Homework

CSC 28 – Discrete Structures for Computer Science

This work is not turned in but should be done before any quiz on it. To maximize your quiz performance: attend lectures, do readings, start homework early, ask for help when you need it, review problems and solutions before the quiz. If anything in this assignment does not make sense, please ask for help.

Reading) My text notes are on the course webpage. If this is not enough, there are lots of tutorials on the web, such as this one by Mustafa Aziz: `https://krovetz.net/28/module_4/tutorial_kmaps.pdf`.

Problems) A shorthand for specifying a truth table is to give the truth table a name and list the variables used in it. For example, $f(a,b,c)$ is naming the truth table f and saying it has three variables, a, b, and c. This is then set equal to a list of numbers where each number in the list represents a 1 in the output of the truth table in the row indicated by the number. (A "sigma" is attached as a prefix to the list by convention.) All other rows in the table are 0. For example, $f(a,b,c) = \Sigma(0,1,5)$ represents a table that has a 1 in its output when *abc* equals 000, 001, or 101 and has a 0 for all other combinations of *abc*.

1) Write an equivalent boolean expression in disjunctive normal form (DNF) for the truth table $f(a,b,c) = \Sigma(1,2,3,5,6,7)$. Use a Karnaugh map and the method seen in class to simplify the boolean expression. Draw a logic circuit matching your simplified circuit.

2) Write an equivalent boolean expression in disjunctive normal form (DNF) for the truth table $f(a,b,c,d) = \Sigma(0,1,3,9,10,11)$. Use a Karnaugh map and the method seen in class to simplify the boolean expression. Draw a logic circuit matching your simplified circuit.

3) Design a circuit for the top segment of a seven-segment display. It should take four wires *abcd* representing the number n (interpret *abcd* as an unsigned binary number). The circuit should output 1 when $0 \leq n \leq 9$ and n is a number for which the top segment should be turned on in a seven-segment display and should output 0 when $0 \leq n \leq 9$ and n is a number for which the top segment should be turned off in a seven-segment display. Since the circuit will never be given an erroneous input, n will never be greater than 9, and so you should use "don't care" for $n > 9$ to reduce the size of your circuit.

4) We know the "or" of minterms in an area-four rectangle in a Karnaugh map is equivalent to the "and" of the variables (or their complements) that are unchanging in the rectangle. Demonstrate this to be true by picking any area-four rectangle in any 3- or 4-variable Karnaugh map and giving a derivation from the "or" of the rectangle's minterms to the simplified expression using the axioms of boolean algebra.

5) A set of logical operations is "functionally complete" if it can be used to re-express every logical statement. For example $\{\wedge, \sim\}$ is functionally complete because any logic statement can be rewritten into a logically equivalent form using just those two operations. To prove that a set of operations is functionally complete, you give a sequence of steps that describes how to convert a statement that is allowed to use $\{\wedge, \vee, \sim, \rightarrow, \leftrightarrow\}$ into a statement that just uses the target operations. Here are the steps to get from $\{\wedge, \vee, \sim, \rightarrow, \leftrightarrow\}$ to $\{\wedge, \sim\}$.

1. Replace any expressions $a \leftrightarrow b$ with $(a \rightarrow b) \wedge (b \rightarrow a)$.
2. Replace any expressions $a \rightarrow b$ with $\sim a \vee b$.
3. Replace any expressions $a \vee b$ with $\sim(\sim a \wedge \sim b)$.

For example, $H \rightarrow (C \vee K) = \sim H \vee (C \vee K) = \sim H \vee \sim(\sim C \wedge \sim K) = \sim(H \wedge (\sim C \wedge \sim K))$.

Nand (written in logic as $a \uparrow b$ and defined $a \uparrow b = \sim(a \wedge b)$) and Nor ($a \downarrow b = \sim(a \vee b)$) are each known to be functionally complete by themselves. Show that $\{\uparrow\}$ is functionally complete by showing $a \wedge b$ is logically equivalent to a statement that uses only $\uparrow$ operation(s) and $\sim a$ is logically equivalent to a statement that uses only $\uparrow$ operation(s). This shows that the functionally complete $\{\wedge, \sim\}$ can be replaced with $\{\uparrow\}$. Do this problem by experimentation until you find the equivalences.

Solutions

1) $f(a,b,c) = \Sigma(1,2,3,5,6,7) = 001 + 010 + 011 + 101 + 110 + 111 = a'b'c + a'bc' + a'bc + ab'c + abc' + abc$. According to the K-map below, this is equal to $b + c$.

2) $f(a,b,c,d) = \Sigma(0,1,3,9,10,11) = 0000 + 0001 + 0011 + 1001 + 1010 + 1011 = a'b'c'd' + a'b'c'd + a'b'cd + ab'c'd + ab'cd' + ab'cd$. According to the K-map below, this is equal to $b'd + ab'c + a'b'c'$. I've drawn the circuit twice, once for if only two-input And and Or gates are allowed, and once for if multi-input gates are allowed. I will specify which is allowed on the quiz.

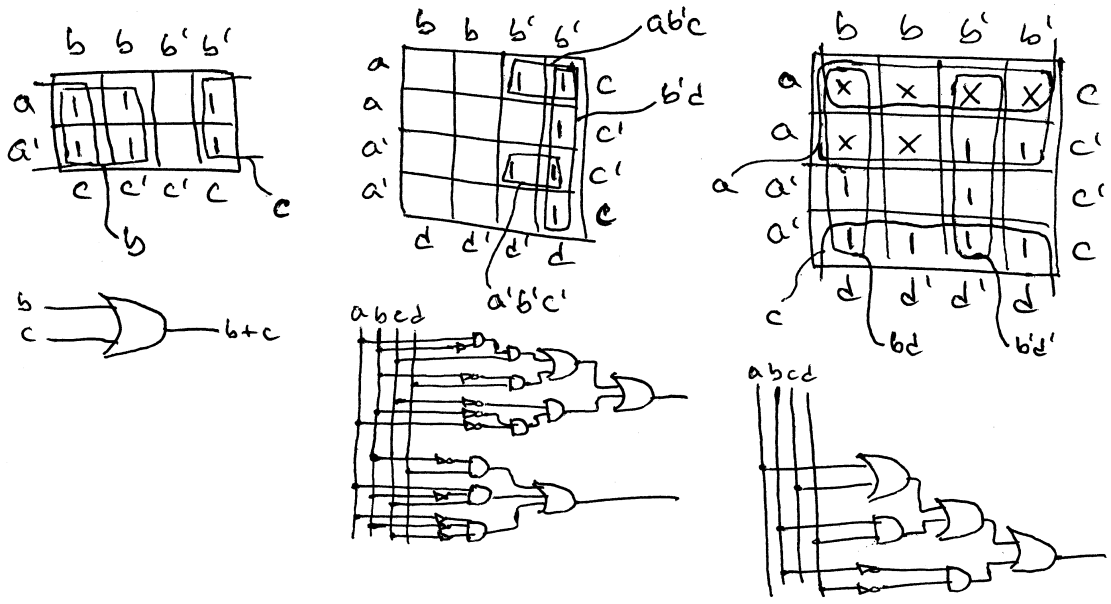
3) The top segment is on a lot. $f(a,b,c,d) = \Sigma(0,2,3,5,6,7,8,9) = 0000 + 0010 + 0011 + 0101 + 0110 + 0111 + 1000 + 1001 = a'b'c'd' + a'b'cd' + a'b'cd + a'bc'd + a'bcd' + a'bcd + ab'c'd' + ab'c'd$. In the K-map below, all don't care positions (1010, 1011, 1100, 1101, 1110, 1111) are marked X. The rectangles must cover all 1s, must not cover any blanks, and may cover any number of Xs (which allows for bigger/fewer rectangles). I found $a + c + bd + b'd'$.

4) There are many different rectangles of area four. I chose the following from a 3-variable K-map.

$abc + abc' + a'bc + a'bc' = ab(c + c') + a'b(c + c')$	Distributive
$= ab1 + a'b1$	Complement
$= ab + a'b$	Identity
$= b(a + a')$	Distributive
$= b1$	Complement
$= b$	Identity

5) Because Nand has a Not as part of its definition you can leverage that to create a Not operation: $a{\uparrow}a = \sim(a \wedge a) = \sim a$. Because Nand has an And as part of its definition you can leverage that to create an And operation: $(a{\uparrow}b){\uparrow}(a{\uparrow}b) = \sim(\sim(a \wedge b) \wedge \sim(a \wedge b)) = (a \wedge b) \vee (a \wedge b) = a \wedge b$.

Actual physical circuits are made out of just Nand or just Nor gates, so it is significant that Nand is functionally complete, and that it takes very few Nand gates to implement other logical operations: Not takes one Nand, And can be done with two, and Or can be done with three ($(a{\uparrow}a){\uparrow}(b{\uparrow}b)$).



Last updated 5pm, February 25, 2020.