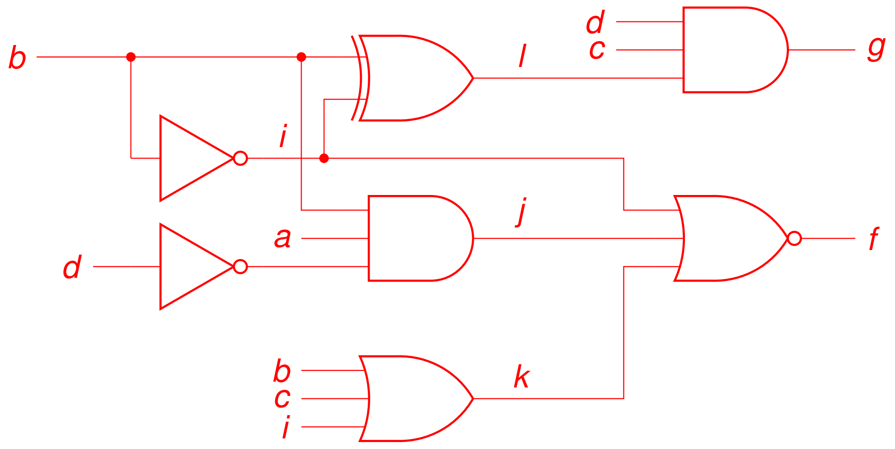
18-240: Structure and Design of Digital Systems

HW1 Solutions [12 problems, 64 points]

Covers lectures L2 – L3 *Due: 11 September 2023*

Drill Problems [32 points]

1. [3 points, Lecture 2] Draw the schematic diagram of this circuit. Do not simplify the circuit, just draw it as described below.



Note that in order to keep from cluttering the diagram, I used the same label in two places to indicate a connection rather than drawing a line between those places. This is good practice and quite commonly used. You should do the same to simplify your own schematics.

2. [6 points, Lecture 3] Write the canonical POS and SOP form for **f** and **g** from problem 1. Then, use a K-map to simplify **f** and **g**. Comment on the differences between the three methods of describing the same functionality (i.e. SystemVerilog, POS/SOP equation, Karnaugh Map).

I start by creating a Boolean equation to represent the circuit.

$$f = (b' + b \cdot a \cdot d' + (b + c + b'))'$$

And then, I simplify

$$\begin{aligned} f &= b \cdot (b' + a' + d) \cdot (b + c + b')' \\ &= (b \cdot b' + b \cdot a' + d \cdot b) \cdot (b' \cdot c' \cdot b) \\ &= (0 + a'b + bd) \cdot (0) \\ f &= 0 \end{aligned}$$

Interesting! Note that final equation is in both the POS and SOP form! The K-map will be easy!

$AB \backslash CD$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

I then followed the same process for **g**. Start with an equation and simplify.

$$\begin{aligned} g &= d \cdot c \cdot (b \oplus b') \\ &= d \cdot c \end{aligned}$$

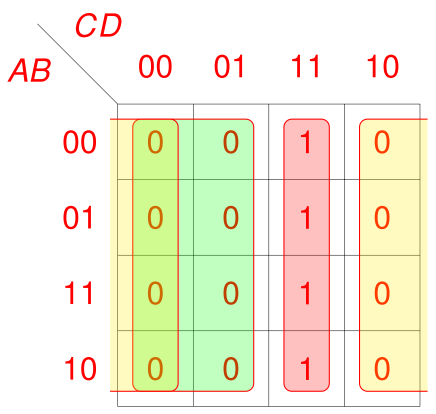
Now that I've gotten rid of anything extraneous, let's add in all the extra pieces to make minterms. To do so, I need to add all the combinations of **a** and **b**. I end up with the SOP form.

$$g = a' \cdot b' \cdot c \cdot d + a' \cdot b \cdot c \cdot d + a \cdot b' \cdot c \cdot d + a \cdot b \cdot c \cdot d$$

Converting to POS form:

$$\begin{aligned} g = {} & (a+b+c+d)(a+b+c+d')(a+b+c'+d)(a+b'+c+d)\cdots \\ & (a+b'+c+d')(a+b'+c'+d)(a'+b+c+d)(a'+b+c+d')\cdots \\ & (a'+b+c'+d)(a'+b'+c+d)(a'+b'+c+d')(a'+b'+c'+d) \end{aligned}$$

Notice that if there are 4 minterms, there must be 12 maxterms (for a 4-input equation).



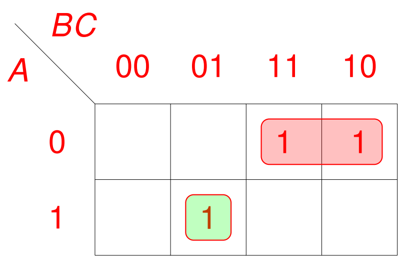
The red circles the minterms to give a simplified equation $g = cd$. Or, you could use the maxterms (the yellow and green circles) to come up with $g = (c) \cdot (d)$ – same thing.

Now, there are at least three ways of expressing `f` and `g` in this problem. The SystemVerilog describes a physical implementation of the gates involved in building the circuit. The POS and SOP equations describe the same functionality, but do so in a canonical (i.e. standardized) form. The K-map is a tool, but it describes the same functionality in relation to other portions of the functionality (i.e. what terms are adjacent to each other) in such a way as to facilitate simplification.

3. [3 points, Lecture 3] For the truth table shown, draw the K-map for functions F, G and H. Use the K-maps to write the simplified Boolean expression for each function.

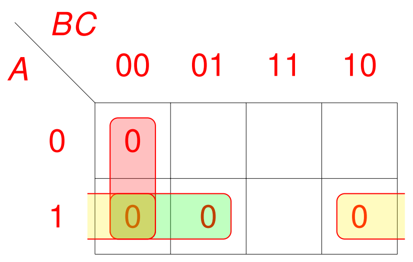
A	B	C	F	G	H
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	0

For F:



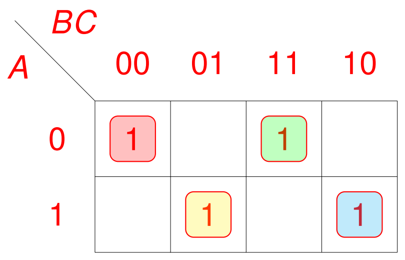
$F(A,B,C) = A'B + AB'C$

For G, let's do a POS-style solution:



$G(A,B,C) = (B+C)\cdot(A'+B)\cdot(A'+C)$

For H:



Yuck! $H(A,B,C) = A'B'C' + A'BC + AB'C + ABC'$

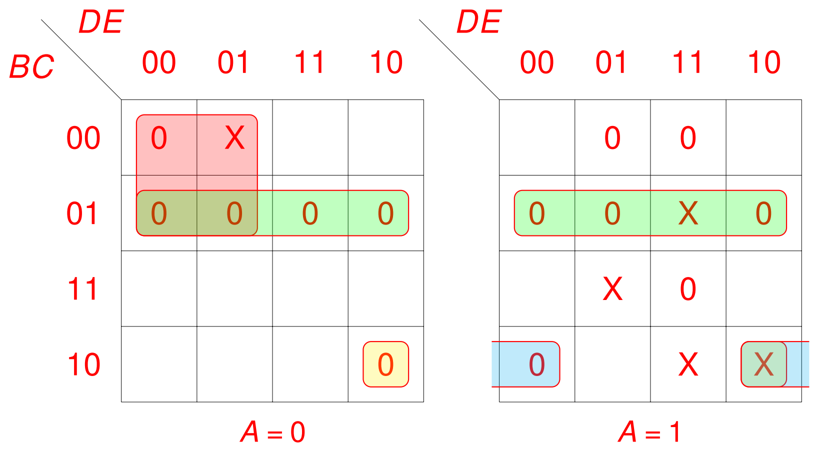
4. [3 points, Lecture 2] Write a SystemVerilog module, using structural style, to model functions F, G and H from the previous problem.

   `hw1prob4.sv` is posted on Canvas.

5. [4 points, Lecture 3] Draw the K-map for the function:

$$F(A,B,C,D,E) = \Pi M(0,4,5,6,7,10,17,19,20,21,22,24,31) \cdot D(1,23,26,27,29)$$

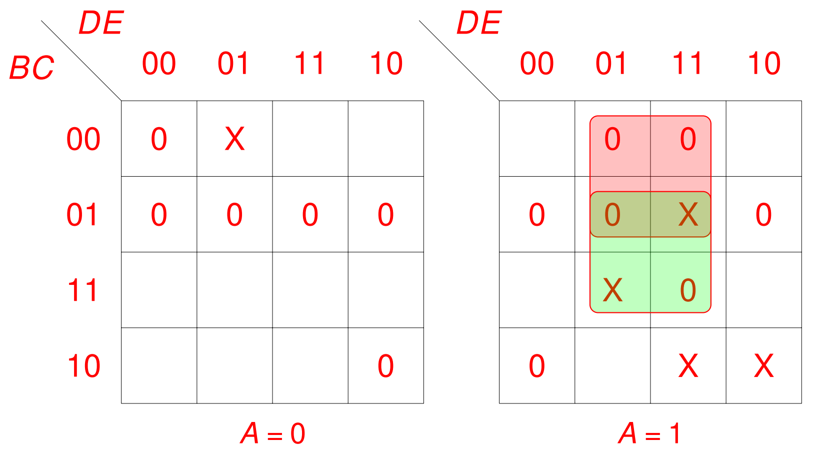
Use the K-map to write the minimized POS Boolean expression.



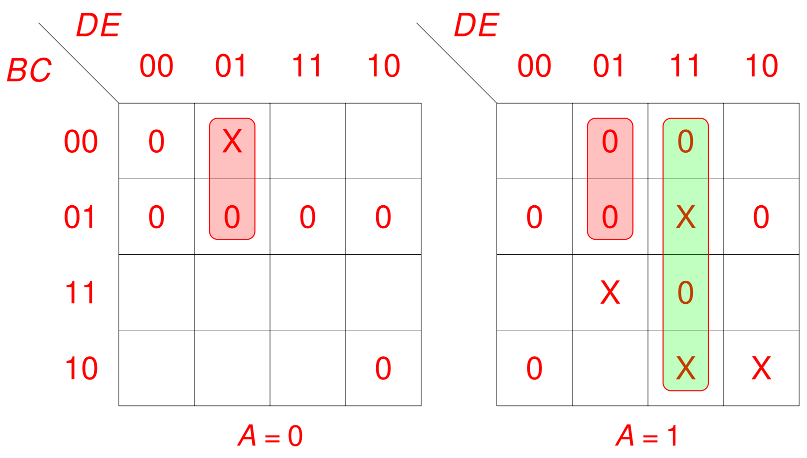
These four terms lead to a partial equation:

$$F(A,B,C,D,E) = (A+B+D)\cdot(B+C')\cdot(B'+C+D'+E)\cdot(A'+B'+C+E)$$

The remaining zeros can be covered in two different, equally efficient, ways.



which would add the terms $(A'+B+E')\cdot(A'+C'+E')$ to the equation. Alternatively,



BC \ DE	00	01	11	10
00	0	X		
01	0	0	0	0
11				
10				0

$A = 0$

BC \ DE	00	01	11	10
00		0	0	
01	0	0	X	0
11		X	0	
10	0		X	X

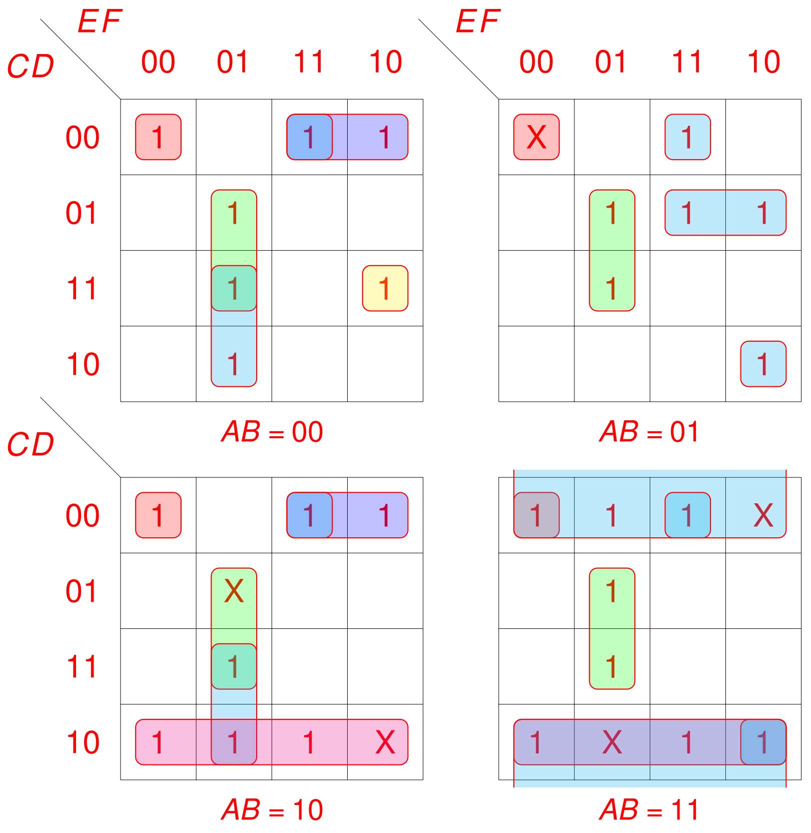
$A = 1$

add $(B + D + E') \cdot (A' + D' + E')$.

6. [6 points, Lecture 3] Draw the K-map for the function:

$$F(A,B,C,D,E,F) = \Sigma m(0,2,3,5,9,13,14,19,21,22,23,26,\\ 29,32,34,35,40,41,43,45,48,49,51,53,56,\\ 58,59,61) + d(16,37,42,50,57)$$

Use the K-map to write the simplified SOP Boolean expression.



The resulting equation is

$F(A,B,C,D,E,F) =$	$C'D'E'F'$	Red, upper left corner of all four sub-maps
$+$	$DE'F$	Green, vertical block of two on all four sub-maps
$+$	$A'B'CDEF'$	Yellow, singleton on upper left sub-map
$+$	$B'CE'F$	Blue, vertical block of two, appears twice
$+$	$B'C'D'E$	Purple, horizontal block of two, appears twice
$+$	ACD'	Pink, bottom row of bottom two sub-maps
$+$	$BCD'EF'$	Blue, single in bottom-left of two right sub-maps
$+$	ABD'	Blue, top and bottom row of AB=11 sub-map
$+$	$A'BC'DE$	Blue, horiz. block of 2 in upper right sub-map
$+$	$C'D'EF$	Blue, single block in upper-right row of all four

7. [2 points, Lecture 3] For the function F in problem 6, how many gates are needed to implement the minterm canonical form (don't count inverters)? How many to implement the reduced form? How many literals are there in each of the implementations?

   To implement the minterm canonical form, one AND gate is necessary for each minterm, plus a single OR gate to join them together. For F, that's 28 AND gates and one OR gate, for a total of 29 gates. Each minterm has all six of the input variables, so that's $28 * 6 = 168$ literals.

   The reduced form is much better! There are 10 terms, so 10 AND gates and one OR gate for a total of 11. Counting the literals in the above equation, I see 41 literals.

   I *like* minimized equations!

8. [5 points, Lecture 2] Write an **`initial`** block that will generate the waveform below. Do not include a **`$monitor`** statement. Put this **`initial`** block in a module (with outputs **`a`** and **`b`**) named **`hw1prob8`**.

   `hw1prob8.sv` is posted to Canvas.

Non-Drill Problems [32 points]

9. [12 points, Lecture 3] **Good times at the Hamster Training Academy**

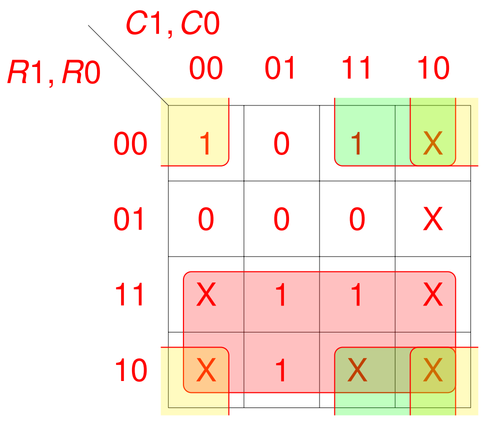
My first step was to create the truth table, to ensure I had thought about every situation. For instance, what should I do about the "missing" sections of the map? Those will be *don't cares* because they should never be used as an input.

R1	R0	C1	C0	A	B	Comment
0	0	0	0	1	0	West (exit)
0	0	0	1	0	0	North (exit)
0	0	1	0	X	X	missing position
0	0	1	1	1	1	South (to 0111)
0	1	0	0	0	0	North (to 0000)
0	1	0	1	0	0	0001 is closer to exit
0	1	1	0	X	X	missing position
0	1	1	1	0	1	East (exit)
1	0	0	0	X	X	missing position
1	0	0	1	1	X	West or South so B is X
1	0	1	0	X	X	missing position
1	0	1	1	X	X	missing position
1	1	0	0	X	X	missing position
1	1	0	1	1	1	1001 is closer to exit
1	1	1	0	X	X	missing position
1	1	1	1	1	1	South (exit)

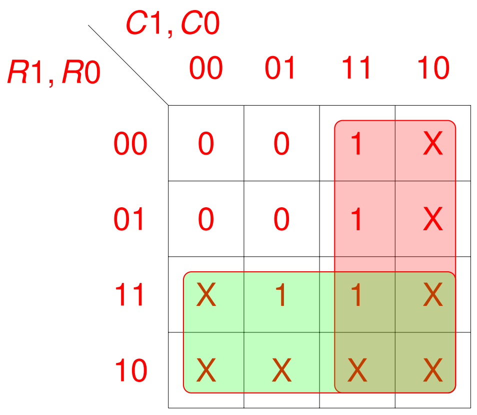
From the truth table, the canonical forms are easy:

$$A(R1, R0, C1, C0) = \Sigma m(0, 3, 9, 13, 15) + d(2, 6, 8, 10, 11, 12, 14)$$
$$B(R1, R0, C1, C0) = \Sigma m(3, 7, 13, 15) + d(2, 6, 8, 9, 10, 11, 12, 14)$$

Putting that in a K-map:



Results in $A = R1 + \overline{R0}C1 + \overline{R0} \cdot \overline{C0}$



$R1, R0$ \ $C1, C0$	00	01	11	10
00	0	0	1	X
01	0	0	1	X
11	X	1	1	X
10	X	X	X	X

Nice and easy: $B = R1 + C1$

10. [2 points, Lecture 3] What is the minimized equation for the function described in this K-map?

The checkerboard pattern gives it away as the XNOR of all four inputs.

$F(A, B, C, D) = \overline{A \oplus B \oplus C \oplus D}$

11. [6 points, Lecture 2] In HW0 you proved that AND does distribute over XOR (i.e. $A \cdot (B \oplus C) = (AB) \oplus (AC)$) . Write a SystemVerilog testbench to show this fact. Your testbench should iterate over all possible values for **`A`**, **`B`** and **`C`**, with a **`$monitor`** statement showing **`A`**, **`B`** and **`C`** as well as both sides of the expression. Further, your testbench should use an if statement to test whether the left side actually does equal the right side and **`$display`**ing an error in the case where they aren't actually equal (yes, I realize this statement won't ever get printed, as you have proved the fact). You should, when developing your testbench, purposely insert an error in the equations to ensure the **`$display`**ed statement would work if there was an error.

`hw1prob11.sv` is posted to Canvas.

12. [12 points, Lecture 2] Given the following logic diagram and initial block that will drive its inputs, list every event on the simulator's "next event" list (i.e. the to-do list) from the beginning of time until the end of the simulation using a table such as below. Each simulation cycle should have a separate row. Refer to the last few slides of Lecture 02 for guidance about what the simulator is doing.

Time	**Next Event List**	**Values of Variables**							
		A	*B*	$\overline{A}$	$\overline{B}$	*f*1	*f*2	*f*3	*F*
start	initial@0	X	X	X	X	X	X	X	X
0	anot=0@1; bnot=1@1; f2=1@4; initial@13	1	0	X	X	X	X	X	X
1	f2=1@4; f1=0@4; f3=0@6; initial@13	1	0	0	1	X	X	X	X
4	f3=0@6; f=0@10; initial@13	1	0	0	1	0	1	X	X
6	f=0@10; initial@13	1	0	0	1	0	1	0	X
10	initial@13	1	0	0	1	0	1	0	0
13	anot=1@14; f3=1@18; initial@26	0	0	0	1	0	1	0	0
14	f1=1@17; f3=1@18; initial@26	0	0	1	1	0	1	0	0
17	f3=1@18; initial@26	0	0	1	1	1	1	0	0
18	f=1@24; initial@26	0	0	1	1	1	1	1	0
24	initial@26	0	0	1	1	1	1	1	1
26	bnot=0@27; f2=0@30; initial@39	0	1	1	1	1	1	1	1
27	f2=0@30; f3=0@32; initial@39	0	1	1	0	1	1	1	1
30	f3=0@32; f=0@36; initial@39	0	1	1	0	1	0	1	1
32	f=0@36; initial@39	0	1	1	0	1	0	0	1
36	initial@39	0	1	1	0	1	0	0	0
39	Simulation is done!	0	1	1	0	1	0	0	0