Unit 1 – Introduction to Logic Circuits

FUNDAMENTAL CONCEPTS

BOOLEAN ALGEBRA

- This is the foundation for designing and analyzing digital systems. It deals with the case where variables assume only one of two values: TRUE (usually represented by the symbol '**1**'), and FALSE (usually represented by the symbol '**0**'). This is also called Two-valued Boolean Algebra or Switching Algebra.
- A circuit consisting of switches can be represented in terms of Boolean algebraic equations. The equations can be then manipulated into the form representing the simplest circuit. The circuit may then be immediately drawn from the equations. This powerful method first appeared in: "A symbolic Analysis of Relay and Switching Circuits", Claude E. Shannon, *Transactions of the AIEE*, vol. 57, no. 12, Dec. 1938, pp. 713-721.

BASIC OPERATIONS

- X and Y are Boolean variables. Boolean variables are used to represent the inputs or outputs of a digital circuit.

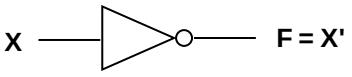
OPERATION	BOOLEAN EXPRESSION	OPERATION
NOT	$X'(\ or\ \bar{X}\ )$	Logical negation
AND	$X.Y$	Logical conjunction of two statements
OR	$X+Y$	Logical disjunction of two statements

TRUTH TABLES AND LOGIC GATES

- **Truth Table**: A tabular listing of function values for all possible combinations of values on its input arguments. If there are n inputs, there are 2^n possible combinations.
- **Logic Gates**: Hardware components that produce a logic 1 or logic 0 depending on the state of inputs. Boolean functions can be implemented with logic gates.

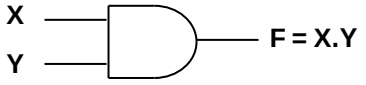
NOT gate:

X	F = X'
0	1
1	0



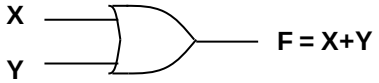
2-input AND gate:

X	Y	F = X.Y
0	0	0
0	1	0
1	0	0
1	1	1

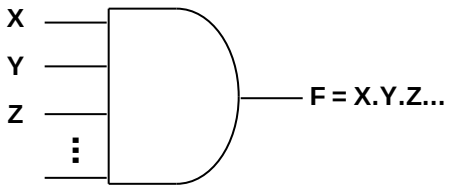


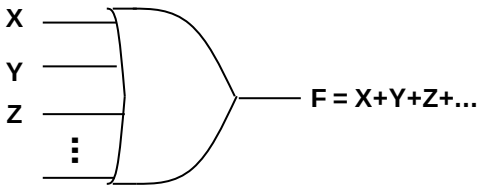
2-input OR gate:

X	Y	F = X+Y
0	0	0
0	1	1
1	0	1
1	1	1



- Logic Gates (AND, OR) can have multiple inputs:





AXIOMS

0.0 = 0	1.1 = 1	0.1 = 1.0 = 0	$\bar{0} = 1$
1+1=1	0+0 = 0	1+0 = 0+1 = 1	$\bar{1} = 0$

THEOREMS

Variable dominant rule	$X.1 = X$ $X + 0 = X$
Commutative rule	$X.Y = Y.X$ $X + Y = Y + X$
Complement rule	$X.\bar{X} = 0$ $X + \bar{X} = 1$
Idempotency	$X.X = X$ $X + X = X$
Identity Element	$X.0 = 0$ $X + 1 = 1$
Double negation	$\bar{\bar{X}} = X$
Associative rule	$X.(Y.Z) = (X.Y).Z$ $X + (Y + Z) = (X + Y) + Z$
Distributive rule	$X.(Y + Z) = X.Y + X.Z$ $\boldsymbol{X + Y.Z = (X + Y).(X + Z)}$

Other Theorems

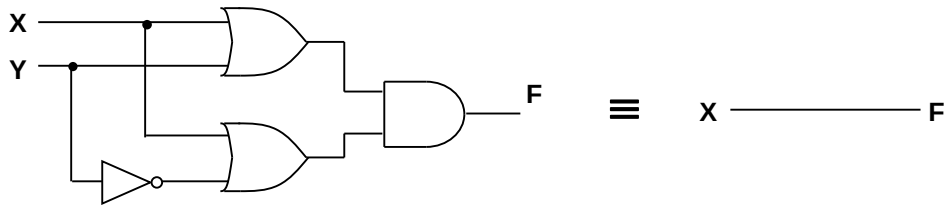
Absorption	$X.(X + Y) = X.X + X.Y = X + X.Y = X.(1 + Y) = X$ $X + X.Y = X.(1 + Y) = X$
Adjacency	$X.Y + X.\bar{Y} = X$ $(X + Y)(X + \bar{Y}) = X$
Consensus	$X.Y + \bar{X}Z + YZ = XY + \bar{X}Z$ $(X + Y)(\bar{X} + Z)(Y + Z) = (X + Y)(\bar{X} + Z)$ Corollary: $(X + Y)(\bar{X} + Z) = \bar{X}Y + XZ$
DeMorgan	$\overline{X.Y} = \bar{X} + \bar{Y}$, $\quad \overline{X.Y.Z\ldots} = \bar{X} + \bar{Y} + \bar{Z} + \cdots$ $\overline{X + Y} = \bar{X}.\bar{Y}$, $\quad \overline{X + Y + Z + \ldots} = \bar{X}.\bar{Y}.\bar{Z}\ldots$
Simplification	$X.(\bar{X} + Y) = X.Y$ $X + \bar{X}Y = X + Y$

- A useful application of the theorems is on the simplification of Boolean functions which leads to the reduction of the amount of logic gates:

✓ **Example:**
$F = (A + \bar{B}C + D + EF)(A + \bar{B}C + \overline{D + EF})$
$F = (X + Y)(X + \bar{Y}), \quad X = A + \bar{B}C, \quad Y = D + EF$
$F = (X + Y)(X + \bar{Y}) = X$
$\rightarrow F = A + \bar{B}C$

✓ **Example:**
$F = \overline{(\overline{X + Y})Z + X\bar{Y}Z}$
$F = \overline{\bar{X}\bar{Y}Z + X\bar{Y}Z}$
$F = \overline{\bar{Y}Z(X + \bar{X})}$
$\rightarrow F = \overline{\bar{Y}Z} = Y + \bar{Z}$

✓ **Example:**
$F = (X + Y)(X + \bar{Y}) = XX + X\bar{Y} + YX + Y\bar{Y} = X + X(\bar{Y} + Y) = X + X = X$



✓ **Example:**
$F = x_1x_2 + \overline{x_1}\,\overline{x_2} + x_2\overline{x_1}$
$F = x_1x_2 + \overline{x_1}(x_2 + \overline{x_2}) = x_1x_2 + \overline{x_1}$
$F = \overline{x_1} + x_1x_2 = (\overline{x_1} + x_1)(\overline{x_1} + x_2)$
$\rightarrow F = \overline{x_1} + x_2$

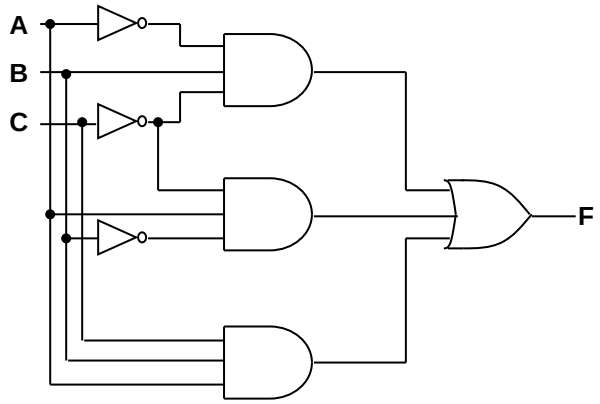
✓ **Example:**
$F = \overline{A(B + \bar{C}) + \bar{A}}$
$F = \overline{A(B + \bar{C})}.A = (\bar{A} + \overline{B + \bar{C}}).A$
$\rightarrow F = (\overline{B + \bar{C}}).A = A\bar{B}C$

DERIVING BOOLEAN FUNCTIONS FROM TRUTH TABLES

Using 1s:

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

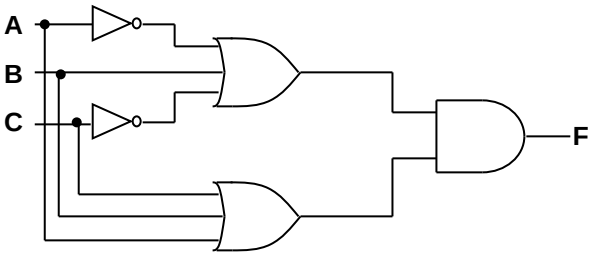
$$F = \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$



Using 0s:

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

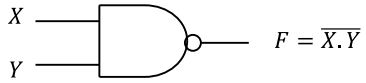
$$F = (A + B + C)(\bar{A} + B + \bar{C})$$



Other Logic Gates

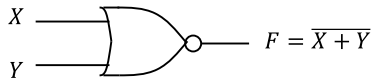
2-input NAND gate

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



2-input NOR gate

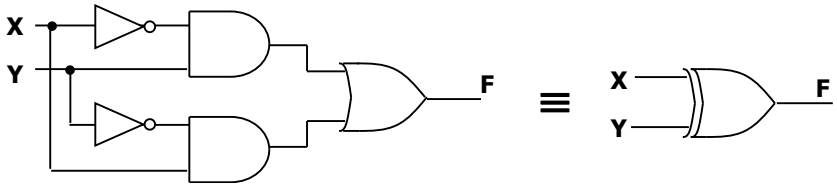
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



2-input XOR gate

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

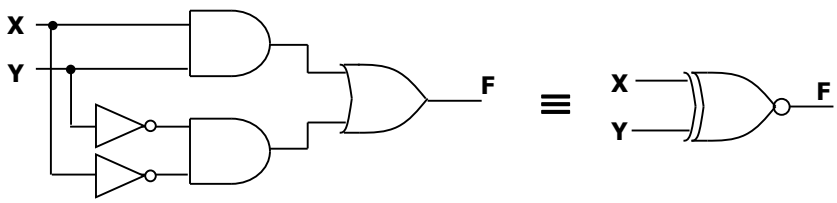
$$F = \bar{X}Y + X\bar{Y} = X \oplus Y$$



2-input XNOR gate:

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

$$F = XY + \bar{X}\bar{Y} = \overline{X \oplus Y}$$



SUM OF PRODUCTS (SOP) AND PRODUCT OF SUMS (POS) USING MINTERMS AND MAXTERMS:

MINTERMS and MAXTERMS (3 variable function)

	x_1	x_2	x_3	**Minterms**	**Maxterms**
0	0	0	0	$m_0 = \overline{x_1}\,\overline{x_2}\,\overline{x_3}$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x_1}\,\overline{x_2}x_3$	$M_1 = x_1 + x_2 + \overline{x_3}$
2	0	1	0	$m_2 = \overline{x_1}\,x_2\overline{x_3}$	$M_2 = x_1 + \overline{x_2} + x_3$
3	0	1	1	$m_3 = \overline{x_1}\,x_2x_3$	$M_3 = x_1 + \overline{x_2} + \overline{x_3}$
4	1	0	0	$m_4 = x_1\overline{x_2}\,\overline{x_3}$	$M_4 = \overline{x_1} + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x_2}x_3$	$M_5 = \overline{x_1} + x_2 + \overline{x_3}$
6	1	1	0	$m_6 = x_1x_2\overline{x_3}$	$M_6 = \overline{x_1} + \overline{x_2} + x_3$
7	1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \overline{x_1} + \overline{x_2} + \overline{x_3}$

- A function with n variables can have up to 2^n minterms (or 2^n maxterms) from m_0 to m_{2^n-1} (or from M_0 to M_{2^n-1})
- Note that: $\overline{m_i} = M_i$.
- Also, for n variables, the total number of different functions is 2^{2^n}.

- A function can be expressed as a sum of minterms or as a product of maxterms:
  - ✓ When a minterm evaluates to 1, it implies that the function evaluates to 1.
    Example: $f(x,y) = xy + \bar{x}\bar{y}$. Here, $f(x,y) = 1$ if $xy = 00{,}11$. Thus, the minterms are m_0 and m_3. $f(x,y) = m_0 + m_3$
  - ✓ When a maxterm evaluates to 0, it implies that the function evaluates to 0.
    Example: $f(x,y) = (x + \bar{y})(\bar{x} + y)$. Here, $f(x,y) = 0$ if $xy = 10{,}01$. Thus, the maxterms are M_1 and M_2. $f(x,y) = M_1M_2$

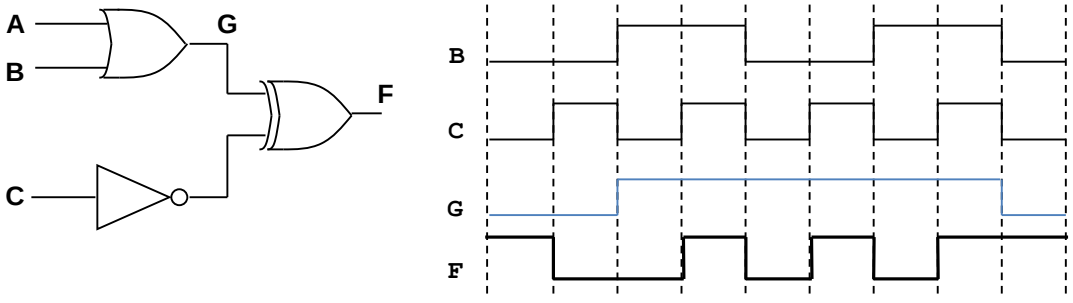
- A sum of products (SOP) that include only minterms or a product of sums (POS) that contain only maxterms are called Canonical Forms.
- If a SOP includes terms that are not minterms (or a POS includes terms that are not maxterms), they are called non-canonical forms. For example:
  - ✓ $F(x_1,x_2,x_3) = x_1x_2x_3 + \overline{x_1}\,\overline{x_2}$
  - ✓ $F(x_1,x_2,x_3) = (x_1 + x_2 + x_3)(\overline{x_1} + \overline{x_2})$
  - ✓ $F(x_1,x_2,x_3) = x_1x_2x_3 + x_1\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} + (\overline{x_1} + x_2 + x_3)$

Example:

X	**Y**	**Z**	**F**	Sum of Products
0	0	0	0	$F = \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + X\bar{Y}Z + XY\bar{Z}$
0	0	1	1	$F(X,Y,Z) = \sum(m_1, m_4, m_5, m_6).$
0	1	0	0	$F(X,Y,Z) = \sum m(1,4,5,6)$ Also: $\bar{F}(X,Y,Z) = \sum m(0,2,3,7)$
0	1	1	0	
1	0	0	1	Product of Sums
1	0	1	1	$F = (X + Y + Z)(X + \bar{Y} + Z)(X + \bar{Y} + \bar{Z})(\bar{X} + \bar{Y} + \bar{Z})$
1	1	0	1	$F(X,Y,Z) = \prod(M_0, M_2, M_3, M_7).$
1	1	1	0	$F(X,Y,Z) = \prod M(0,2,3,7)$ Also: $\bar{F}(X,Y,Z) = \prod M(1,4,5,6)$

- Note how $F(X,Y,Z) = \sum m(1,4,5,6) = \prod M(0,2,3,7)$.

TIMING DIAGRAMS



DIGITAL DESIGN

- When designing digital circuits, specifications are provided for the desired circuit.
- Boolean variables are used to represent the state of inputs (e.g.: switches, buttons) and outputs (e.g.: LEDs, locks).

- **Truth Table Method**: This is a simple methodology of the design of digital circuits:
  - ✓ The relationship between the inputs and outputs of the circuit is determined by completing the Truth Table.
  - ✓ The Boolean functions are then generated and simplified. We can then sketch the resulting circuit using logic gates.
  - ✓ This simple, yet powerful technique works very well for a small number of inputs. For large number of inputs, this method is impractical as the number of entries grow exponentially. For a circuit requiring 16 input Boolean variables, the truth table would need 65536 entries.

  - ✓ **Example** (Majority gate): Design a 3-input circuit that generates a '0' when two or more of the inputs are '0'. It also generates a '1' when two or more of the inputs are '1'. Provide the truth table, the Boolean equation, and sketch the logic circuit.

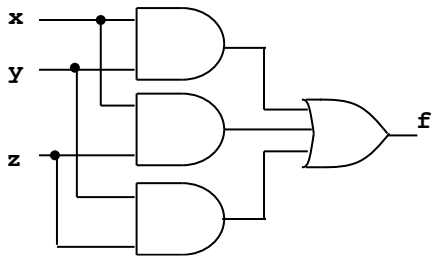
x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$f = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$

$f = \bar{x}yz + x(\bar{y}z + y\bar{z} + yz) = \bar{x}yz + x(\bar{y}z + y)$
$f = \bar{x}yz + x(\bar{y} + y)(z + y)$

$f = \bar{x}yz + xz + xy = xz + y(x + \bar{x}z)$

$f = xz + y(x + \bar{x})(x + z) = xz + yx + yz$



XILINX FPGA IMPLEMENTATION - DESIGN FLOW

- **Design Entry**: Here, the circuit is specified via a Hardware Description Language (HDL), Schematic, or a waveform. The process of verification of the HDL syntax of schematic connections is called *Synthesis*.
- **Behavioral Simulation**: This is a crucial step. Your Design Entry might be 'error-free' syntax-wise, but it might not work as expected. Here, we provide time-varying stimuli to the inputs of a logic circuit and verify that the outputs are correct. When the stimuli is written in HDL, it is called a '*test-bench*'. This process is very similar to using a signal generator to create the inputs, and using a scope to visualize the outputs over time.
- **Physical Mapping:** Here we specify which inputs and outputs map to the specific components of the FPGA we selected and the Printed Circuit Board (PCB) that houses the FPGA. In Xilinx Vivado, this is done via a file called Constraints File (.xdc)
- **Timing Simulation**: Behavioral Simulation only simulates the circuit 'logically', i.e., it does not take into account analog and electrical effects. Timing simulation does consider the delay that exist between inputs and outputs, and therefore it is very useful to determine glitches, hazards, etc.
- **Implementation**: Here, we "program" the FPGA. In this step, we grab a configuration file (called 'bitstream') and then download it onto the FPGA configuration memory.

PRACTICE EXERCISES

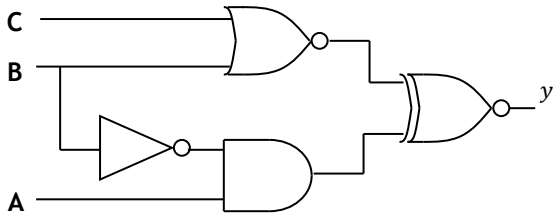
- Simplify the following functions:
  - ✓ $F = \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + X\bar{Y}Z + XY\bar{Z}$
  - ✓ $F(X,Y,Z) = \sum(m_0, m_2, m_6)$
  - ✓ $F = (X + Y + Z)(X + Y + \bar{Z})$
  - ✓ $F = (\bar{A}B + C + D)(\bar{A}B + D)$
  - ✓ $F = A(C + \bar{D}B) + \bar{A}$
  - ✓ $F(X,Y,Z) = \prod(M_3, M_4, M_7)$
- Given $f = x + (y \oplus z)$, express f using the canonical Sum of Products (SOP), i.e., sum of minterms.
- Using Boolean Algebra Theorems, prove that:
  - ✓ The XOR operation is associative: $a \oplus b \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = b \oplus (a \oplus c)$.
  - ✓ $b(a \oplus c) = (ba) \oplus (bc)$
  - ✓ $x \oplus y = \bar{x} \oplus \bar{y}$, $x \oplus \bar{y} = \bar{x} \oplus y = \overline{x \oplus y}$

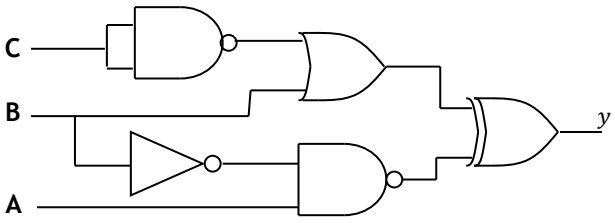
- Provide the Boolean functions and sketch the logic circuit. Use the two representations: i) Sum of Products, ii) Product of Sums. Also, provide the minterms and maxterms representations.

A	B	C	F1	F2I	F3	F4	F5	F6	F7
0	0	0	0	1	0	1	0	1	0
0	0	1	1	0	1	1	1	0	0
0	1	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1
1	0	0	1	0	1	0	0	1	0
1	0	1	0	1	0	0	1	0	0
1	1	0	1	1	0	0	0	1	1
1	1	1	1	1	1	0	1	0	1

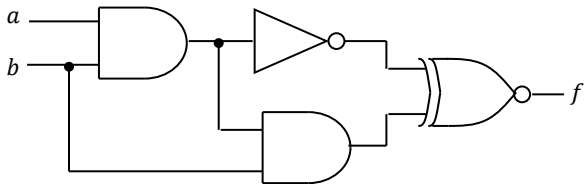
- Simplify the function: $f(x,y,z) = f_1 + f_2.f_3$, where $f_1(x,y,z) = \sum m(3,5)$, $f_2(x,y,z) = \sum m(3,5,6)$, $f_3(x,y,z) = \prod M(0,3,5,7)$.

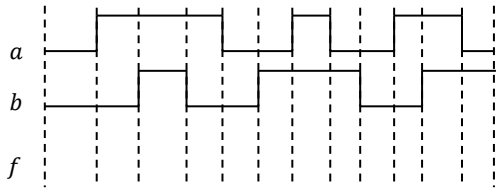
- Obtain the logic function (and minimize if possible) of the following circuits:





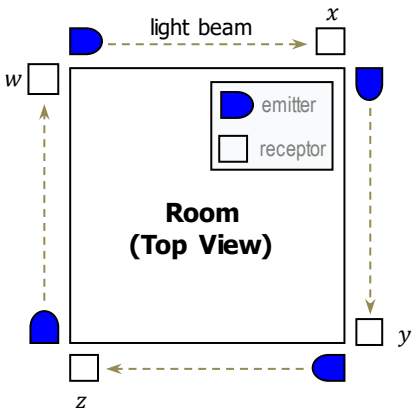
- Complete the timing diagram of the following circuit:





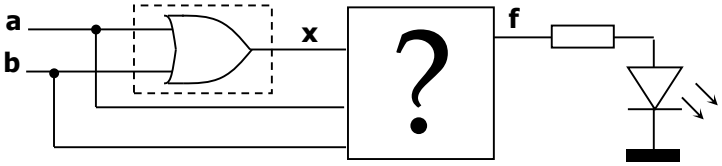
- Design a circuit that detects if somebody enters a room protected by 4 optical sensors (x,y,z,w). The circuit must activate an alarm ($f = 1$) if one or more of the sensors activates. The sensor consists of an emitter and a receptor. A sensor activates (e.g.: $x = 1$) when the light beam is blocked from its receptor.

  ✓ Complete the truth table and provide the simplified Boolean expression.
  ✓ Sketch the minimized logic circuit.



- A doctoral student is defending his Dissertation. A 4-member committee is in charge of evaluating the work. The members vote on whether to accept or reject the work. A simple majority vote is required. In case of a tie, the chair of the committee makes the final determination. If we assign (x,y,z,w) to the vote of each committee member (w represents the vote of the chair of the committee), where '1' means accept, and '0' reject, design a circuit that generates $f = 1$ when the committee accepts the work, and $f = 0$ if the work is rejected.

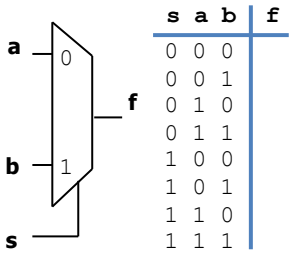
- Design a circuit that verifies the logical operation of the OR gate. f = '1' (LED ON) if the OR gate works properly. Assumption: when the OR gate is not working, it generates 1's instead of 0's and vice versa. Tip: First, generate the truth table.



- Security combination: A lock only opens when the 8 switches are set as in the figure. Get the function that opens the lock (a logical '1' is generated) when the switches are configured as in the figure. Each switch represents a Boolean variable. Here, an open lock is represented by an LED that is ON.

- The circuit (trapezoid) has the logic function: $f = \bar{s}a + sb$.
  ✓ Complete the truth table for f, and sketch the logic circuit.
  ✓ We can use several instances of this circuit to implement different functions. For the given inputs, provide the resulting function g (minimize the function).

in1	in2	in3	in4	in5	in6	in7
x_1	1	x_3	x_1	0	x_3	x_2



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