Unit-2

Boolean Algebra and Logic Gates

Unit-2

Boolean Algebra and Logic Gates

Boolean Algebra

Boolean Algebra Laws AND laws

1. ⬚⬚ · 0 = 0 ⬚⬚⬚⬚⬚⬚⬚⬚

2. $\square\square \cdot 1 = \square\square$


 3. $\square\square \cdot \square\square = \square\square$

4. $\square\square \cdot \square\square^{\text{n}} = 0$

- Commutative laws

1. $\square\square + \square\square = \square\square + \square\square$
2. $\square\square \cdot \square\square = \square\square \cdot \square\square$

- OR laws

1. $\square\square + 0 = \square\square$

2. $\square\square + 1 = 1$


 3. $\square\square + \square\square = \square\square$

4. $\square\square + \square\square^{\text{n}} = 1$

- Associative laws

1. ◻◻ + ◻◻ + ◻◻ = ◻◻ + (◻◻ + ◻◻) 2. ◻◻ · ◻◻ ◻◻ = ◻◻(◻◻ · ◻◻)

Boolean Algebra Laws

- Distributive laws

1. ◻◻ ◻◻ + ◻◻ = ◻◻◻◻ + ◻◻◻◻
2. ◻◻ + ◻◻◻◻ = (◻◻ + ◻◻)(◻◻ + ◻◻)

- Idempotent laws

1. ◻◻ · ◻◻ = ◻◻
2. ◻◻ + ◻◻ = ◻◻

- De Morgan's Theorem

1. ⬚⬚ + ⬚⬚ = ⬚⬚ ⬚⬚

2. ⬚⬚⬚⬚ = ⬚⬚ + ⬚⬚

Break the line change the sign

- Redundant Literal Rule 1. ⬚⬚ + ⬚⬚ ⬚⬚ =

⬚⬚ + ⬚⬚ 2. ⬚⬚ ⬚⬚ + ⬚⬚ = ⬚⬚⬚⬚

- Absorption laws

1. ⬚⬚ + ⬚⬚⬚⬚ = ⬚⬚

2. ⬚⬚(⬚⬚ + ⬚⬚) = ⬚⬚ $\overline{\overline{A}} = A$

Proof of ⬚ + ⬚ + ⬚ = ⬚ ⬚ ⬚ L.H.S. R.H.S.

A	B	C	A+B+C	A+B+ C	A	⬚	⬚	A B C
0	0	0	0	1	1	1	1	1
0	0	1	1	0	1	1	0	0
0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	0	0

1	0	0	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	0	1	0
1	1	1	1	0	0	0	0	0

From truth table, it is clearly visible that L.H.S. = R.H.S. Hence, **the complement of a sum of variables is equal to the product of their individual complements.**

Proof of $\overline{A+B+C} = \overline{A}\,\overline{B}\,\overline{C}$ L.H.S. R.H.S.

A	B	C	A B C	$\overline{A\,B\,C}$	$\overline{A}$	$\overline{B}\,\overline{C}$	$\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}\,\overline{C}$

							$\overline{(ABC)}$	$+ C^{\overline{(ABC)}}$
0	0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0	1
0	1	0	0	1	1	0	1	1
0	1	1	0	1	1	0	0	1
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	0	1	1
1	1	1	1	0	0	0	0	0

From truth table, it is clearly visible that L.H.S. = R.H.S. Hence, **the complement of a product of variables is equal to the sum of their individual complements.**

Reducing Boolean Expression (Example – 1)

 Reduce the expression

$$F = F + F[FFF + F + F^{\prime}F]$$

$$F = F + F[FFF + F + F^{\prime}F]$$

$$F = F + F[FFF + FFF + F^{\prime}F]$$

$$F = F + FFFFFF + FFFFFF + FFFF^{\prime}F$$

$$F = F + FFFFFF + FFFF + FFFF^{\prime}F$$

$$F = F(1 + FFFF) + FFFF(1 + F^{\prime})$$

$$F = F + FFFF$$

(1 + A = 1)

(Distributive law) (Distributive law) (A.A = A)

Reducing Boolean Expression

(Example – 2) Reduce the expression

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}(\text{□□□□} + \text{□□□□}^{\text{□□}})]$$

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}(\text{□□□□} + \text{□□□□}^{\text{□□}})]$$

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}(\text{□□□□□□□□□}^{\text{□□}})]$$

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}(\text{□□}^{\text{□□}} + \text{□□}^{\text{□□}})(\text{□□}^{\text{□□}} + \text{□□})]$$

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}(\text{□□}^{\text{□□}}\text{□□}^{\text{□□}} + \text{□□}^{\text{□□}}\text{□□} + \text{□□}^{\text{□□}}\text{□□})]$$

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}\text{□□}^{\text{□□}} + \text{□□}^{\text{□□}}\text{□□}^{\text{□□}}\text{□□} + \text{□□}^{\text{□□}}\text{□□}^{\text{□□}}\text{□□} + \text{□□}^{\text{□□}}\text{□□}^{\text{□□}}\text{□□}]$$

$$\text{□□} = \text{□□}[\text{□□} + \text{□□}^{\text{□□}}\text{□□}^{\text{□□}} + 0 + \text{□□}^{\text{□□}}\text{□□}^{\text{□□}}\text{□□}^{\text{□□}} + 0]$$

$$\text{□□} = \text{□□□□} + \text{□□□□}^{\text{□□}}\text{□□}^{\text{□□}} +$$

$$\text{□□□□}^{\text{□}} \quad \text{□□}^{\text{□}} \quad \text{□□}^{\text{□}}$$

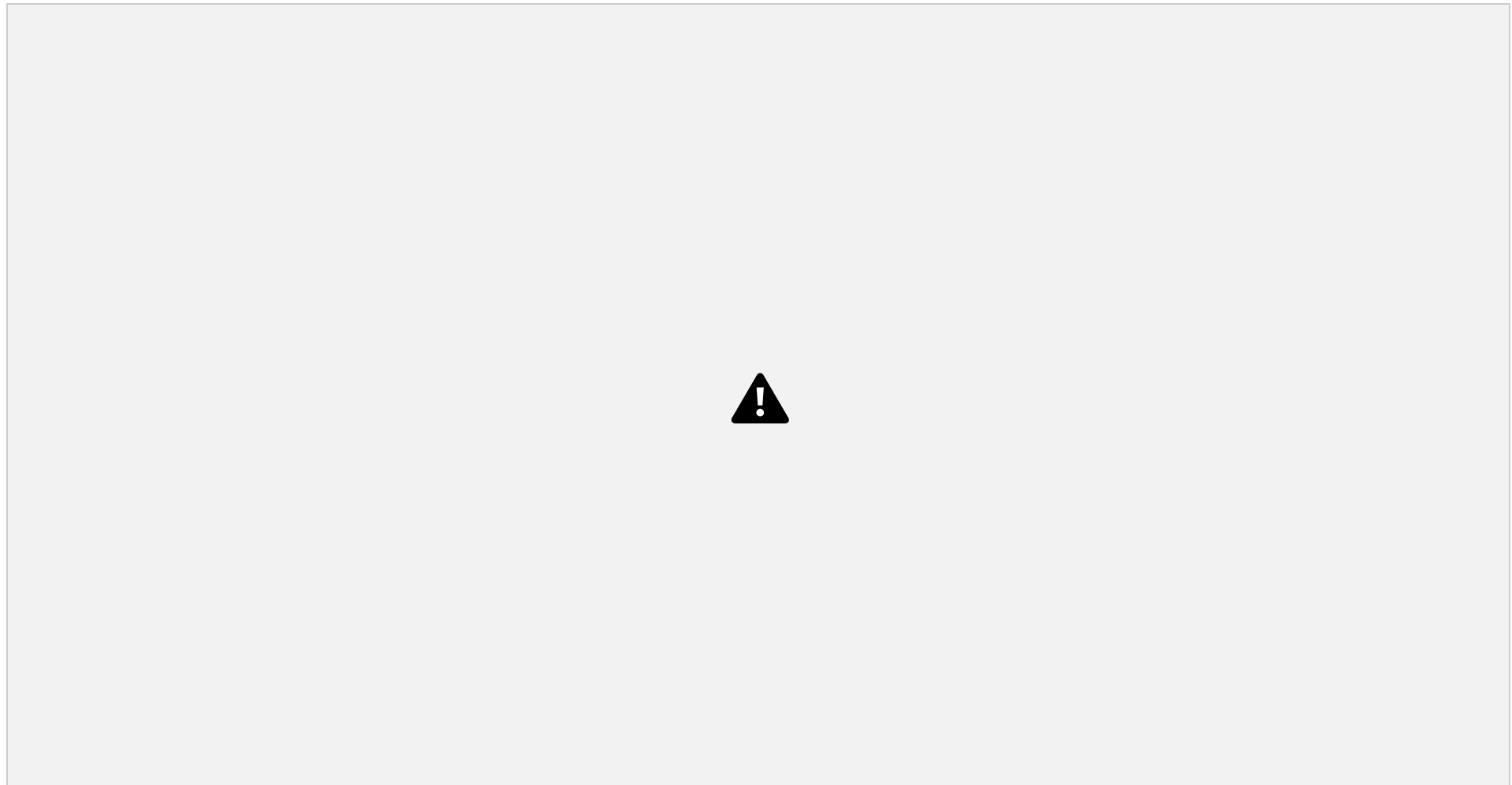
$$\text{□□} = \text{□□□□} + 0 + 0$$

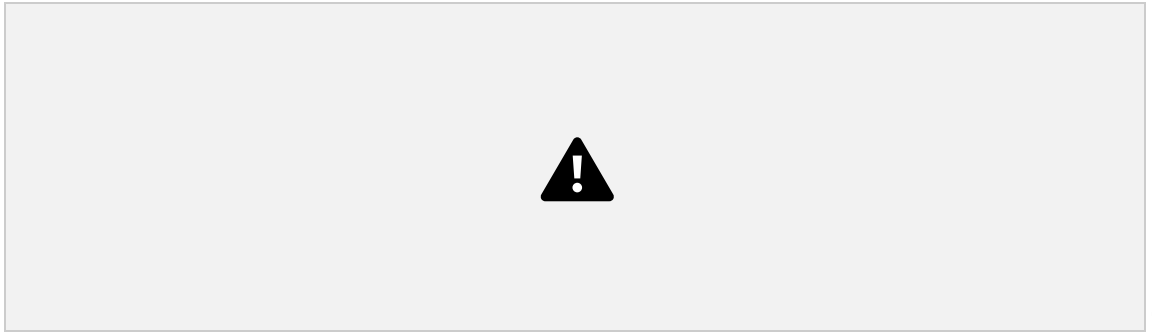
$$\text{□□} = \text{□□□□}$$

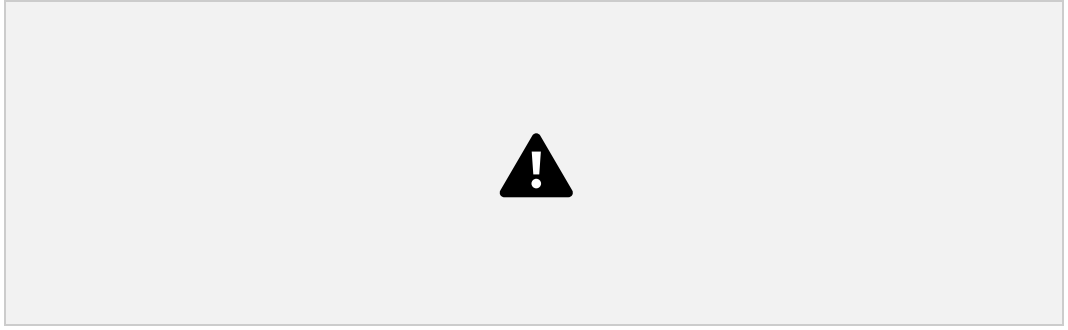
law) (Distributive law) (A.A’ = 0)

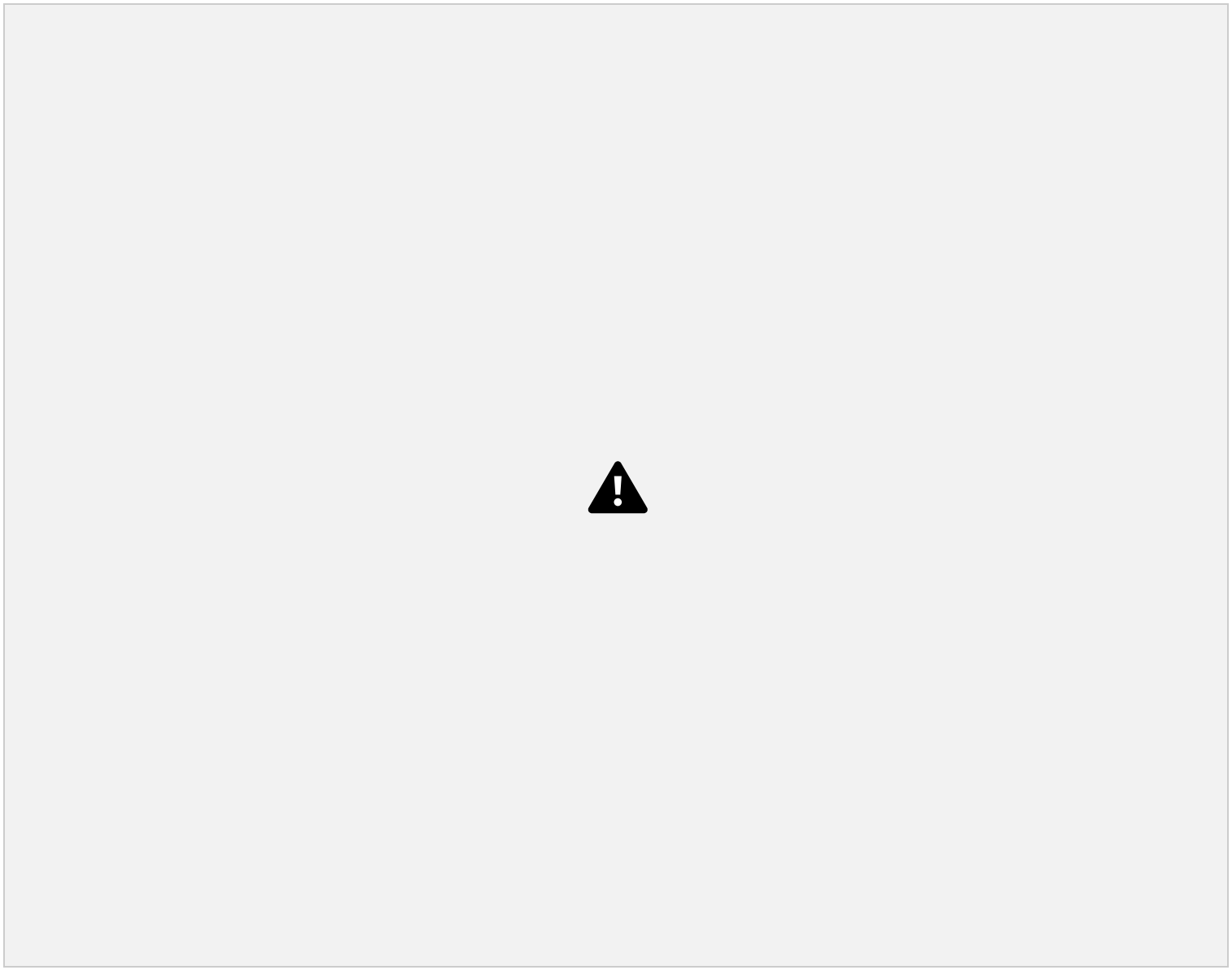
(Distributive law) (A.A’ = 0)

(De-Morgan’s law) (De-Morgan’s law) (Distributive









Logic Gates

Logic Gates

- Most basic logical unit of the digital system is gate

circuit. Types of gate circuits are as follows

1. AND Gate
2. OR Gate
3. NOT Gate (Inverter)
4. NOR Gate
5. NAND Gate
6. XOR Gate
7. XNOR Gate

⚠

1. AND Gate AND Gate has an output which is normally at logic level “0” and only goes “HIGH” to a logic level “1” when ALL of its inputs are at logic level “1”

Logic Notation

2-input AND Gate Truth Table

A

B C

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

$$C = A \cdot B$$

2. OR Gate

- OR Gate or Inclusive-OR gate has an output which is normally at logic level “0” and only goes “HIGH” to a logic level “1” when one or more of its inputs are at logic level “1”.

Logic Notation

2-input OR Gate Truth Table

A

B C

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

$$C = A + B$$

Inverter Gate Truth Table

3. NOT (Inverter) Gate

NOT gate has an output which is always opposite to input level.

Logic Notation

A C

A	C

0	1
1	0

$$\text{NOT}(A) = A'$$

4. NOR Gate

- NOR Gate is an OR gate followed by an inverter.
- NOR Gate has an output which is normally at logic level “1” and only goes “LOW” to a logic level

“0” when one or more of its inputs are at logic level “1”.

Logic Notation

2-input NOR Gate Truth Table

A

B C

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

$$C = A + B'$$

5. NAND Gate

- NAND Gate is an AND gate followed by an inverter.
- NAND Gate has an output which is normally at logic level “1” and only goes “LOW” to a logic level “0” when ALL inputs are at logic level “1”.

Logic Notation

2-input NAND Gate Truth Table

A

B C

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

$$C = (A \cdot B)'$$

6. Exclusive-OR (X-OR) Gate

- X-OR gate that has 1 state when one and only one of its two inputs assumes a logic 1 state and has 0 state when all of its input are same.

- Also known as **anti-coincidence gate** or **inequality detector**.

2-input XOR Gate Truth Table

A

B C

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

Logic Notation

$$A \oplus B = C$$

7. Exclusive-NOR (X-NOR) Gate

- X-NOR gate that has 1 state when all of its input are same and has 0 state when one of its input has 0 state and other input is 1 state.
- Also known as **coincidence gate** or **equality detector**.

Logic Notation

2-input XNOR Gate Truth Table

A

B C

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

$$C = A \odot B$$

NAND as Universal Gate

(AB)’ ((AB)’)’ = AB AND using NAND

A A’ A B NOT using NAND

A’

A

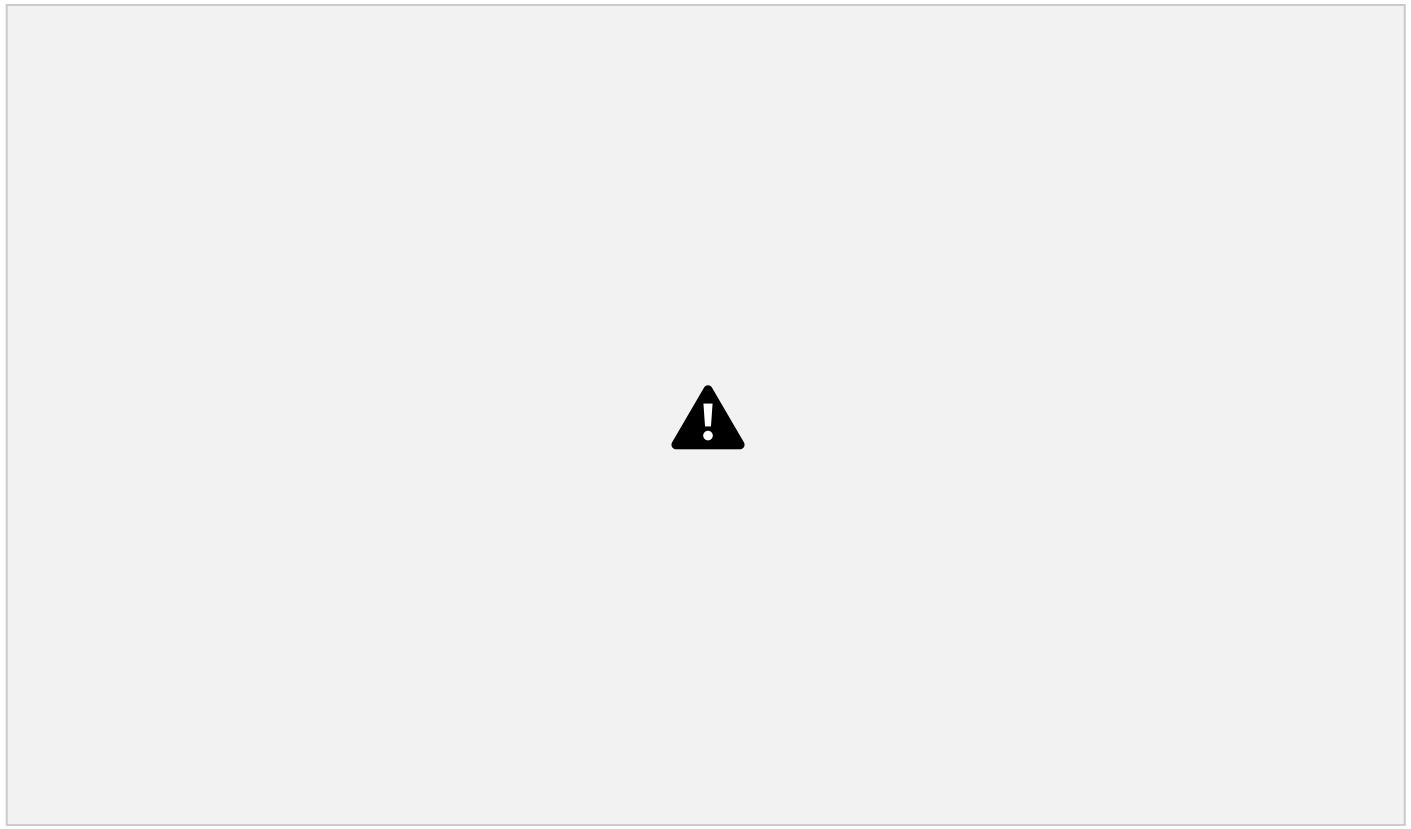
(A’B’)’ = (A+B)

B

B’

OR using NAND

XOR using NAND



NOR as Universal Gate A A’ $^{A}_{B}$ (A+B)’ ((A+B)’)’ = A+B

NOT using NOR
OR using NOR

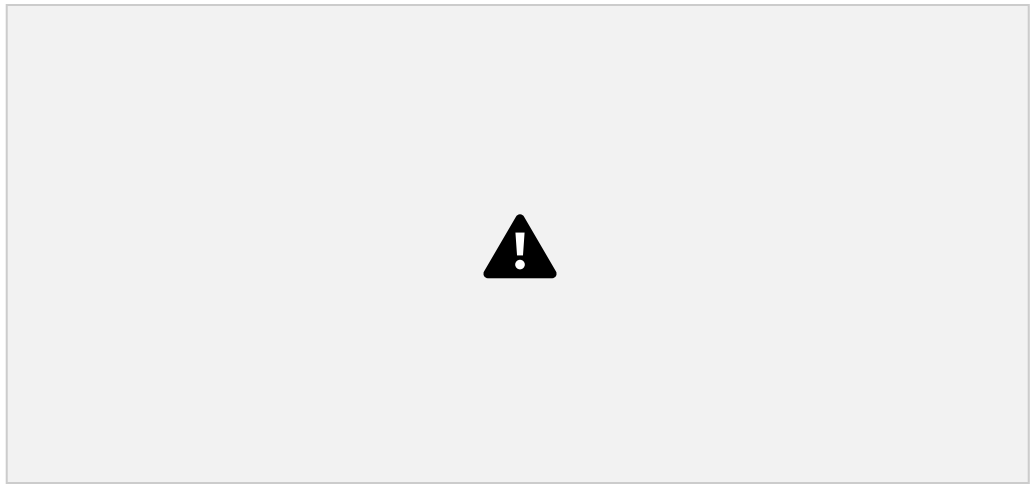
A’
A

(A’+B’)’ = AB

B
B’

AND using NOR

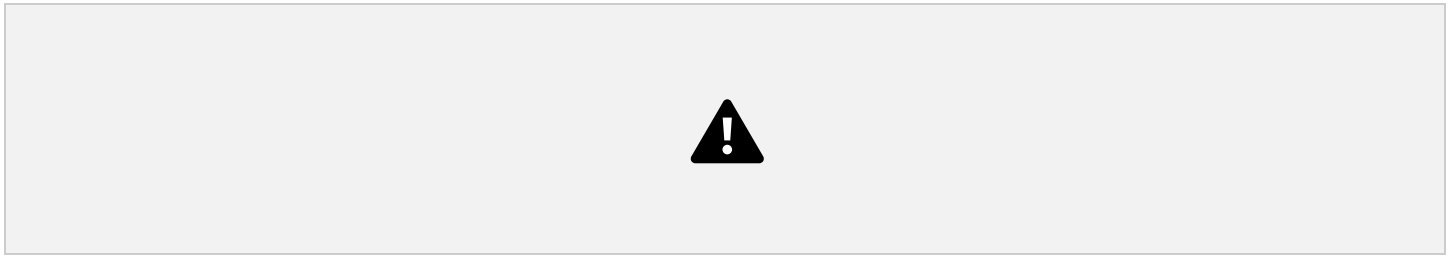
XOR using NOR



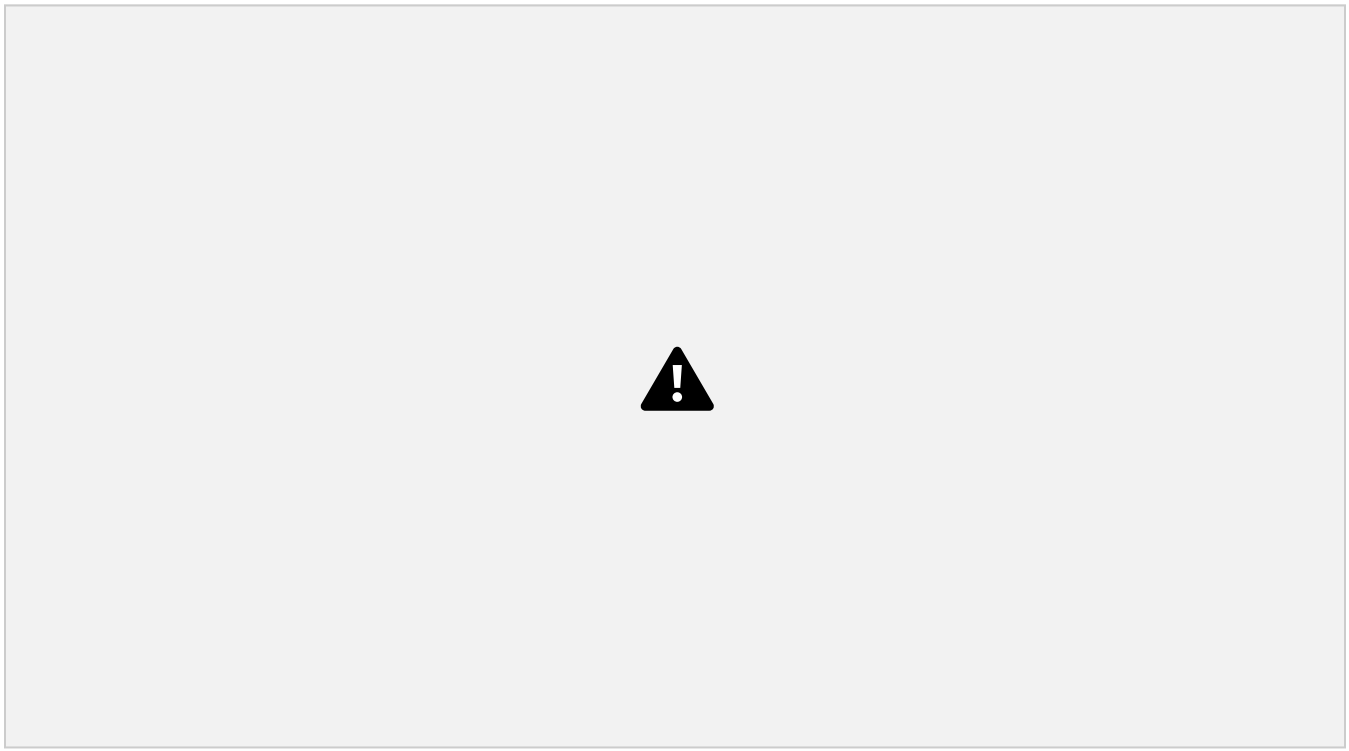
Principle of Duality

- The principle of duality says that given an expression which is always valid in boolean algebra, the dual expression is also always valid.
- To form the dual of an algebraic expression you simply need to:
- 1) Interchange AND and OR operators

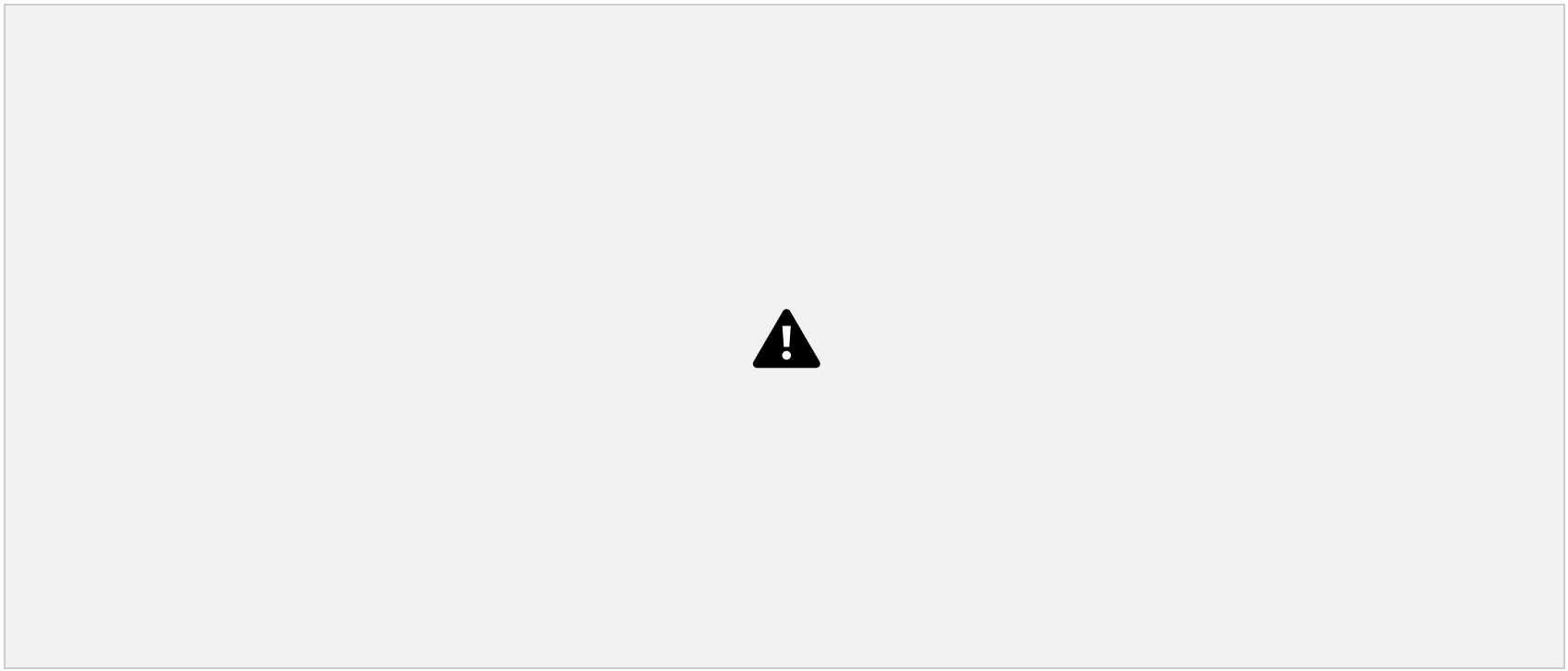
- 2) Replace 1’s with 0’s and 0’s with 1’s



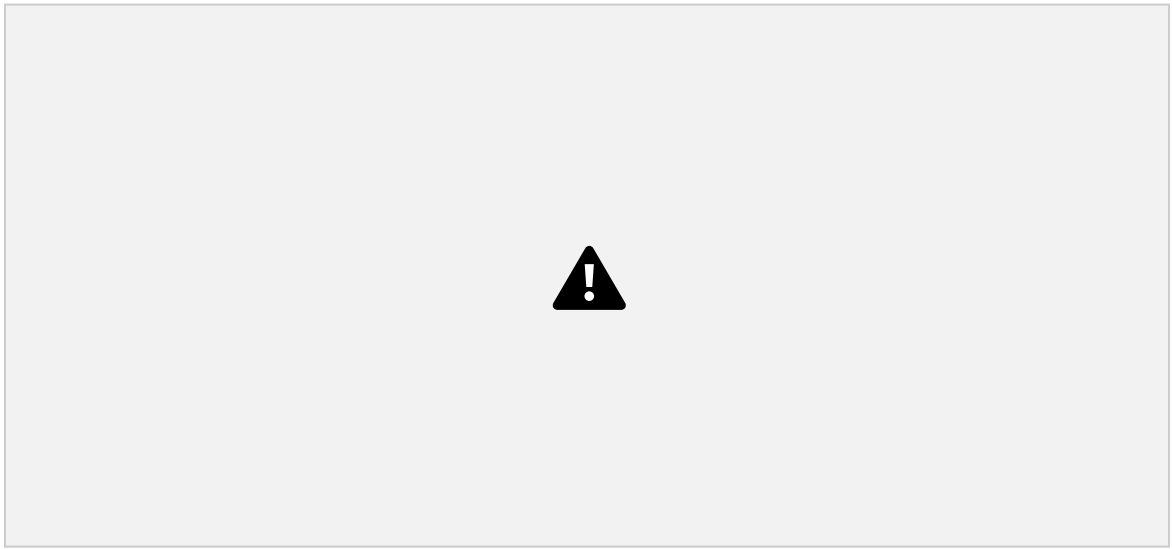
Example

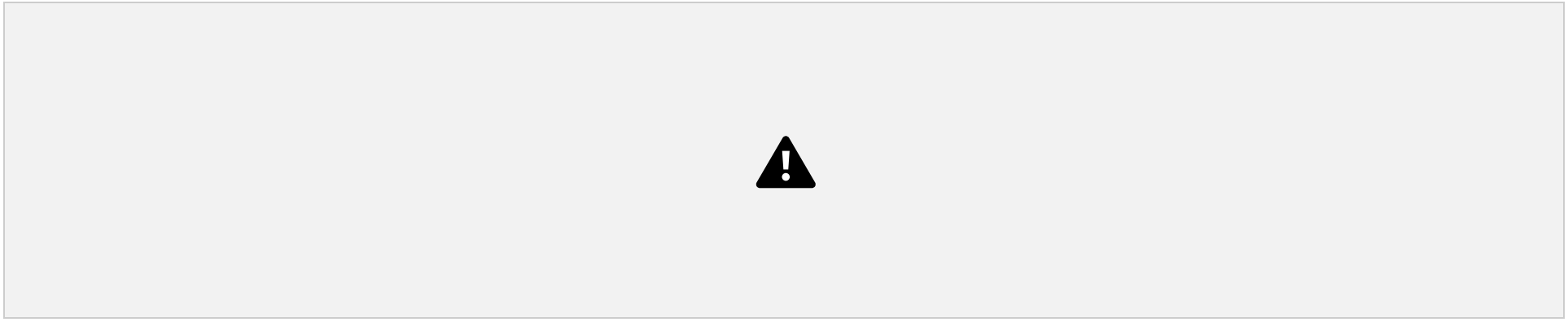


27



Example





- Logical functions are generally expressed in terms of different combinations of logical variables with their true forms as well as the complement forms.
- An arbitrary logic function can be expressed in the following forms.

(i) **Sum of the Products (SOP)**

(ii) **Product of the Sums (POS)**

2

	Logical product (AND function) of several variables	Logical sum (OR function) of several variables
	AND	OR

	True or complemented	True or complemented
	ABC′	A + B + C′
	Standard product	Standard sum
	Variables multiplied (concatenated)	Variables added (plus signs)

Logical sum (OR) of two or more logical product (AND) terms

Logical product (AND) of two or more logical sum (OR) terms

OR on AND operated variables AND on OR operated variables

Product terms combined using OR Sum terms combined using AND

$Y = AB + BC + AC$ or

$= A'B + BC + AC'$

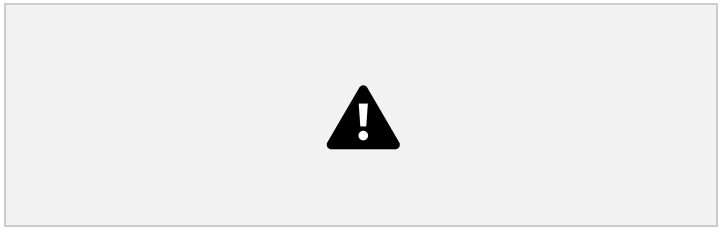
Y

$Y = (A + B + C)(A + B' + C)(A + B + C')$ or

$Y = (A + B + C)(A' + B' + C')$

Sum of product terms Product of sum terms

Standard form for digital circuits Alternate form for digital circuits



- Before discussing about canonical or normal form of SOP and POS , we must be familiar with

- **Minterms** & **Maxterms** 

AND operation of all variables in true or complemented form

OR operation of all variables in true or

complemented form

AND (Product) OR (Sum)

Each variable appears exactly once in each product term

Each variable appears exactly once in each sum term

x'y'z, xy'z, xyz', xyz x + y + z, x' + y + z, x + y' + z, x + y + z'

2^n for n variables 2^n for n variables

True form (1), complemented form (0)

True form (0), complemented form (1)

Concatenation of variables

with AND operation

Addition of variables with OR operation

f(x, y, z) = x'y'z + xy'z + xyz' +

All variables must appear in each product term for it to be a minterm

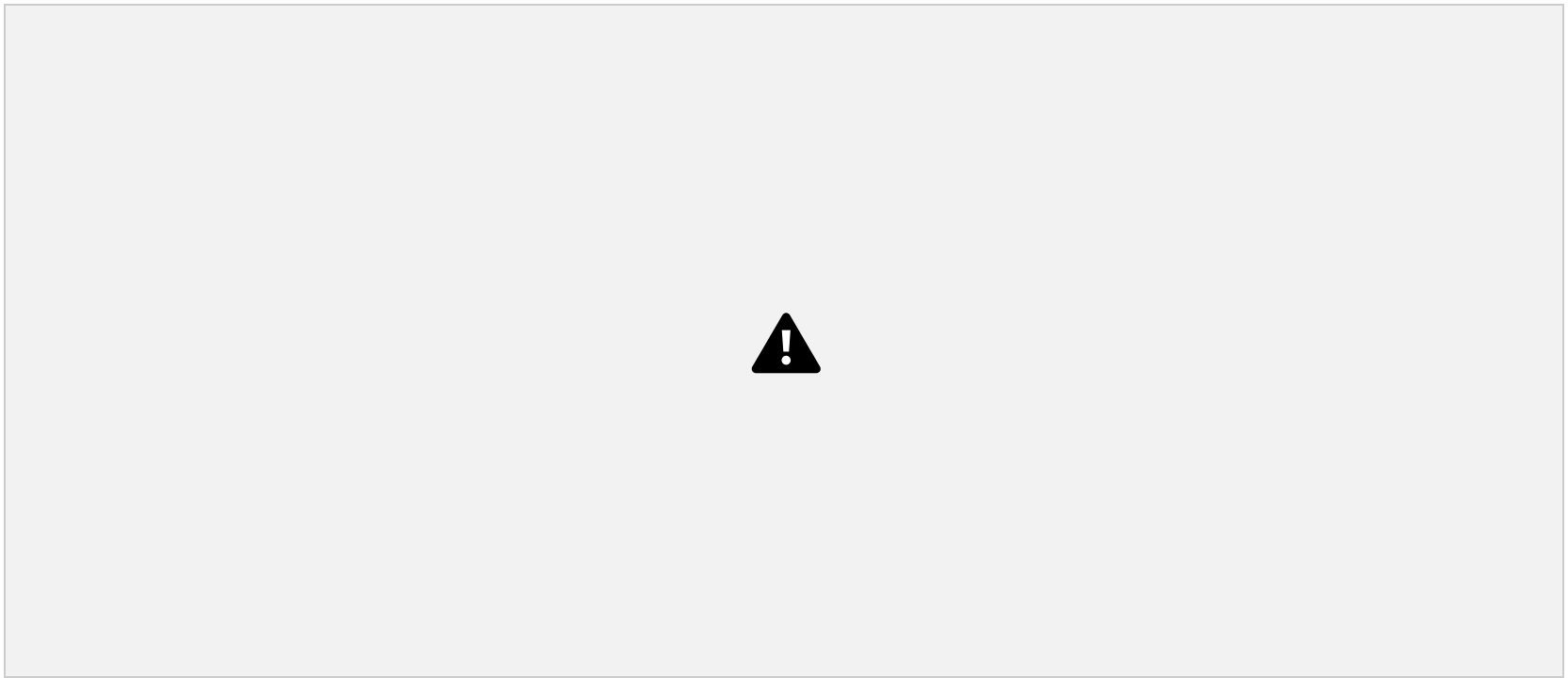
All variables must appear in each sum term for it to be a maxterm

$_{xyz}f(x, y, z) = (x + y + z)(x' + y + z)(x + y' + z)(x + y + z')$

$f(x, y) = x + xy$ Not a minterm

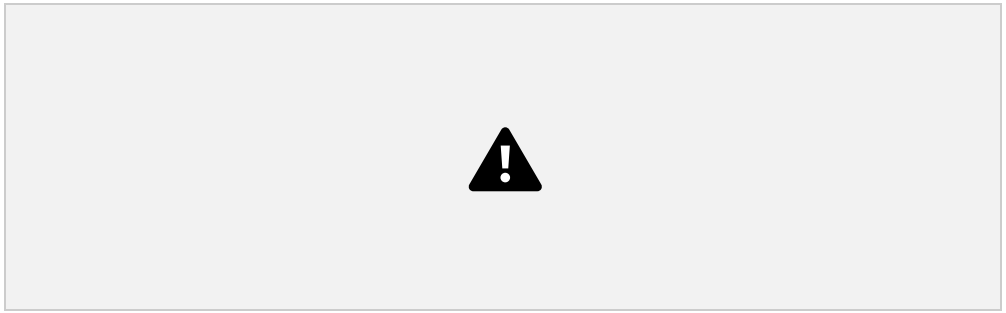
$f(x, y, z, w) = (x + y')(x' + y' + z + w')(x + y + z')$ Not all maxterms

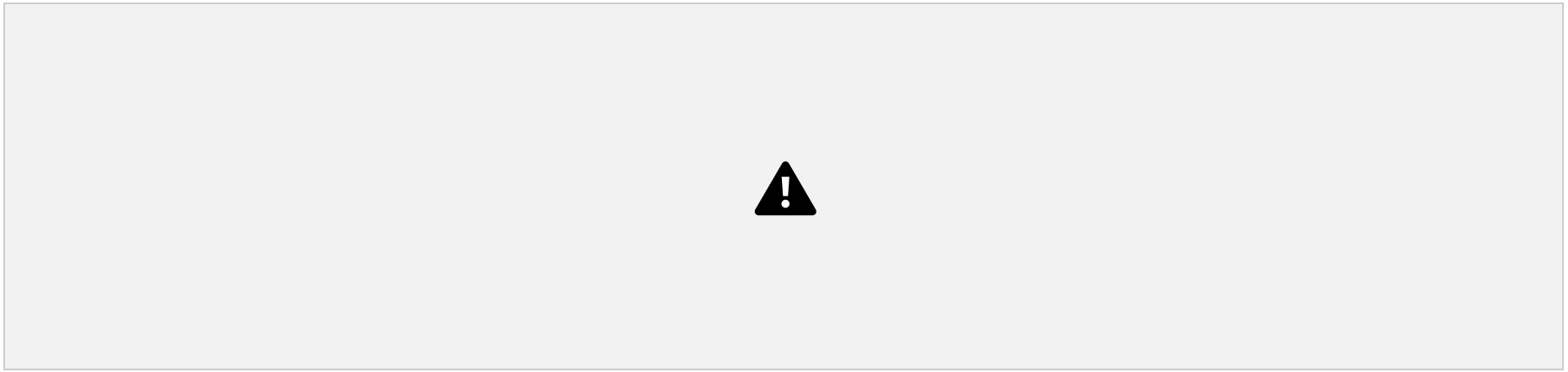
7



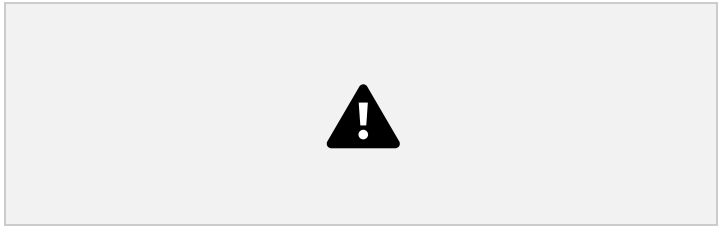
- When a Boolean function is expressed as the logical sum of all the minterms from the rows of a truth table, for which the value of the function is 1, it is referred to as the canonical sum of product expression.
- For example, if the canonical sum of product form of a three-variable logic function F has the minterms A′BC, AB′C, and ABC′, this can be expressed as the sum of the decimal codes corresponding to these minterms as below

9

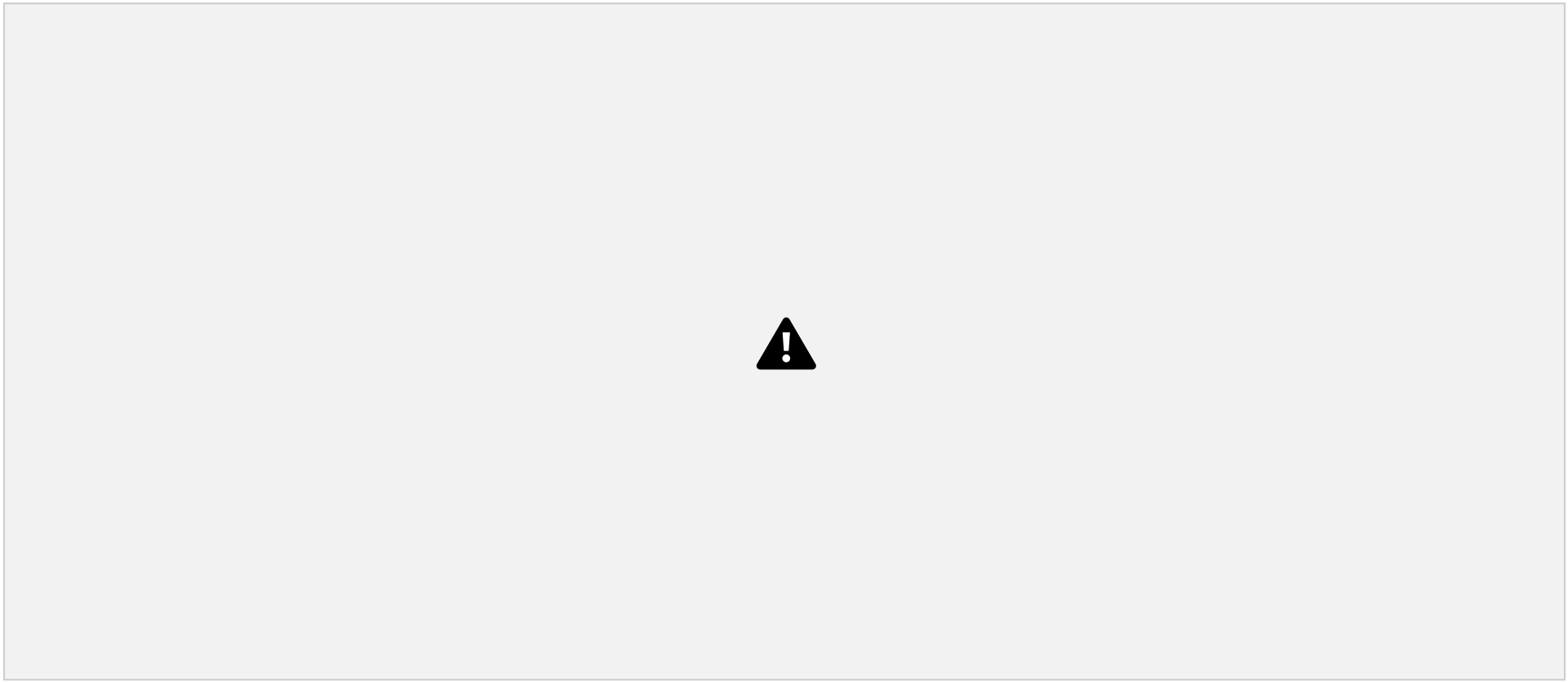




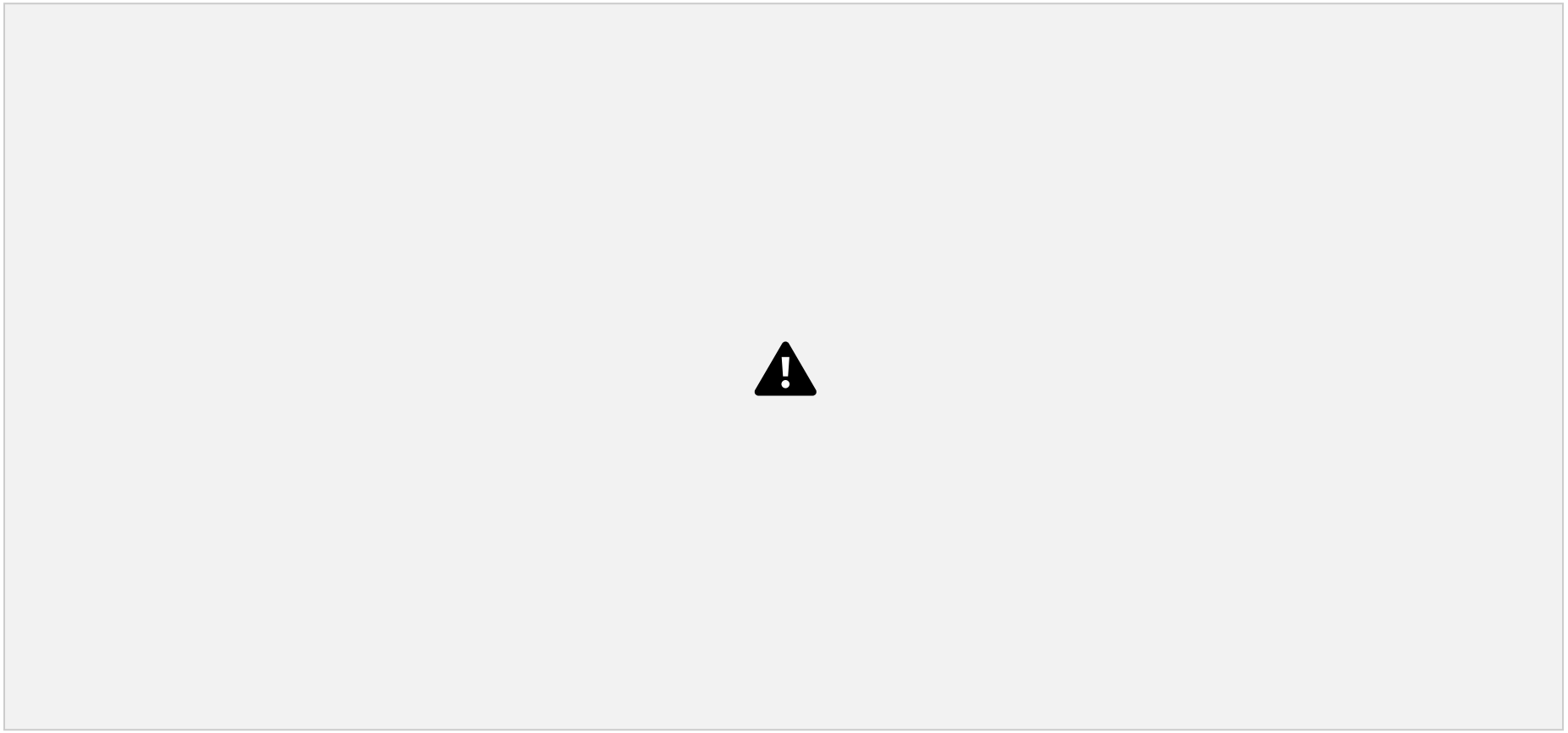
10



11

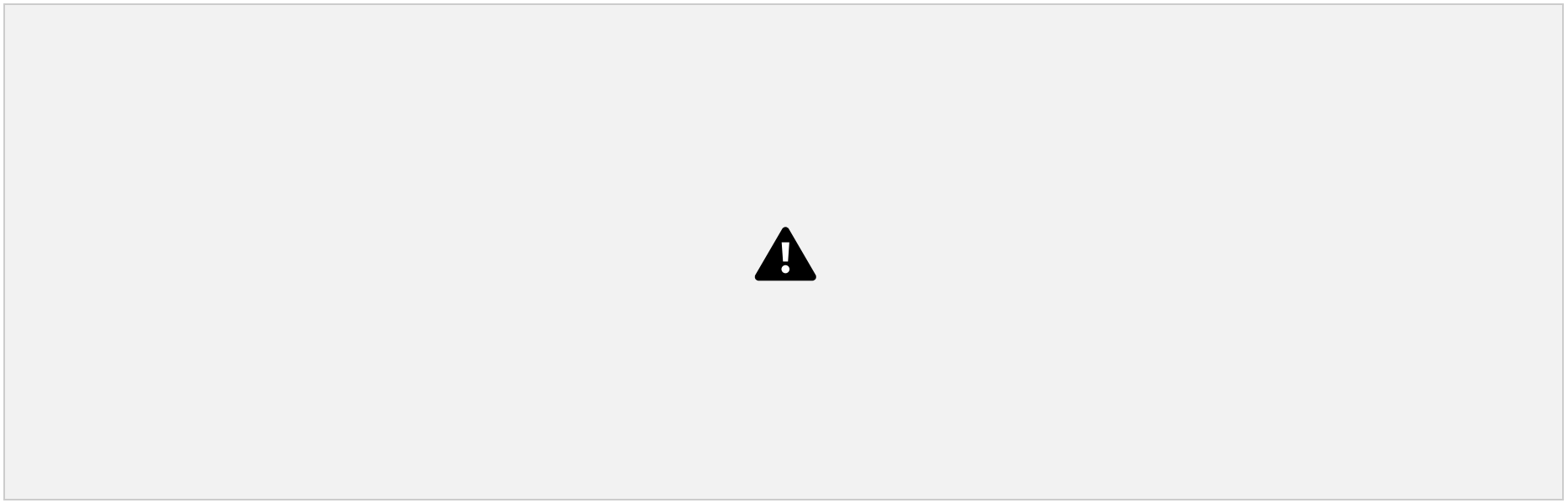


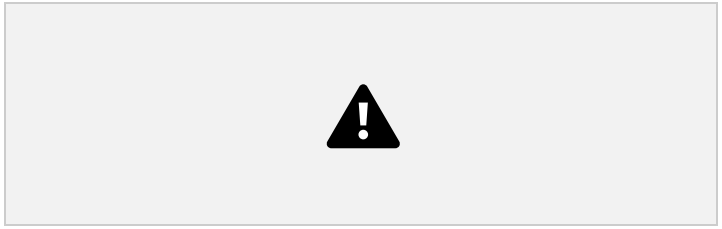
12



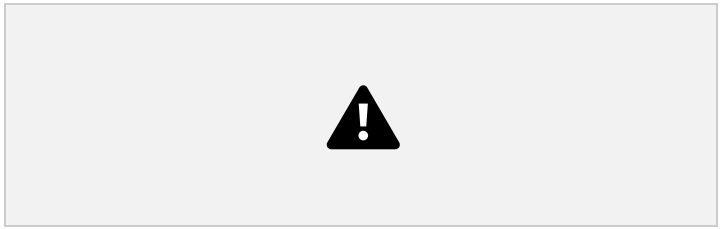
When a Boolean function is expressed as the logical product of all the maxterms from the rows of a truth table, for which the value of the function is 0, it is referred to as the canonical product of sum expression.

- For example, if the canonical product of sums form of a three variable logic function F has the maxterms A + B + C, A + B′ + C, and A′ + B + C′, this can be expressed as the product of the decimal codes corresponding to these maxterms as below,

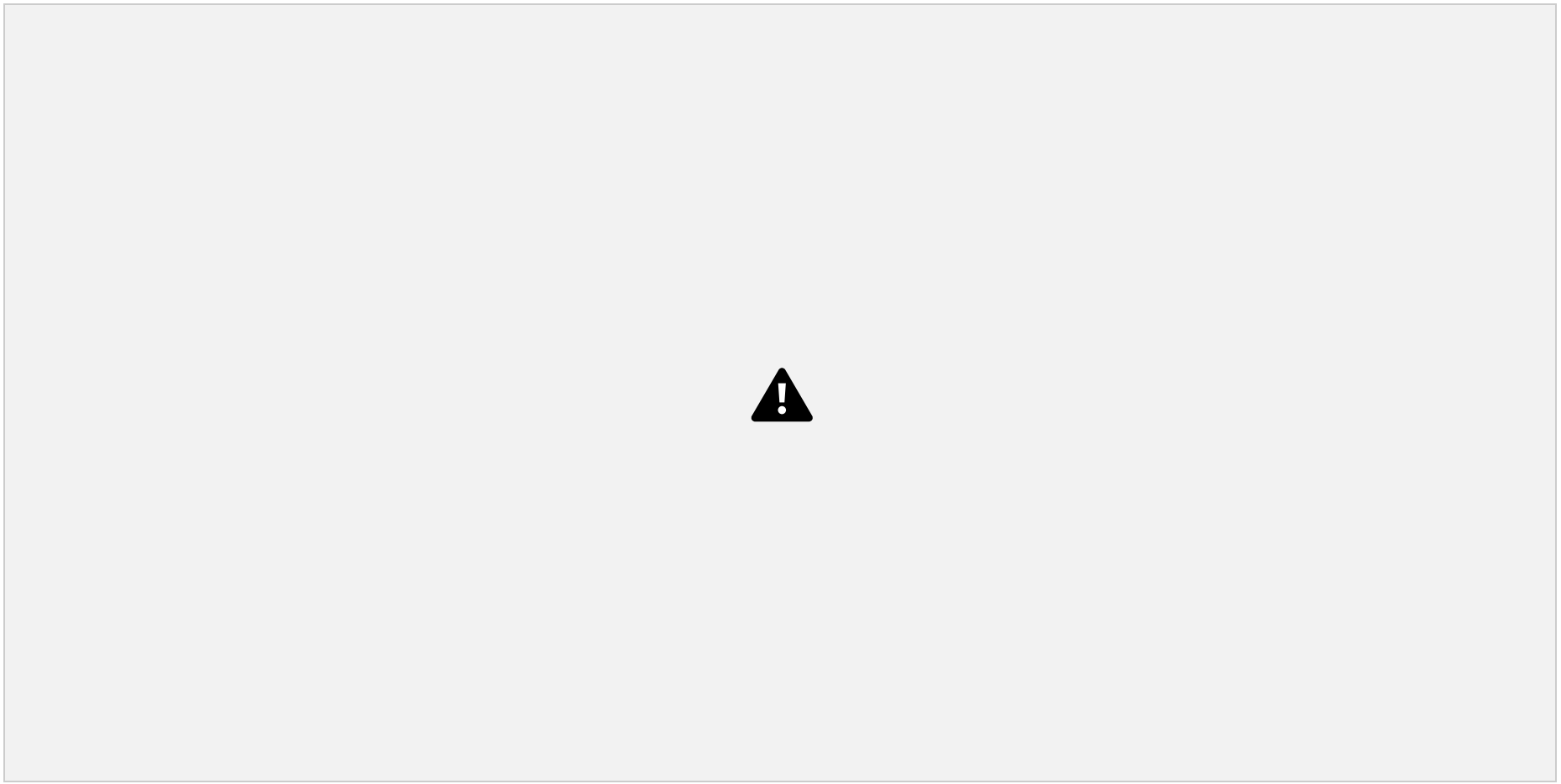


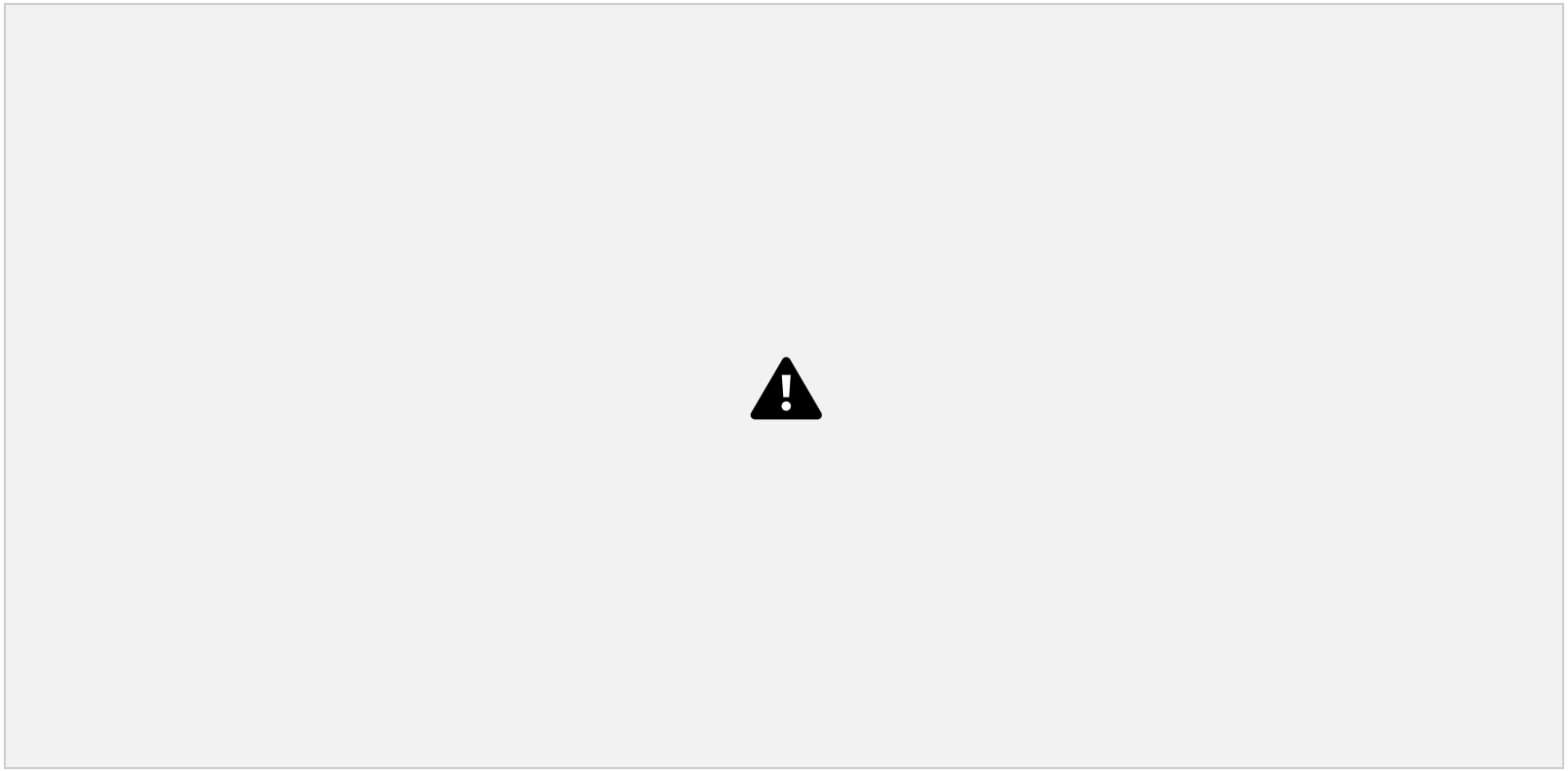


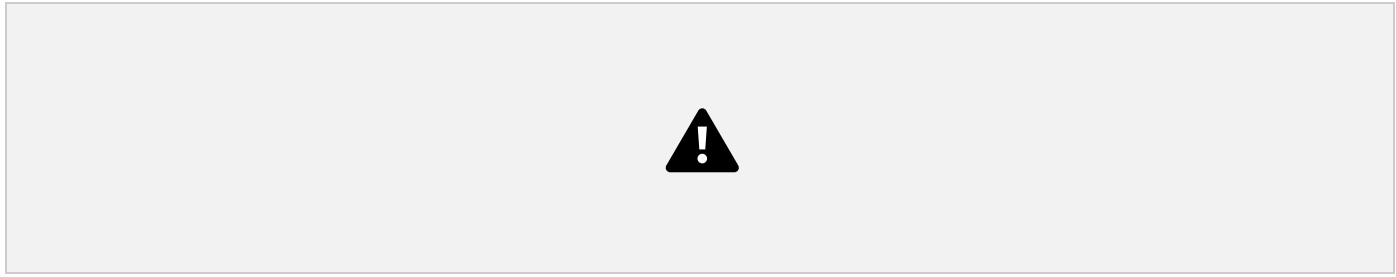
16



17







21