**Digital Logic Design Exercise**

1. (Laws and Theorems of Boolean Algebra) Prove using Boolean algebra that *a**c*  *ab*  *ac*  *a**b*  *a**c*  *ab*  *b**c*. Write the particular law you are using in each step.

Proof: *a**c*  *ab*  *ac*  *a**b*

= a’c’ + ab + (ac + a’b’)

= a’c’ + ab + (ac + a’b’ + b’c)

= a’c’ + ab + ac + a’b’ + b’c + b’c

= (a’c’ + b’c + a’b’) + (ab + b’c + ac)

= (a’c’ + b’c) + (ab + b’c)

= a’c + ab + b’c

1. Prove using Boolean algebra that (*a*  *c*)(*a*  *c*)(*b*  *c*  *d*)(*a*  *b*  *d*)  (*a*

* *c*)(*a*  *c*)(*b*  *d*). Write the particular law you are using in each step.

Proof: (*a*  *c*)(*a*  *c*)(*b*  *c*  *d*)(*a*  *b*  *d*)

= (a + c) (a’ + c’) ((ac) + (b’ + d’))

= (a + c) (a’ + c’) ac + (a + c) (a’ + c’) (b’ + d’)

= (a + c) (a’ac + c’ac) + (a + c) (a’ + c’) (b’ + d’)

= (a + c) (0 + 0) + (a + c) (a’ + c’) (b’ + d’) =

= 0 + (a + c) (a’ + c’) (b’ + d’)

= (a + c) (a’ + c’) (b’ + d’)

1. Use Karnaugh map to simplify function *f* (*a*, *b*, *c*, *d*)  *m*(0,1, 2, 3, 4, 5, 7,8,12)

*  *d*(10,11). List **all possible** minimal two-level **sum of products** expressions. Show the switching functions. No need for the diagram.

ab

cd 00 01 11 10

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 1 | 1 |  |  |
| 1 | 1 |  | x |
| 1 |  |  | x |

00

01

11

10

F = a’d + a’b’ + c’d’ or F = a’d + b’c + c’d’ or F = a'd + c'd' + b'd'

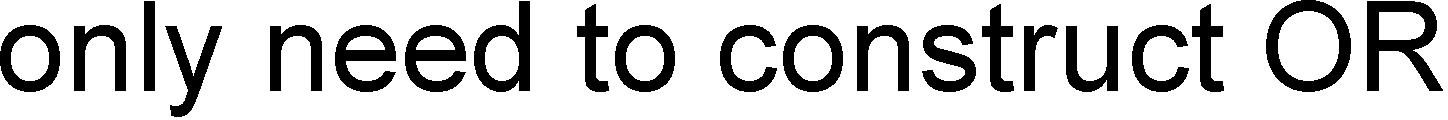
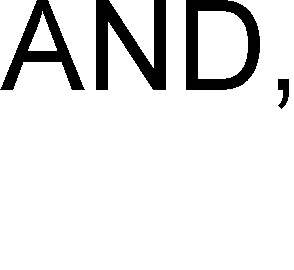
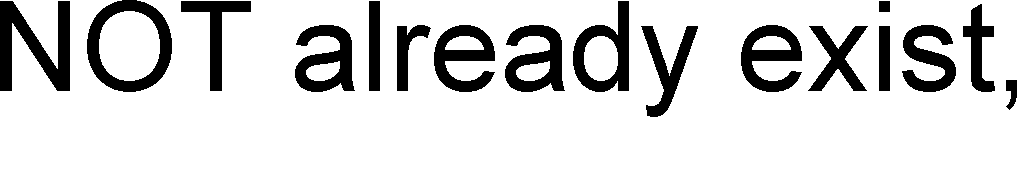
4.Define what is the Universal Set of Gates on the space below.

A **set** G of **gates** is **universal** for a computational model if any function. computable in this model can be computed by a circuit that uses only **gates** in G. For classical circuits, both {AND, NOT} and the single-**gate set** {NAND} are **universal**.

5.Universal Set of Gates: Check if the set in the following list is universal and explain your decision. Assuming constants 0 and 1 are available as inputs.

Solution:

1. {AND, NOT}



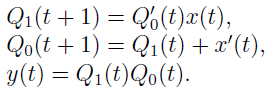
1. {NAND}



1. {XOR, NOT}

Not universal, because AND or OR can not be constructed using XOR & NOT

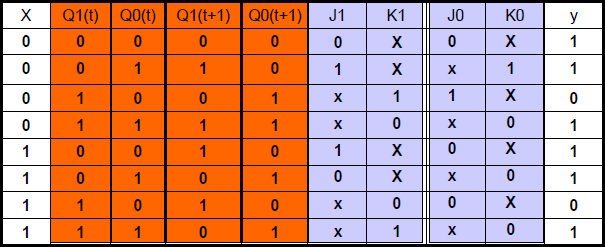
6.A state machine is described by the following state equations.



1. Write the state table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | Q1(t) | Q0(t) | Q1(t+1) | Q0(t+1) | y |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

1. Design the system with two JK flip-flops and a minimal AND-OR-NOT network.



\_\_\_