	$\bar{A}.B$	$\bar{A}.B + C$
0 0 0	1	0	
0 0 1	1	0	
0 1 0	1	1	
0 1 1	1	1	
1 0 0	0	0	
1 0 1	0	0	
1 1 0	0	0	
1 1 1	0	0	

Since a truth table covers all possible input combinations, it has 2^n rows, where n is the number of input variables.

A function with 4 variables W,X,Y,Z would have $2^4 = 16$ rows.

Truth Tables

The function can also be represented as a truth table. The rows for the input values must follow the pattern shown.

$$F = \overline{A}.B + C$$

A B C	$\overline{A}$	$\overline{A}.B$	$\overline{A}.B + C$
0 0 0	1	0	0
0 0 1	1	0	1
0 1 0	1	1	1
0 1 1	1	1	1
1 0 0	0	0	0
1 0 1	0	0	1
1 1 0	0	0	0
1 1 1	0	0	1

Since a truth table covers all possible input combinations, it has 2^n rows, where n is the number of input variables.

A function with 4 variables W,X,Y,Z would have $2^4 = 16$ rows.

Have a go …..

$$\mathbf{G = W(X + \overline{Y})}$$

Have a go .....

$$G = W(X + \overline{Y})$$

W	X	Y			
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Have a go …..

$$G = W(X + \overline{Y})$$

W	X	Y	$\overline{Y}$	$X + \overline{Y}$	$W(X + \overline{Y})$
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Have a go …..

$$G = W(X + \overline{Y})$$

W	X	Y	$\overline{Y}$	$X + \overline{Y}$	$W(X + \overline{Y})$
0	0	0	1		
0	0	1	0		
0	1	0	1		
0	1	1	0		
1	0	0	1		
1	0	1	0		
1	1	0	1		
1	1	1	0		

Have a go …..

$$\mathbf{G = W(X + \overline{Y})}$$

W	X	Y	$\overline{Y}$	$X + \overline{Y}$	$W(X + \overline{Y})$
0	0	0	1	1	
0	0	1	0	0	
0	1	0	1	1	
0	1	1	0	1	
1	0	0	1	1	
1	0	1	0	0	
1	1	0	1	1	
1	1	1	0	1	

Have a go …..

$$G = W(X + \overline{Y})$$

W	X	Y	$\overline{Y}$	$X + \overline{Y}$	$W(X + \overline{Y})$
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1

Example

For the following function produce a logic gate diagram and a truth table.

$$F = \overline{A}.B + \overline{B}.C$$

The order of precedence is NOT, AND, OR. This means that when we evaluate the function, we evaluate <u>NOT</u> first then <u>AND</u> then <u>OR</u>.

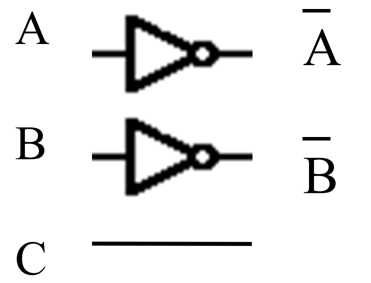
This can be illustrated by using parenthesis.

$$F = (\overline{A}.B) + (\overline{B}.C)$$

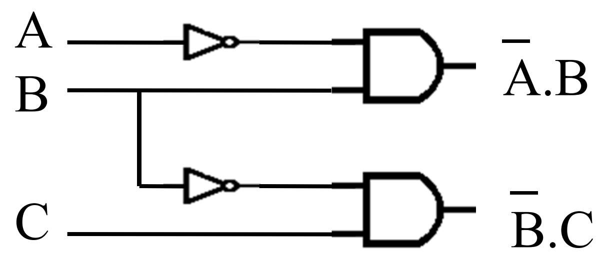
Evaluate the AND terms before 'OR'ing

To construct a logic diagram for $F = \overline{A}.B + \overline{B}.C$

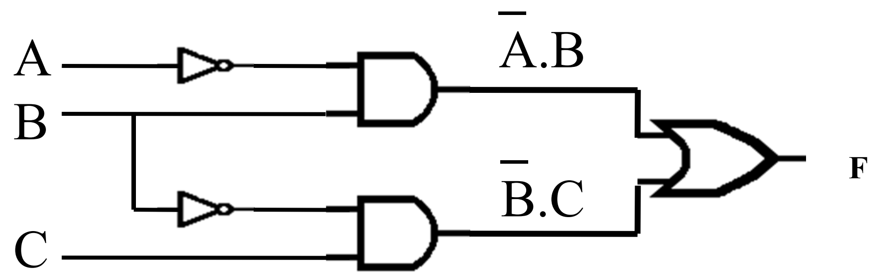
1. The inputs (generally taken from the left) are A,B,C Form the **NOT** of A and B as NOT has the highest order of precedence.



2. Form the **AND** terms $\overline{A}.B$ and $\overline{B}.C$



3. **OR** $\overline{A}.B$ with $\overline{B}.C$



To construct a truth table for $F = \overline{A}.B + \overline{B}.C$

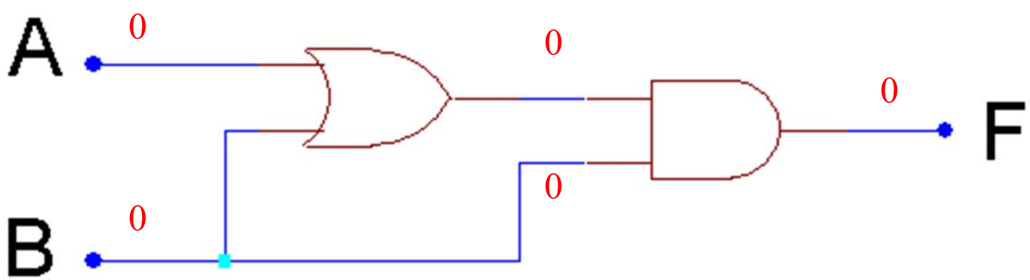
There are 3 variables A,B,C so the truth table will have $2^3 = 8$ rows covering all possible input combinations. (2 because this is a binary system)

ABC	$\overline{A}$	$\overline{B}$	$\overline{A}.B$	$\overline{B}.C$	F
0 0 0	1	1	0	0	0
0 0 1	1	1	0	1	1
0 1 0	1	0	1	0	1
0 1 1	1	0	1	0	1
1 0 0	0	1	0	0	0
1 0 1	0	1	0	1	1
1 1 0	0	0	0	0	0
1 1 1	0	0	0	0	0

Literal Analysis

A logic circuit is formed by a combination of logic gates. The output is a combination of the input values.

A logic schematic is commonly used to show the relationships between the inputs and outputs.



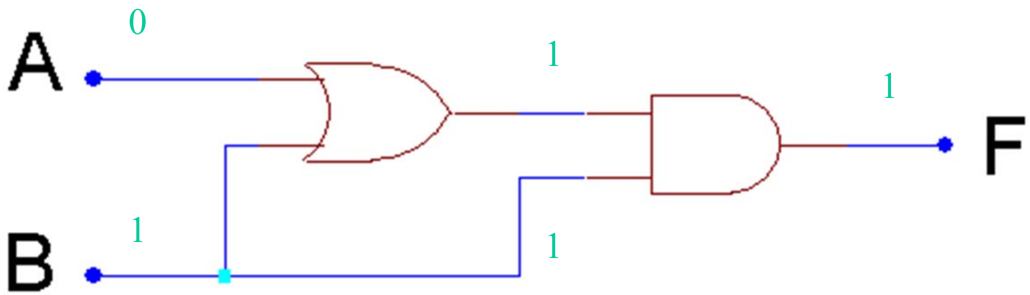
A	B	F
0	0	0
0	1	
1	0	
1	1	

For a simple circuit, the value of the output F, for inputs A and B, can be found by propagating the values of 1 and 0 from the input to the output. This is known as **Literal Analysis**.

Literal Analysis

A logic circuit is formed by a combination of logic gates. The output is a combination of the input values.

A logic schematic is commonly used to show the relationships between the inputs and outputs.



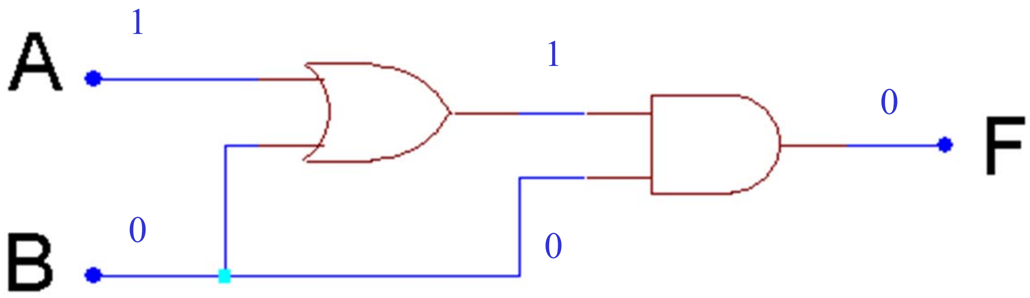
A	B	F
0	0	0
0	1	1
1	0	
1	1	

For a simple circuit, the value of the output F, for inputs A and B, can be found by propagating the values of 1 and 0 from the input to the output. This is known as **Literal Analysis**.

Literal Analysis

A logic circuit is formed by a combination of logic gates. The output is a combination of the input values.

A logic schematic is commonly used to show the relationships between the inputs and outputs.



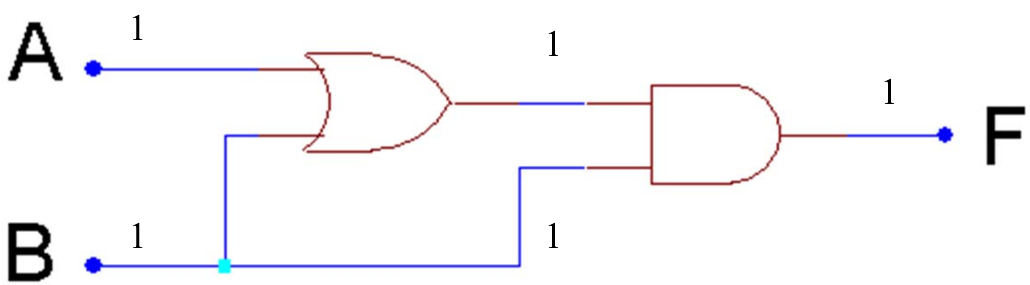
A	B	F
0	0	0
0	1	1
1	0	0
1	1	

For a simple circuit, the value of the output F, for inputs A and B, can be found by propagating the values of 1 and 0 from the input to the output. This is known as **Literal Analysis**.

Literal Analysis

A logic circuit is formed by a combination of logic gates. The output is a combination of the input values.

A logic schematic is commonly used to show the relationships between the inputs and outputs.

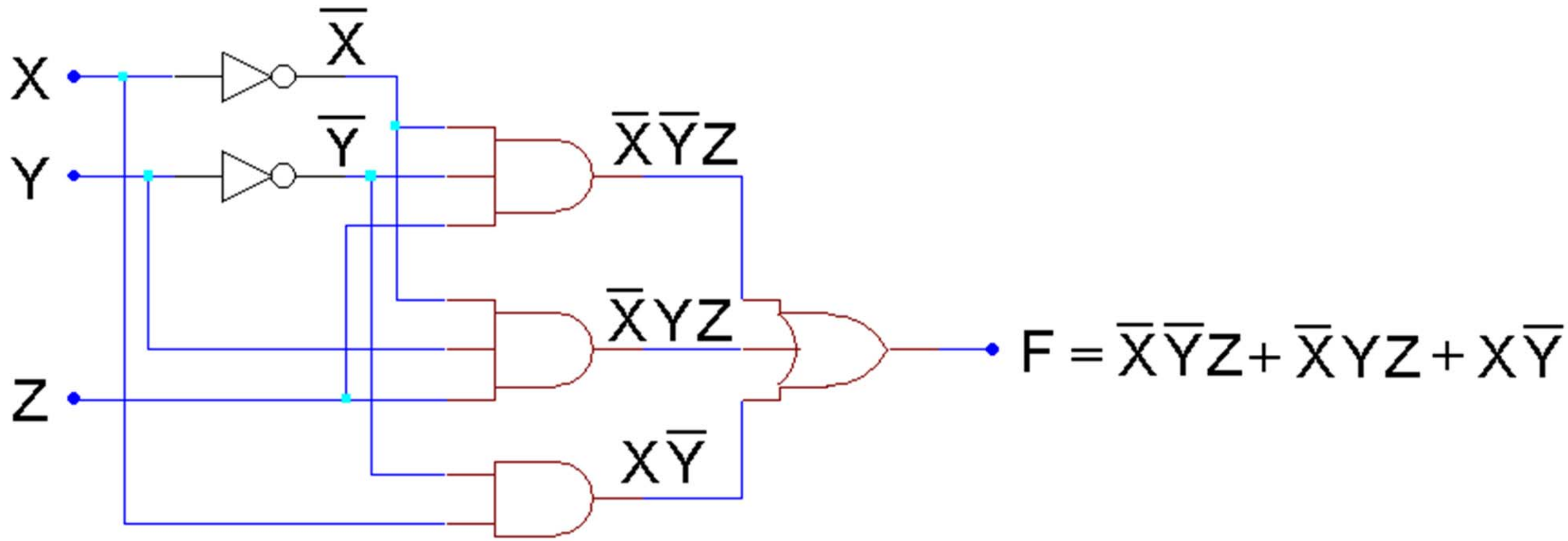


A	B	F
0	0	0
0	1	1
1	0	0
1	1	1

For a simple circuit, the value of the output F, for inputs A and B, can be found by propagating the values of 1 and 0 from the input to the output. This is known as **Literal Analysis**.

Symbolic Analysis

For larger circuits, literal analysis becomes impractical and it is necessary to obtain an algebraic expression that formally defines the relationship between the output and the input.



Instead of propagating literal values through the system, symbolic analysis propagates logic expressions through the system. The resulting expression defines the behaviour of the system for all input combinations.

Deriving Expressions from Truth Tables

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

The table tells us that f is '1' if :

a = '0' AND b = '1' OR

a = '1' AND b = '0' OR

a = '1' AND b = '1'

This is the same as saying :

$\overline{a}$ = '1' AND b = '1' OR

a = '1' AND $\overline{b}$ = '1' OR

a = '1' AND b = '1'

Which gives the expression :

$f = \overline{a}.b + a.\overline{b} + a.b$

1. Look down the output column of the truth table for the first '1'.
2. Look at the input values responsible for the '1'.
3. AND together those variables, remembering to complement '0' terms.
4. Repeat the ANDing process for each row of the truth table with a '1' output.
5. OR together all of the terms generated.

Other common logic gates

X	Y	$\overline{X.Y}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND

X	Y	$\overline{X + Y}$
0	0	1
0	1	0
1	0	0
1	1	0

NOR

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

XOR

Exclusive OR

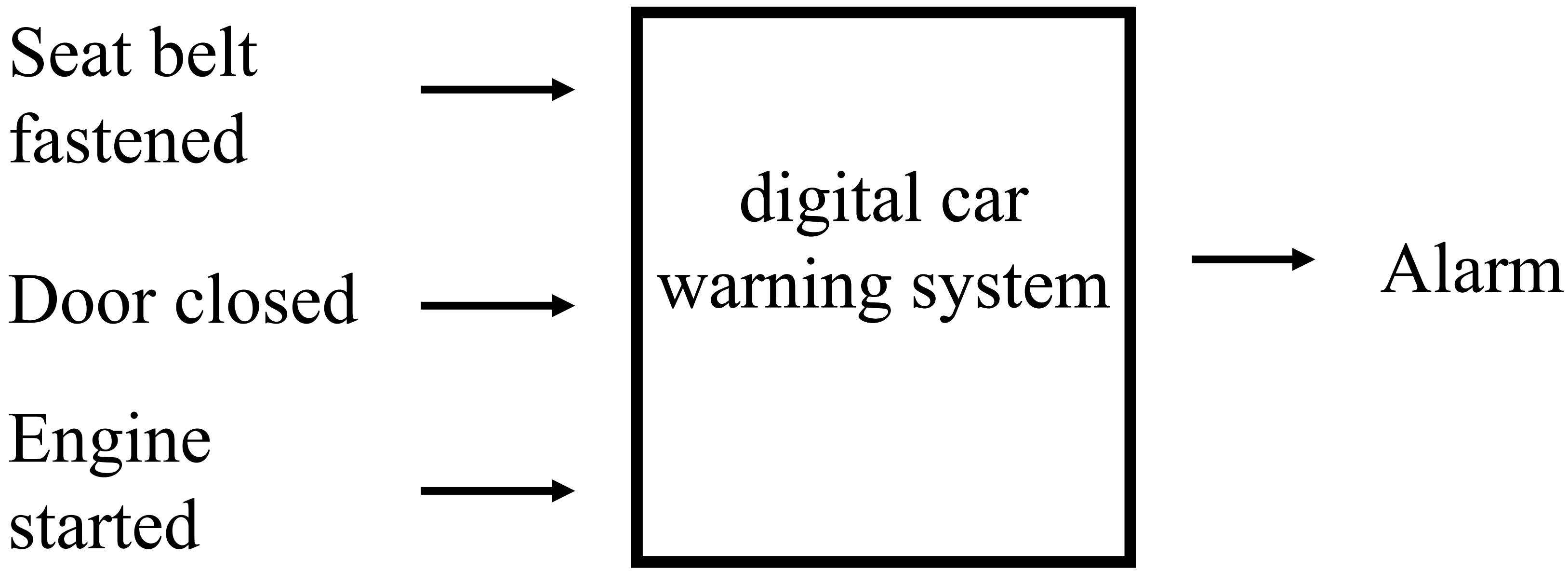
X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

XNOR

Exclusive NOR

Design Problem

"A car alarm is required that must sound if the engine is started and the seat belt is not fastened or the door is open."



Car Alarm Solution

1. Assign logic variables:

Seat Belt (S) , fastened S = 1

Driver Door (D), closed D = 1

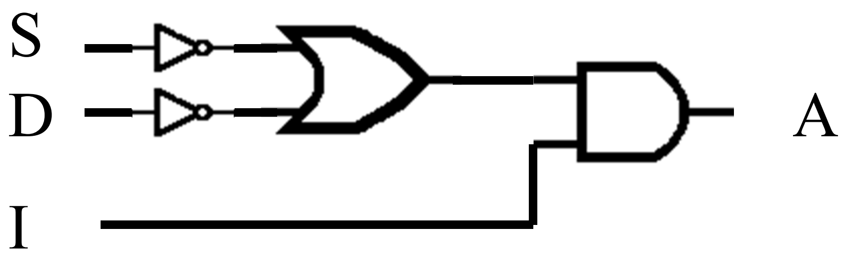
Ignition On (I), on I= 1

Alarm (A) , Alarm On = 1

2. Deduce logic equation:

$$A = I.(\overline{S} + \overline{D})$$

3. Draw logic circuit:

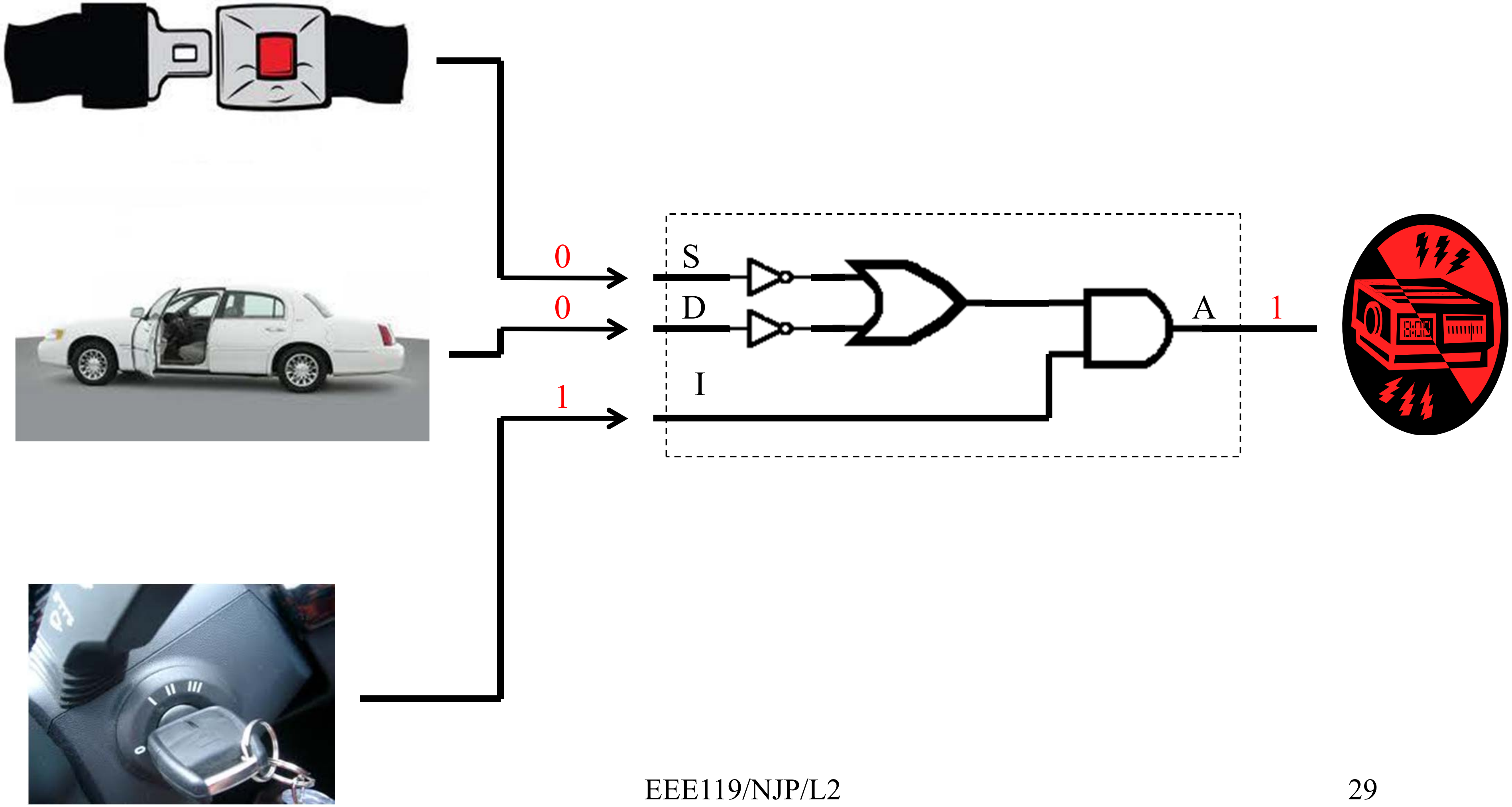


Seat Belt (S) , fastened S = 1

Driver Door (D), closed D = 1

Alarm (A) , Alarm On = 1

Ignition On (I), on I= 1



Summary

- Logic circuits can be analysed literally or algebraically
- A truth table can be used to display the output of a function for all possible combinations of the inputs
- Logic expressions and circuits can be derived from a statement of the problem