LE4.5.1: 2-input functions

4/4 points (ungraded)

Consider the 2×2 K-map needed to hold the truth table for a 2-input Boolean function.

If the truth table for the 2-input function contained only a single "1" in the output column, what is the maximum number of prime implicants that could be circled in the corresponding K-map?

Maximum number of prime-implicants: 1 ✔ Answer: 1

If the truth table for the 2-input function contained exactly two "1s" in the output column, what is the maximum number of prime implicants that could be circled in the corresponding K-map?

Maximum number of prime-implicants: 2 ✔ Answer: 2

If the truth table for the 2-input function contained exactly three "1s" in the output column, what is the maximum number of prime implicants that could be circled in the corresponding K-map?

Maximum number of prime-implicants: 2 ✔ Answer: 2

If you only had a supply of 2-input NAND gates to build a circuit, what is the *minimum* number of 2-input NANDs you would need to implement any arbitrary 2-input Boolean function? Hint: think about your answers to the questions above and what they imply about a minimal sum-of-products expression for an arbitrary 2-input function. Then think about how to implement a sum-of-products circuits using only NANDs.

Minimum number of 2-input NANDs needed: 5 ✔ Answer: 5

Explanation

A 2×2 K-map containing a single "1" would have exactly one prime implicant.

A 2×2 K-map containing two "1s" would have at most two prime implicants, e.g., when the two 1's were diagonal from each other.

A 2×2 K-map containing three "1s" would have at exactly two prime implicants, forming an "L" in some orientation.

Thus the minimal sum-of-products expression would have at most two product terms, which we could implement using three 2-input NANDs using the NAND/NAND form shown in the video. And we might need to build up to two inverters to compute the NOT of one or both of the inputs. We can build an inverter by connecting the signal to be inverted to both inputs of the 2-input NAND. So we'd need at most five 2-input NAND gates.

Submit

Answers are displayed within the problem

LE4.5.2: Minimal Sum Of Products

3 points possible (ungraded)

The 3-input boolean function $G(A, B, C)$ computes $\overline{A} \cdot \overline{C} + A \cdot \overline{B} + \overline{B} \cdot \overline{C}$.

A) How many 1's are there in the output column of G's 8-row truth table?

- 3
- 4
- 5
- 6
- none of the above

B) A minimal sum-of-products expression for G is:

- $A \cdot \overline{B} \cdot C + \overline{A}$
- $A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C}$
- $A \cdot \overline{B} + \overline{A} \cdot \overline{C}$
- $\overline{A} \cdot \overline{C} + \overline{B} \cdot \overline{C}$
- all of the above

- none of the above

C) There's good news and bad news: the bad news is that the stockroom only has G gates. The good news is that it has as many as you need. Using only combinational circuits built from G gates, one can implement

- any function (G is universal)
- only functions with 3 inputs or less
- only functions with the same truth table as G
- only inverting functions
- only non-inverting functions

Submit

Discussion

Hide Discussion

Topic: 4. Combinational Logic / LE4.5

Add a Post

Show all posts by recent activity

question and answer are misconducts
the question asks for the minimum number of 2 input nand and the answer is we need at most 5 2-... 2

Minimum number of 2-input NANDs needed?
I think the minimum number of NANDs is 9. As to a 2-input Boolean function, say f(A,B), then f(A,B)... 2

G(A,B,0) == Nand ?
I can see that if I only take the terms where C=0 in the Truth table of G ⟶ We get NAND's truth ... 2

Why just A.~B+~A.~C?
For part B of question LE4.5.2, could someone explain to me why the answer is just A.~B+~A.~C b... 2

[STAFF] Minimum number of 2-input NANDs needed to implement arbitrary 2-input Boolean function 8

It seems to me the minimum number of 2-input NANDs needed to implement any arbitrary 2-input ...