DSM Tutorial Quiz Questions (Pre-Midsem)

Consider the equation $L + M = N$ where $L = (312)_8$ and $N = (451)_8$. Find the value of M represent the answer in hexadecimal.

Convert 3210_5 to base 3

First find, $3210_5 = 430_{10}$

Then find, $430_{10} = 120221_3$

Convert 212_3 to base 5

First, $212_3 = 23_{10}$

Then find, $23_{10} = 43_5$

1. **Write the divisibility rule of 3 for binary numbers**

A. A binary number N can be expressed as:
N = (b[n-1])*2^(n-1) + (b[n-2])*2^(n-2) + ………… + b[0]*2^0
where b[i] can either be 0 or 1.
We can say that 2^k and 2^(k+2) have the same remainder when divided by 3 because their difference is divisible by 3.
2^(k+2) - 2^k = 2^k (4 - 1) = 2^k (3) —> divisible by 3

So, 2^0, 2^2, 2^4…..will have the same remainder when divided by 3,i.e,1
Similarly, 2^1, 2^3, 2^5…..will have the same remainder when divided by 3,i.e,2

Hence, the remainder of b[i]*2^i will be b[i]%3 if i is even. It will be (2*b[i])%3 if i is odd.
For the number N to be divisible by 3, the sum of these remainders should be divisible by 3, i.e, R = b[0]+2*b[1]+b[2]+2*b[3]+b[4]+2*b[5]+.... should be divisible by 3.

For example, let us consider the binary number 11011. Here, b[0] = 1, b[1] = 1, b[2] = 0, b[3] = 1, b[4] = 1. R = 1*1+2*1+1*0+2*1+1*1 = 6 which is divisible by 3. Hence, the given binary number is also divisible by 3.

Question: "*Product of two n-bit two's-complement numbers requires fewer than 2n bits to represent it.*"

Find a case in which this statement is not correct.

Answer: multiplying (-2^{n-1}) with itself will need $2n$ bits to represent $2^{(2n-2)}$

Solution:

Consider the number (-2^{n-1}), it belongs to n-bit two's-complement system and can be represented as 100....00.

On multiplying -2^{n-1} with itself:

$$-2^{n-1} \times -2^{n-1} = 2^{(2n-2)}$$

In a 2's complement system, $2^{(2n-2)}$ requires $2n$ bits to represent it. (1 sign bit + $2n-1$ magnitude bits)

Question: How many unique Boolean functions can exist for n number of inputs?

Answer: 2^{2^n}

Solution:

For n inputs, the possible number of min-terms are 2^n let this be called k

Any boolean function is a is combination of minterms (ordering does not matter).

$$\text{No of boolean function considering exactly 0 minterm} = {}^kC_0$$
$$\text{No of boolean function considering exactly 1 minterm} = {}^kC_1$$
$$\text{No of boolean function considering exactly 2 minterm} = {}^kC_2$$
$$...$$
$$\text{No of boolean function considering exactly k minterm} = {}^kC_k$$

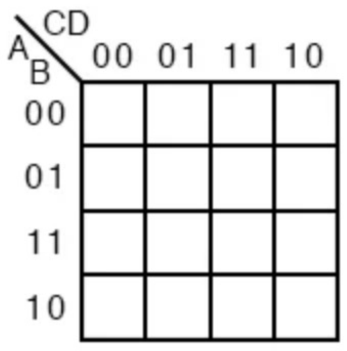
Therefore, total no of boolean functions possible is:

$${}^kC_0 + {}^kC_1 + {}^kC_2 + ... {}^kC_k = (1+1)^k = 2^k$$

substituting $k = 2^n$, the total number of possible boolean functions is 2^{2^n}

1. What is the condition for an equation of a sum of minterms in a 4-variable Karnaugh map (K-map) to be **non-simplifiable** based on the positioning of 1s and 0s? Explain why this happens.

A. Assume that two 1s are adjacent (includes cyclic adjacency). If two 1s are adjacent, they will be in either the same row or the same column:



Adjacent 1s in the Same Row:

- The 1s can occupy any of the column pairs: (1,2), (2,3), (3,4), or (4,1).
- For pairs (1,2) and (3,4), the adjacent columns share a common variable, either C or C′..
- For pairs (2,3) and (4,1), the adjacent columns share D or D′.
- Each of these pairs has three variables in common, and the terms formed will be of the type xyzw′+xyzw allowing for simplification by removing w.

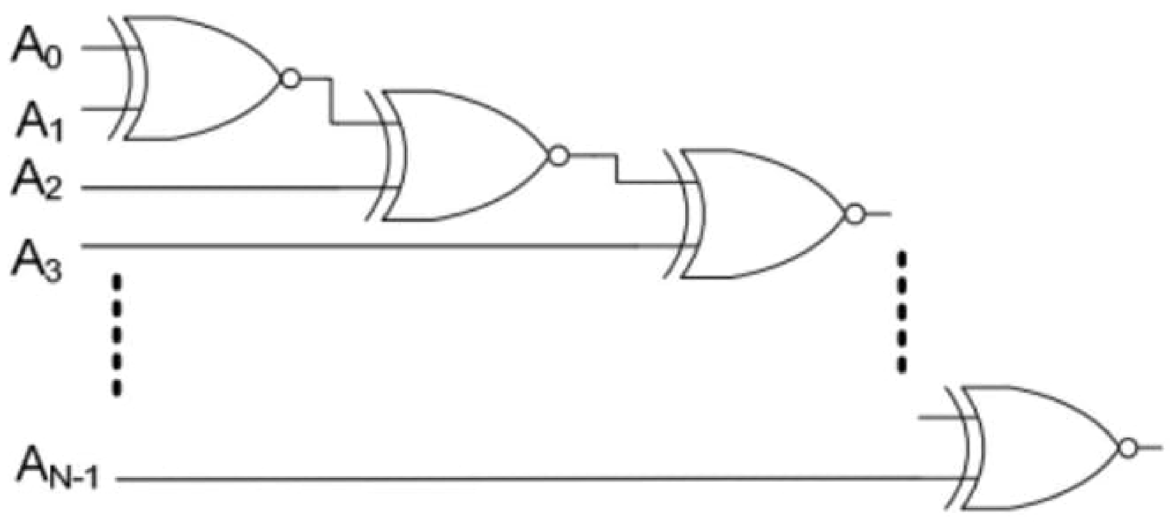
Adjacent 1s in the Same Column:

- The same logic applies if the 1s are adjacent in a column, where adjacent rows share either A and A′ or B and B′.
- The terms will simplify similarly, leading to a reduction.

Therefore, to prevent any simplification, **no two 1s should be adjacent** either directly or cyclically. If this condition is met, no grouping or reduction can occur, and the equation remains non-simplifiable.

Q2

N number of XNOR gates are connected as shown below. How does this circuit work? Explain?



Ans

two different ways based on N-value. (a) N is odd (b) N is even
(a) If N is odd, there will be even number of XNOR gates in the circuit. Take an example of N=3, So there are 2 XNOR Gates. The two bubbles will get cancelled and this works as XOR. Same works for any odd N. So if N is odd it works as XOR Gate.
(b) If N is even, the circuit works as XNOR Gate. (Apply the same logic). One extra bubble will be there to make XOR to XNOR. You may verify the same for N=4.

2. You are playing a best-of-3 tennis match. Develop a Boolean expression that represents the condition in which you'll win the match. Additionally, provide a logic gate implementation of this expression using only a single type of logic gate.

A. Let Si represent the result of the i-th set, where Si=1 if you win the set and Si=0 if you lose. The Boolean expression for winning the match is:

F=S1S2+S1′S2S3+S1S2′S3

This expression means you win if:

- You win the first two sets, or
- You lose one of the first two sets but win the third set.

Simplifying further:

F=S1S3+S2S3+S1S2

This represents the possible ways to win the match.

Implementation using NAND Gates:

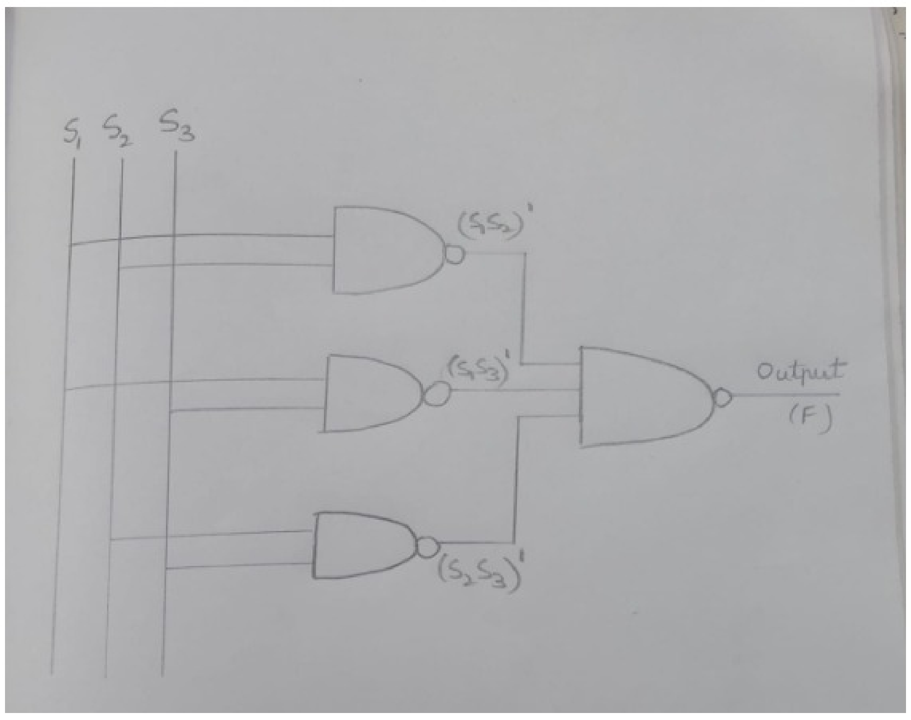
Since the goal is to implement this expression using only NAND gates, we apply **De Morgan's law**. The NAND operation is defined as:

(ABC)′=A′+B′+C′

Using this, we can rewrite the Boolean expression as:

F=((S1S3)′(S2S3)′(S1S2)′)′

This representation uses only NAND gates to express the condition for winning the match.

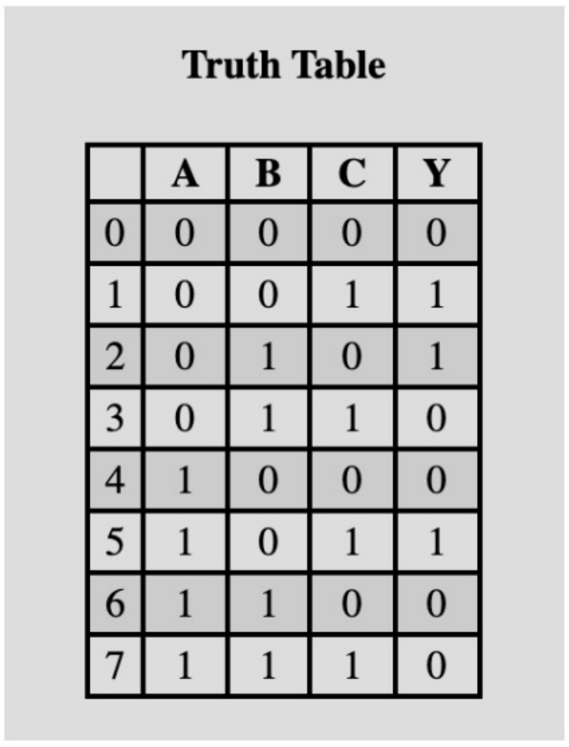


Question 2

Given the following truth table, represent it as a logic gate circuit with only 3 AND/OR gates. Assume that you have

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

as inputs.



Truth Table

	A	B	C	Y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Answer

$$\overline{A}BC + \overline{AB}C + A\overline{B}C$$

Apply the Distributive Law $(AB + AC = A(B + C))$

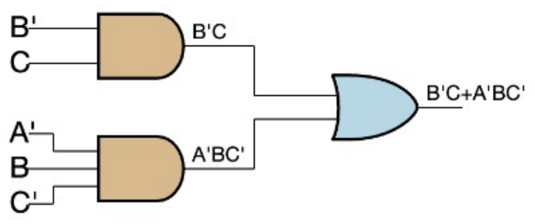
$$\overline{B}C(\overline{A} + A) + \overline{AB}C$$

Apply the Complement Law $(A + \overline{A} = 1)$

$$\overline{B}C \cdot 1 + \overline{AB}C$$

Apply the Identity Law $(A1 = A)$

$$\overline{B}C + \overline{AB}C$$



Q1

The grouping of k adjacent cells, in a N variable K-Map will lead to a term of ____ literals?

Ans

N - log_2 (k)

Soln

In a N variable k-map,
Having a group of size 1, i.e single cell means that the cell requires the description of all variables to describe the group output.
Having a group of size 2, means that 1 variable is same across the 2 cells, and it only requires us N-1 variables to describe the group output.
Having a group of size 4, means that 2 variables are same across the 4 cells, and it only requires us N-2 variables to describe the group output.
Having a group of size 8, means that 3 variables are same across the 8 cells, and it only requires us N-3 variables to describe the group output.

…. (observing the pattern) ….

Having a group of size k, means that log_2 (k) variables are same across the k cells, and it only requires us N - log_2 (k) variables to describe the group output.

Q1

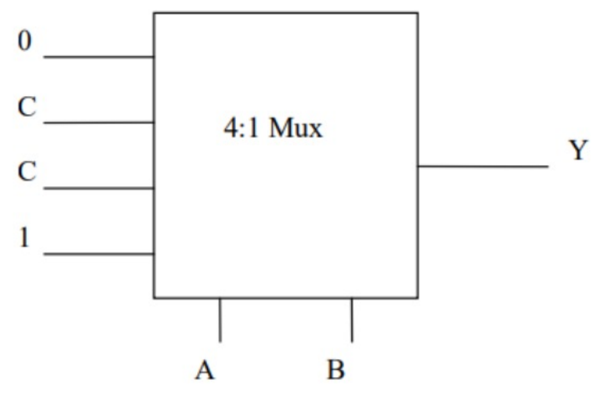
Design a circuit for 3-input majority function using a 4:1 Mux

Ans

The majority function gives 1 if the input has more number of 1s than zeros.

1. If A=B=0, the output will be zero irrespective of C
2. If A=B=1, the output is 1 irrespective of C.
3. But if A=1,B=0 or A=0,B=1, we can not decide the majority without knowing C. if C is 1, majority is 1, otherwise majority is 0.

So I0 = 0, I1=I2 = C and I3 =1



Q2

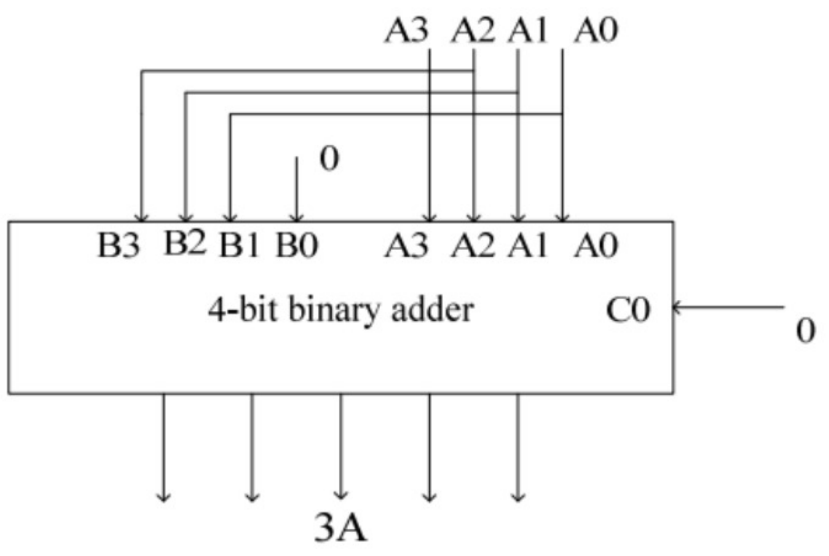
Using a 4-bit binary adder, design a circuit which multiplies the input by 3?
Note: you need not handle overflow bit (i.e 5th bit).

Ans

Lets say input is X (4 bits).
We need to find 3X
Can be written as 3X = 2X + X

2X is simply a left shift of bits.

Adding 2X with X using 4-bit binary adder looks like this:



Note to TAs: You can ignore the left most output channel since it is a overflow bit.

1. Design an OR gate using half adders

A. A half adder takes two inputs and provides two outputs:
    i. Sum (S) = A XOR B
    ii. Carry (C) = A AND B

Required: A OR B
We know that: A OR B = A XOR B XOR (A AND B)
To implement the OR gate, we need 2 half adders:
-> Let inputs be A and B
-> First Half Adder: Connect A and B to the inputs of the half adder. The half adder will produce S and C.
-> Second Half Adder: Connect S and C to the inputs of the half adder. The half adder will produce S XOR C (Sum) and S AND B (Carry).
S XOR C = A XOR B XOR (A AND B) = A OR B
Therefore, the sum bit of the second half adder is the output of the newly designed OR gate.