ES 4 Problem Set 2 solutions

After solving the problems, look at the solutions posted on the course website and categorize your work for each problem on the following scale:

- ● Completely correct
- ◕ Nearly correct, but made a small mathematical or copying error
- ◒ Solved part of the problem correctly
- ◔ Started some work in the right direction
- ○ Incorrect, or didn't even know where to start on the problem
- Include a question mark (?) in addition to one of the above symbols if you don't feel like you understand the question or the solution well enough to make a definite judgement.

Problems with an asterisk (*) are optional.

1.a	1.b	1.c*	1.d	1.e*	2	3.a	3.b*	3.c	3.d*

What questions do you have about these concepts and skills?

What things are you uncertain about (even if you don't have a specific question)?

Approximately how long did it take you to complete the homework?

How long did you take going over the solutions and writing this reflection?

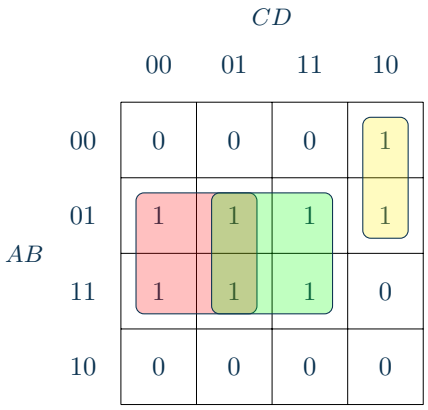
Turn in this self-assessment sheet on Gradescope. You do not need to turn in anything else, although we're happy to look at your work if you have questions!

Problem 1: Logic minimization

Minimize the truth tables and logic functions below.

(a) $\overline{A}BC + \overline{A}B\overline{C} + \overline{A}C\overline{D} + AB\overline{C} + BCD$

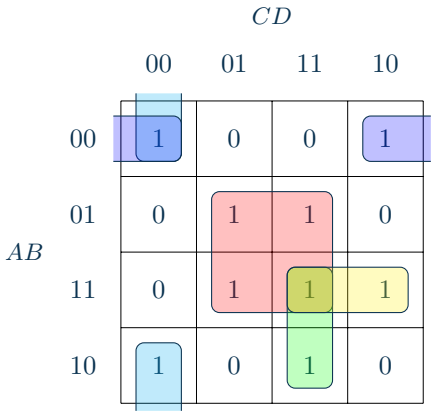
Solution:



$B\overline{C} + BD + \overline{A}C\overline{D}$

(b) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD$

Solution:



$BD + \overline{A}\,\overline{B}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D} + ABC + ACD$

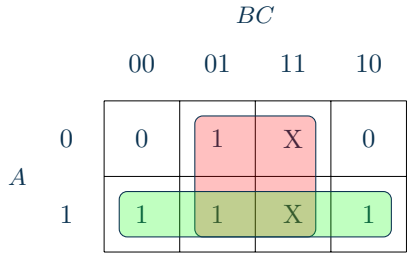
We made some progress, but the final answer is still gross. This is the sort of thing where a multiplexer or LUT really shines, because it can implement *any* 4-input truth table with the same logic. Trying to do this with individual gates would be frustrating.

(c)

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	X
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	X

Solution:

In this case, we include the X's because they allow us to draw a larger circle and eliminate several variables:

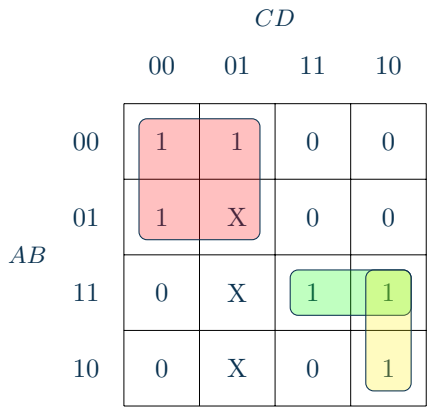


$A + C$

(d)

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	X
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	X
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	X
1	1	1	0	1
1	1	1	1	1

Solution:



$\overline{A}\,\overline{C} + ABC + AC\overline{D}$

(e)

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	X
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	X
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	X
1	1	1	0	1
1	1	1	1	0

Solution:

AB \ CD	00	01	11	10
00	1	0	X	0
01	1	1	1	0
11	0	X	0	1
10	X	1	1	0

$\overline{A}\,\overline{C}\,\overline{D} + \overline{A}BD + A\overline{B}D + ABC\overline{D}$

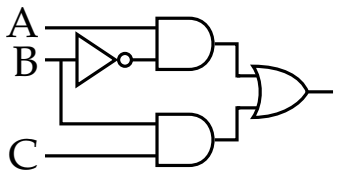
Interestingly, the X's don't allow any additional reduction of the equation!

Problem 2: Gate implementation

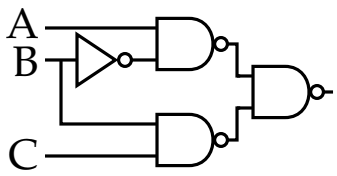
Implement the equation $Y = A\overline{B} + BC$ using only NAND gates (no inverters, just NANDs):

Solution:

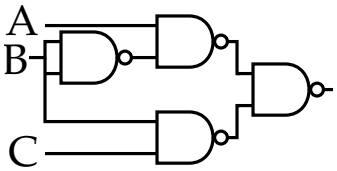
One way to accomplish this is with bubble pushing. Start by drawing the logic diagram as written:



Then we can use involution and bubble-pushing to switch the gates to NANDs:



Finally we need to generate $\overline{B}$, which we can do by using B as both inputs to a NAND gate:



Remember, the whole point of this is that inverting logic (NAND/NOR/NOT) is faster (lower propagation delay), smaller (fewer transistors) and uses less power than non-inverting logic. This actually has real utility when you're designing logic at the lowest level.

Problem 3: Multiplexers

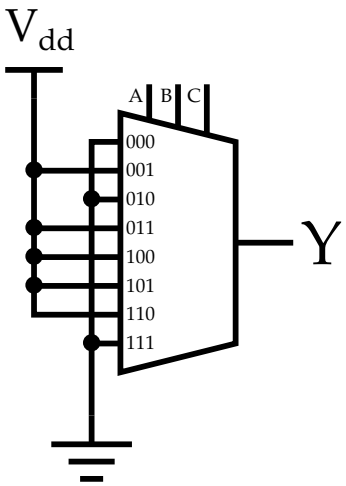
(a) Implement the equation $Y = A\overline{C} + \overline{A}C + \overline{B}C$ using an 8:1 multiplexer.

Solution:

This equation isn't in canonical form, so each term represents more than one row of the truth table (and therefore more than one input to the mux.) Start by writing out the truth table.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

From here, the multiplexer implementation is straightforward:



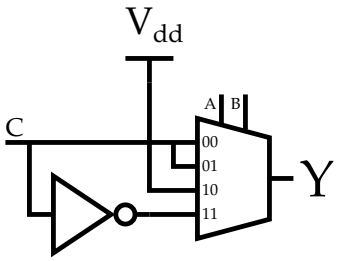
(b) Implement the equation $Y = A\overline{C} + \overline{A}C + \overline{B}C$ using a 4:1 multiplexer and an inverter.

Solution:

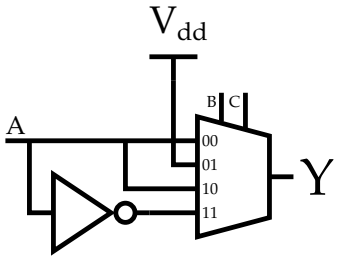
Start by writing out the truth table:

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

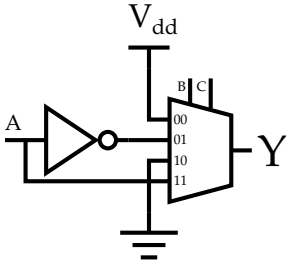
You could pick any two of the variables to be the select lines, but let's use A and B:



Here's an alternate solution where B and C are the select lines, with A as the input variable (look carefully, the differences are subtle!):



(c) Write a truth table showing what this circuit computes:



Solution:

Be careful here: since B and C are the select lines you can't just write the truth table the traditional way and group adjacent pairs of lines (which would imply that A and B are the select lines).

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

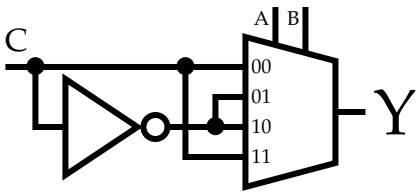
(d) Implement the equation $Y = A \oplus B \oplus C$ using a 4:1 multiplexer.

Solution:

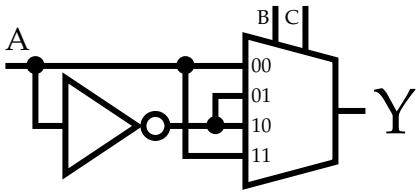
Start by writing out the truth table:

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Here's the solution if we keep A and B as the select signals, and use C as an input:



And here's a solution where B and C are the select signals:



In this case, it doesn't actually matter! This is because XOR is commutative and the equation $A \oplus B \oplus C$ could just as easily be $C \oplus B \oplus A$.

Practice Problems - For review

These are selected problems from the textbook (at the end of each chapter) which may be helpful for practice and review. The answers to these problems are online at `https://booksite.elsevier.com/9780128000564/solutions.php`.

- 2.17 (simplifying equations, drawing circuits)

- 2.27 (bubble pushing)
- 2.31 (minimizing with don't-cares)
- 2.35 (writing and minimizing equations, drawing circuits)