PHY 307: Electronics-II

Tutorial-4

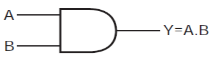
Logic gates, Boolean algebra

Logic gates: The logic gate is the most basic building block of any digital system, e.g., mobile phones, computers etc.. Each one of the basic logic gates is a piece of an electronic circuit (hardware), which implements basic logic expression. The laws of Boolean algebra could be used to do manipulation with binary variables and simplify logic expressions. The three basic logic gates are the OR gate, the AND gate and the NOT gate.

OR gate: An OR gate performs an ORing operation on two or more than two logic variables. The OR operation on two independent logic variables A and B is written as $Y = A+B$ and reads as Y equals A OR B and not as A plus B. The output of an OR gate is HIGH when any of its inputs is HIGH.

An OR gate.

AND gate: An AND gate is a logic circuit having two or more inputs and one output. The output of an AND gate is HIGH only when all of its inputs are in the HIGH state.



An AND gate.

NOT gate: A NOT gate is a one-input, one-output logic circuit whose output is always the complement of the input, i.e., a LOW input produces a HIGH output, and vice versa.

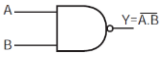
A NOT gate

EXCLUSIVE-OR Gate: The EXCLUSIVE-OR (EX-OR) gate, is a two-input, one-output gate. The output of an EX-OR gate is a logic '1' when the inputs are unlike and a logic '0' when the inputs are like.

$$Y = A'\oplus B' = A'B+AB'$$

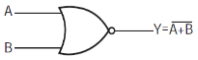
An EX-OR gate

NAND Gate: NAND stands for NOT AND. An AND gate followed by a NOT circuit makes it a NAND gate. $Y= (A.B)'$



An NAND gate

NOR Gate: NOR stands for NOT OR. An OR gate followed by a NO T circuit makes it a NOR gate. $Y =(A+B)'$



A NOR gate

Forms of Boolean Expressions:

- Sum-of-products form (SOP) – first the product (AND) terms are formed then these are summed (OR) – eg: $ABC + DEF + GHI$
- Product-of-sum form (POS) – first the sum (OR) terms are formed then the products are taken (AND) – eg: $(A+B+C)\ (D+E+F)\ (G+H+I)$
- It is possible to convert between these two forms using Boolean algebra (DeMorgan's)
- Canonical form is not efficient but sometimes useful in analysis and design
- In an expression in canonical form, every variable appears in every term

$$f(A,B,C, D) = ABC'D + AB'CD + AB'CD'$$

PHY 307: Electronics-II

Tutorial-4

Boolean algebra

1. **Boolean algebra:** Derive the Boolean expression and construct the switching circuit for column #4 and #5 in the truth table shown below.

   Homeowork: Practice the same problem for the column#6.

1 *A*	2 *B*	3 *C*	4	5	6
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	0	0

2. **Boolean algebra:** Simplify the following expression:

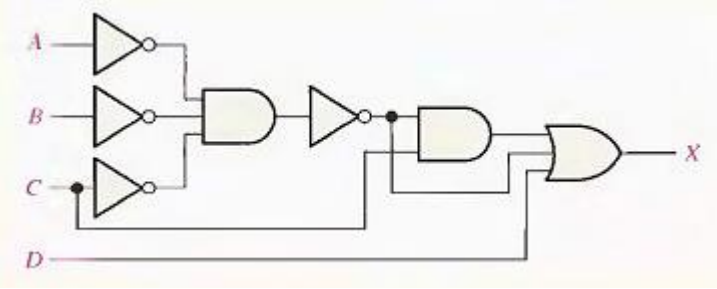
   (i). (A.B+C.D) [(A'+B').(C' +D')]

   (ii). A.B.C.D+A.B.C'.D'+A.B.C.D'+A.B.C'.D+A.B.C.D.E +A.B.C'.D'.E' +A.B.C'.D.E

3. **Logic system using NOR-gates:** Use **NOR**-gates only to devise the logic system:

   (i). $Z = P'.Q + P.(Q + R)$; (ii). $Z = A.B' + B.C' + C.D'$

4. **Logic circuit to Boolean expression:** Reduce the combinational logic circuit (shown below) to a minimum form,



5. **Conversion to standard Sum-of-products (SOP) form:** Convert the following Boolean expression into standard SOP form,

AB’C + A’B’ + ABC’D

6. **Conversion to standard Product-of-sums (POS) form:** Convert the following Boolean expression into standard POS form,

(A’ + B + C).(B’ + C + D’).(A + B’ + C’ + D)