

Question no. 1

Simplify the following Boolean logic functions to the format in sum of product first and then create a truth table in Excel for each as the verification reference of the circuit design, finally implement by online tools at <https://circuitverse.org/simulator>

$$a. f = (A + C' + D')(B' + C' + D)(A + B' + C')$$

Solution :-

$$f = (A + C' + D')(B' + C' + D)(A + B' + C')$$

$$f = (A * B' * A) + (A * B' * B') + (A * B' * C') + (C' * B' * A) + (C' * B' * B') + (C' * B' * C') + (D' * B' * A) + (D' * B' * B') + (D' * B' * C')$$

$$f = (A * B') + 0 + 0 + 0 + 0 + 0 + (C' * B') + 0 + 0 + 0$$

$$f = (A * B') + (C' * B')$$

A	B'	C'	$f=(A*B')+(C'*B')$
1	0	1	0
1	0	0	0
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	1
0	1	1	1
0	1	0	1

$$b. f = (Z + X)(Z + Y)(Y + X)$$

Solutions :

Apply the Distributive Law:

$$f = (z + x)(z' + y')(y' + x)$$

Use the Distributive Law again to expand the expression:

$$f = (z * z' + z * y' + x * z' + x * y')(y' + x)$$

Simplify using the fact that $z * z' = 0$ and $x * x = x$:

$$f = (0 + zy' + xz' + xy')(y' + x)$$

Apply the Distributive Law once more to expand the expression:

$$f = 0y' + zy'y' + xz'y' + xy'y' + 0x + zy' + 0xz' + xyz'$$

Simplify further by noticing that $y * y' = 0$ and $x * 0 = 0$:

$$f = 0 + 0 + 0 + 0 + 0 + zy' + 0 + xyz'$$

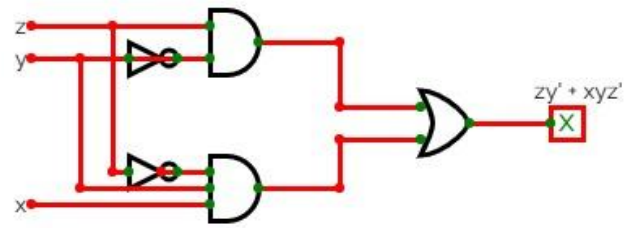
Finally, simplify the expression:

$$f = zy' + xyz'$$

So, the simplified expression is:

$$f = zy' + xyz'$$

x	y	z	zy'	xyz'	f=zy'+xyz'
0	1	0	0	0	0
0	1	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	0	0	0	1	1
1	0	1	0	0	0



c. $f = (X + Y)Z + XYZ$

Apply De Morgan's Law to $(x + y)$:

$$(x + y)' = x'y'$$

Now, substitute this into the original expression:

$$f = (x'y')z + x'y'z'$$

Factor out the common term $x'y'$:

$$f = x'y'(z + z')$$

Since $z + z' = 1$, the expression simplifies to:

$$f = x'y'(1)$$

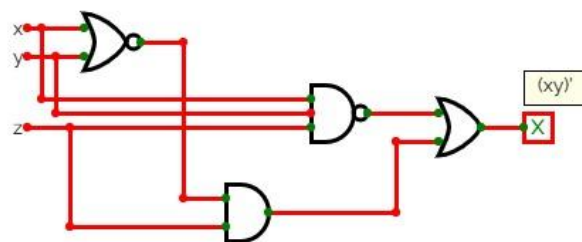
Any term ANDed with 1 is the term itself, so the expression further simplifies to:

$$f = x'y'$$

So, the simplified expression is:

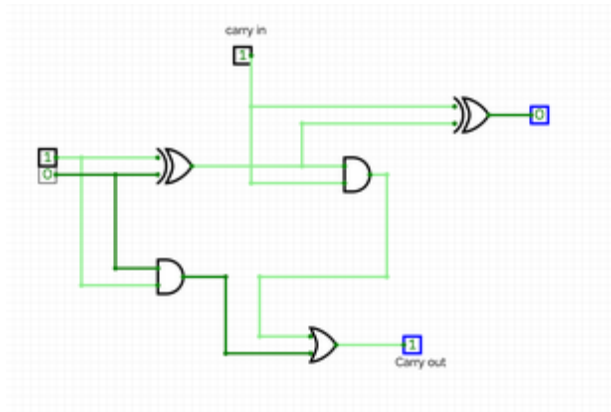
$$f = x'y'$$

x	y	$f=x'y'$
0	1	1
0	1	0
1	0	0
1	0	0

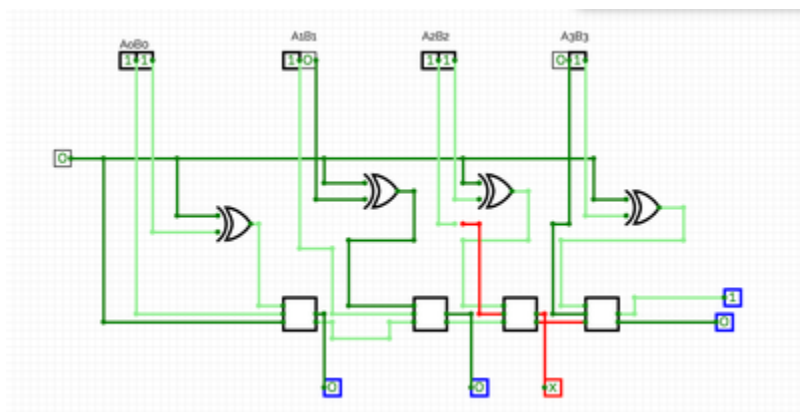


Question no. 2

Designing a 4-bit adder-subtractor circuit involves using four cascaded full adders, each with three inputs (A, B, and C_{in}) and two outputs (S and C_{out}). The circuit operates using logic gates such as AND, OR, and XOR.



Pic : full adder



Pic: 4-bit adder

When performing addition ($S=0$), the B bits are directly input, with the carry-in of the least significant bit (C_0) set to 0. The adder simply sums A and B.

For subtraction ($S=1$), the B input is inverted (to obtain the 1's complement), and a carry-in of 1 is introduced into the least significant bit adder (to create the 2's complement). The adder then adds A to the 2's complement of B, effectively subtracting B from A.

In testing the design with specific cases:

(a) To calculate $A + B = 7 + (-3)$, A (7) is represented as 0111 in binary, and B (-3) is the 2's complement of 3, which is 1101 in binary. In this addition ($S=0$) scenario, the expected result is 1000, which is -8 in decimal. The sign bit indicates it's negative, so to find the absolute value, you take the 2's complement of 1000, resulting in 8 in decimal.

(b) For $A - B = -6 - (-1)$, A (-6) is represented as the 2's complement of 6, which is 1010 in binary, and B (-1) is the 2's complement of 1, which is 1111 in binary. In this subtraction ($S=1$) scenario, the expected result is 1001, which is -7 in decimal. In 2's complement arithmetic, the interpretation of results depends on the sign bit. If the sign bit is 1 (indicating a negative number), the result must be converted to its absolute value by taking the 2's complement.

Question no.3

Here we have given

Load 104

Add 105

Store 106

We can translate these to machine code as:

Load 104 translates to:

Binary: 0001 0000 1000

Hex: 1 08

Add 105 translates to:

Binary: 0011 0000 1001

Hex: 3 09

Store 106 translates to:

Binary: 0010 0000 1010

Hex: 2 0A

therefore the machine code for the given assembly instruction are 1 08 3 09 2 0A

The operation to be performed is determined by the binary representation of the instruction code. We'll represent these instructions in 4-bit format (based on the table provided) and use the remaining 12 bits for the address.

Instruction	4-bit code	Address	Machine Code
Load	0001	0000 1000	0001 0000 1000
Add	0011	0000 1001	0011 0000 1001
Share	0010	0000 1010	0010 0000 1010

Now let's design a circuit:

https://circuitverse.org/simulator

McAfee Security Dell Practice | Geeksfor... Gmail YouTube Maps Computer Organiza... Reading 1b: Counti... HTML Styles CSS Other favorites

CircuitVerse Project Circuit Tools Help

Untitled

Main x +

CIRCUIT ELEMENTS

PROPERTIES

PROJECT PROPERTIES

Project:
Untitled

Circuit:
Main

Clock Time (ms):
- 500 +

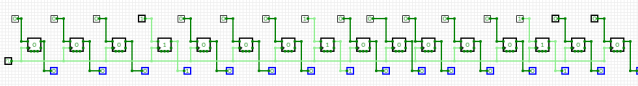
Clock Enabled: ☒

Lite Mode: ☐

Edit Layout

Delete Circuit

TIMING DIAGRAM



10:33 PM
11/2/2023