Notes - Chapter 1

BOOLEAN ALGEBRA

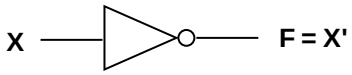
Let's deal with the case when variables assume only one of two values: TRUE (represented by the symbol '1'), and FALSE (represented by the symbol '0'). This is also called Two-valued Boolean Algebra or Switching Algebra.

Basic Operations: (X and Y are Boolean variables)

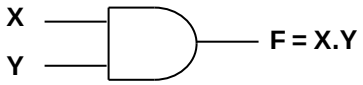
OPERATION	BOOLEAN EXPRESSION	OPERATION
NOT	$X'(\ or\ \bar{X}\ )$	Logical negation
AND	$X.Y$	Logical conjunction of two statements
OR	$X+Y$	Logical disjunction of two statements

Truth Tables and logic symbols:

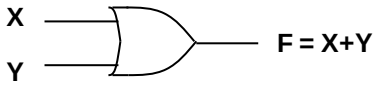
X	F = X'
0	1
1	0



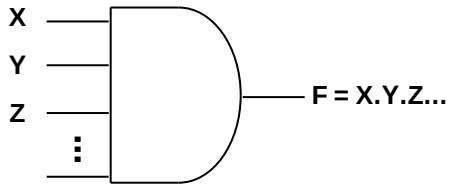
X	Y	F = X.Y
0	0	0
0	1	0
1	0	0
1	1	1

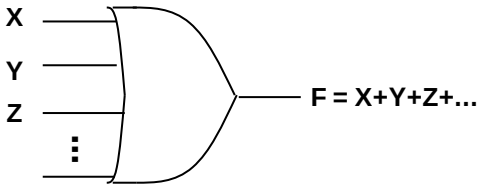


X	Y	F = X+Y
0	0	0
0	1	1
1	0	1
1	1	1



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





Boolean Algebra Postulates (axioms)

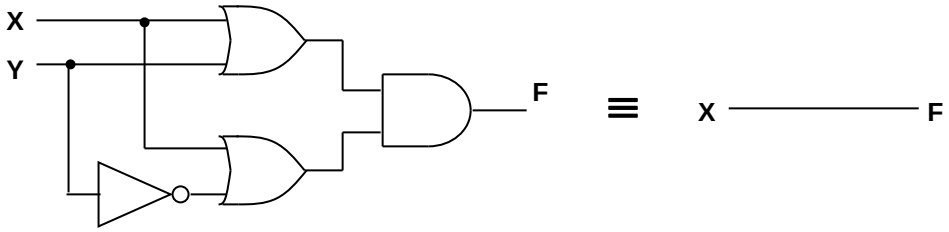
Variable dominant rule	$X.1 = X$ $X + 0 = X$
Commutative rule	$X.Y = Y.X$ $X + Y = Y + X$
Distributive rule	$X.(Y + Z) = X.Y + X.Z$ $X + Y.Z = (X + Y).(X + Z)$ $(*)$
Complement rule	$X.\bar{X} = 0$ $X + \bar{X} = 1$

Boolean Algebra Theorems

Absorption	$X.(X+Y) = X.X + X.Y = X + X.Y = X.(1+Y) = X$ $X + X.Y = X.(1+Y) = X$
Adjacency	$X.Y + X.\bar{Y} = X$ $(X+Y)(X+\bar{Y}) = X$
Associative	$X.(Y.Z) = (X.Y).Z$ $X + (Y+Z) = (X+Y) + Z$
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	$\overline{X.Y} = \bar{X} + \bar{Y}$, $\quad \overline{X.Y.Z\ldots} = \bar{X} + \bar{Y} + \bar{Z} + \cdots$ $\overline{X+Y} = \bar{X}.\bar{Y}$, $\quad \overline{X+Y+Z+\ldots} = \bar{X}.\bar{Y}.\bar{Z} + \cdots$
Double negation	$\bar{\bar{X}} = X$
Idempotency	$X.X = X$ $X + X = X$
Identity	$X.0 = 0$ $X + 1 = 1$
Simplification	$X.(\bar{X}+Y) = X.Y$ $X + \bar{X}Y = X + Y$

Example:

$$F = (X+Y)(X+\bar{Y}) = XX + X\bar{Y} + YX + Y\bar{Y} = X + X(\bar{Y}+Y) = X + X = X$$

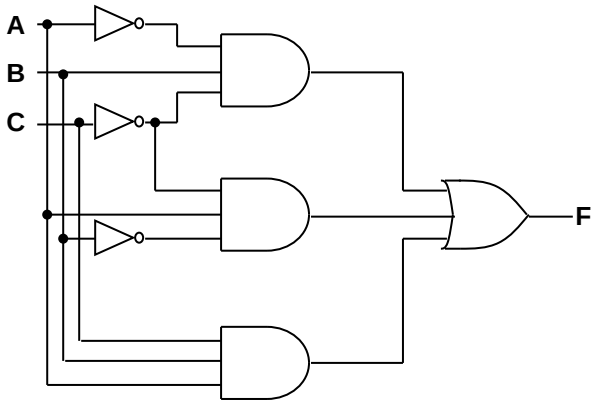


DERIVING BOOLEAN FUNCTIONS FROM TRUTH TABLES:

Using 1s:

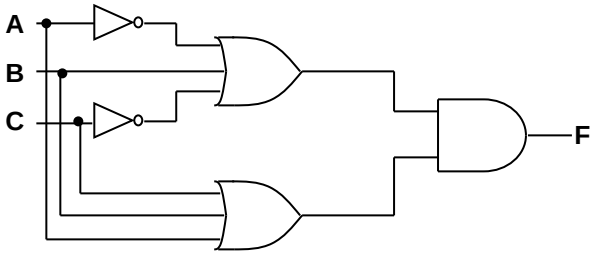
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

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



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



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

Sum of Products (SOP) and Product of Sums (POS) using minterms and maxterms:

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: $\bar{F}(X,Y,Z) = \prod M(1,4,5,6)$

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, however 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 FPGA, this is done via a file called Constraints File (.ucf).
- **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.

PRACTICE EXERCISES

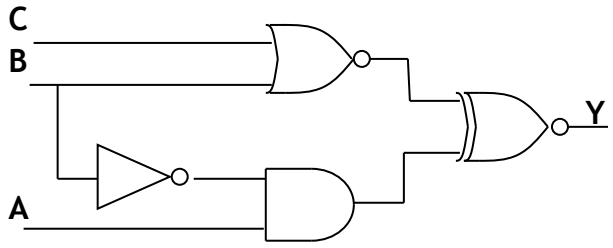
- Simplify the following functions:

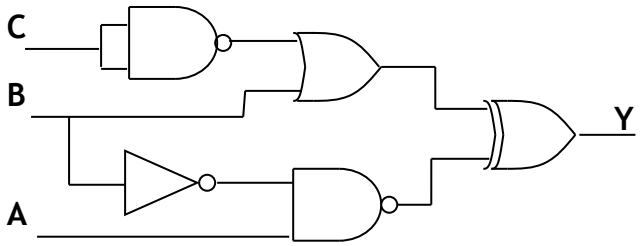
✓ $F = \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + X\bar{Y}Z + XY\bar{Z}$	✓ $F = (X + Y + Z)(X + Y + \bar{Z})$
✓ $F(X,Y,Z) = \sum(m_0, m_2, m_6).$	✓ $F = (\bar{A}B + C + D)(\bar{A}B + D)$
✓ $F(X,Y,Z) = \prod(M_3, M_4, M_7)$	✓ $F = A(C + \bar{D}B) + \bar{A}$

- 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.

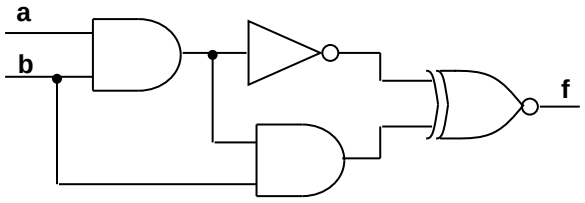
A	B	C	F1	F2	F3	F4	F5	F6	F7
0	0	0	0	1	0	1	0	1	0
0	0	1	1	0	1	1	1	0	0
0	1	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1
1	0	0	1	0	1	0	0	1	0
1	0	1	0	1	0	0	1	0	0
1	1	0	1	1	0	0	0	1	1
1	1	1	1	1	1	0	1	0	1

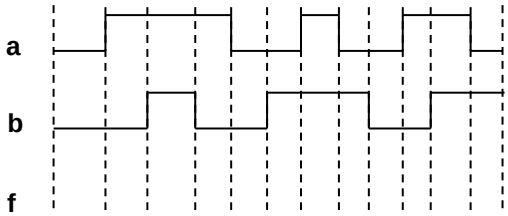
- Obtain the logic function (and minimize if possible) of the following circuits:



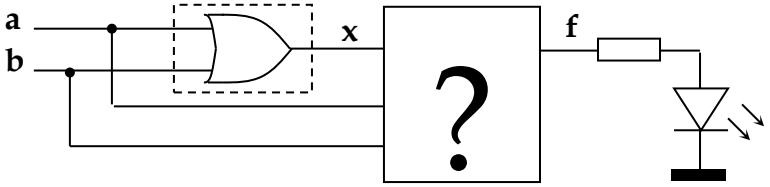


- Determine the timing diagram of the following circuit:





- 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 is generating 1's instead of 0's and vice versa. Tip: First, generate the truth table.



- Security combination: We have a lock that only opens when we set eight (8) switches in the following configuration:

Each switch represents a Boolean variable. Get the function that opens the lock (a logical '1' is generated) when the switches are configured as in the figure. The ISE project 'sec_comb.zip' implements this system, you can verify it on your Nexys Board. Here, an open lock is represented by an LED that is ON.