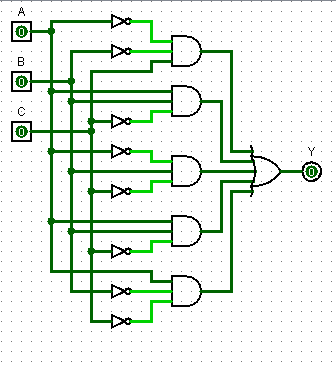
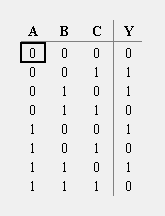
**Q:1 Simplify given expression using standard sum of product, also show step by step process of building a circuit and designing a truth table.**

**1/ F1(A, B, C) = ~A ~B C + B ~C + A ~C**

**SOL: ~A~B C + B ~C (A + ~A) + A ~C (B + ~B’)**

**~A ~B C + B ~C A + B ~C ~A + A ~C B + A ~C ~B**

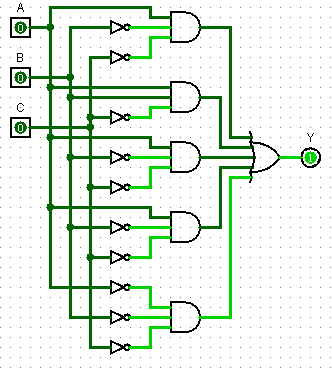


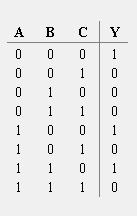


**2/ F2(A, B, C) = A ~B ~C + A ~C + ~B ~C**

**SOL: A ~B ~C + A ~C (B + ~B) + ~B ~C (A + ~A)**

**A ~B ~C + A B ~C + A ~B ~C + A ~B ~C + ~A ~B ~C**



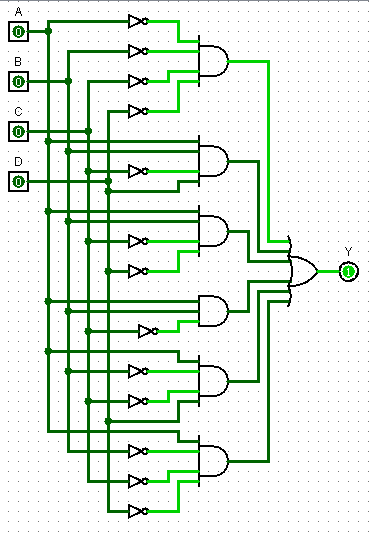


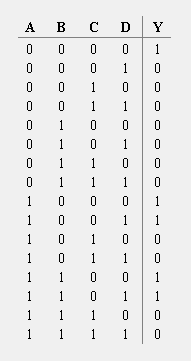
**3/ F1(A, B, C, D) =~A ~B ~C ~D + A B ~C + A ~C**

**SOL: = ~A ~B ~C ~D + A B ~C (D + ~D) + A ~C (B + ~B)**

**~A ~B ~C ~D + A B ~C D + A B ~C ~D + A B ~C + A ~B ~C (D + ~D)**

**~A ~B ~C ~D + A B ~C D + A B ~C ~D + A B ~C + A ~B ~C D + A ~B ~C ~D**



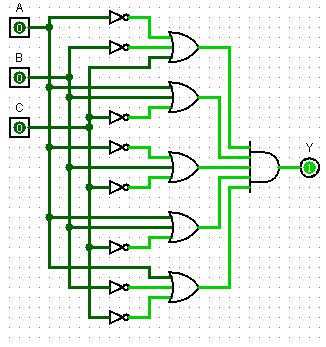


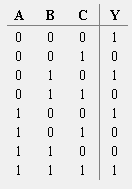
**Q:2 Simplify given expression using standard product of sum, also show step by step process of building a circuit and designing a truth table.**

**1/ F1 +(A, B, C) = (~A + ~B + C) (B + ~C) (A+~C)**

**SOL: (~A + ~B + C) (B + ~C) (A + ~A) (A + ~C) (B + ~B)**

**(~A + ~B + C) (A + B + ~C) (~A + B + ~C) (A + B + ~C) (A + ~B + ~C)**

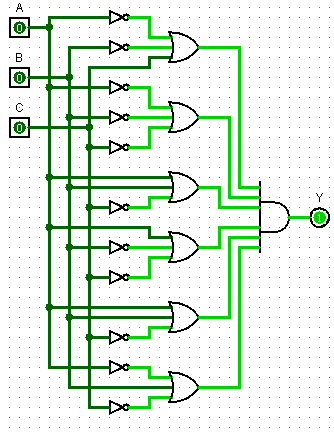
****

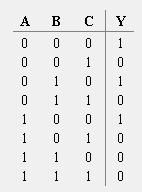
****

**2/ F (A, B, C) = (~A + ~B) (A + ~C) (B+ ~C)**

**SOL: (~A + ~B) (C + ~C) (A + ~C) (B + ~B) (B+ ~C) (A + ~A)**

**(~A + ~B + C) (~A + ~B + ~C) (A + B + ~C) (A + ~B + ~C) (A + B + ~C) (~A + B + ~C)**

****

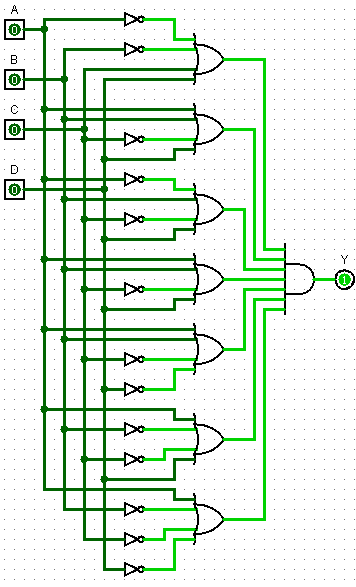
****

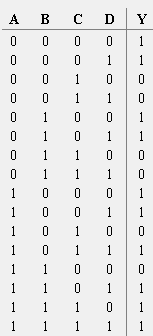
**3/ F (A, B, C, D) = (~A + ~B + C + D) (B + ~C + D) (A + ~C)**

**SOL: (~A + ~B + C + D) ( B + ~C + D) (A + ~A) (A + ~C) (B + ~B)**

**(~A + ~B + C + D) (A + B + ~C + D) (~A + B + ~C +D) (A + B + ~C) (D + ~D) (A + ~B + ~C) (D + ~D)**

**(~A + ~B + C + D) (A + B + ~C + D) (~A + B + ~C +D) (A + B + ~C + D) (A + B + ~C + ~D) (A + ~B + ~C + D) ( A + ~B + ~C + ~D)**

****

****

**Q:3 Why do we convert SOP and POS into their canonical form?**

**Ans: We usually perform SOP and POS of Boolean expressions to make the simplification, evaluation and implementation. SOP and POS also helps to solve Boolean expression more easier and its also a systematic way.**

**Q:4 What is Combinational Analysis?**

**Ans: The Combinational Analysis module of Logisim allows you to convert between these three representations in all directions. It is a particularly handy way of creating and understanding circuits with a handful of one-bit inputs and outputs.**

**Q:5 What are minterms and maxterms?**

**Ans: A minterm is a Boolean AND function containing exactly one instance of each input variable or its inverse. A maxterm is a Boolean OR function with exactly one instance of each variable or its inverse. For a combinational logic circuit with n input variables, there are 2 n possible minterms and 2 n possible maxterms**