EE M16: Logic Design of Digital Systems **Fall 2016**

Midterm Solutions

Instructor: Prof. Danijela Cabric *Scribes: Ghaith Hattab, Mihir Laghate*

1. Reduce the following expression using Boolean algebra postulates and theorems. The simplified expression should have the minimum number of gates. Show the intermediate steps.

$$f(a,b,c,d) = \overline{\overline{(acd)}(\bar{a}+\bar{b}+\bar{d})(\overline{ad}+c)} + \overline{a\bar{b}}(\bar{a}+\bar{b}c+\bar{b}\bar{c})$$

Solution

$$\begin{aligned}
f(a,b,c,d) &= acd + \overline{(\overline{a}+\overline{b}+\overline{d})} + \overline{(\overline{ad}+c)} + (\overline{a}+b)(\overline{a}+\overline{b}c+\overline{b}\cdot\overline{c}) && \because \text{DeMorgan's Law} \\
&= acd + abd + ad\overline{c} + (\overline{a}+b)(\overline{a}+\overline{b}) && \because \text{DeMorgan's Law} \\
&= ad(c+b+\overline{c}) + \overline{a} + \overline{a}\cdot\overline{b} + \overline{a}b + b\overline{b} && \because \text{Distributive Law} \\
&= ad + \overline{a}(1+\overline{b}+b) + 0 && \because x+\overline{x}=1, 1+x=1, x\overline{x}=0 \\
&= ad + \overline{a} && \because x+\overline{x}=1, 1+x=1, 0+x=x \\
&= \overline{a} + d
\end{aligned}$$

2. Consider the following function

$$f(a,b,c,d) = \Sigma m(1,7,9,11,13,15).$$

1. Use K-maps to minimize **both** the sum of products and products of sums forms. Write the Boolean expressions.

2. Implement the function using the minimal number of gates. You can use either NOR gates or NAND gates with maximum number of four inputs per gate.

Solution

1.

For the Sum of Products:

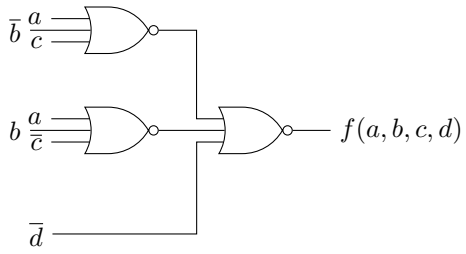
ab \ cd	00	01	11	10
00	0	1	0	0
01	0	0	1	0
11	0	1	1	0
10	0	1	1	0

For the Product of Sums

ab \ cd	00	01	11	10
00	0	1	0	0
01	0	0	1	0
11	0	1	1	0
10	0	1	1	0

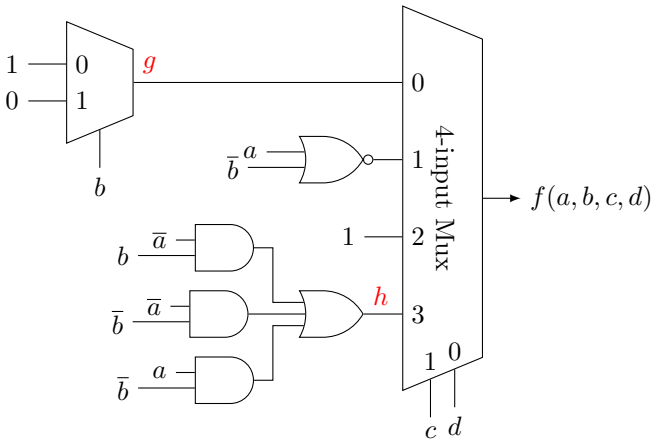
$f(a,b,c,d) = ad + bcd + \overline{b} \cdot \overline{c}d$ $\qquad$ $f(a,b,c,d) = d(a+\overline{b}+c)(a+b+\overline{c})$

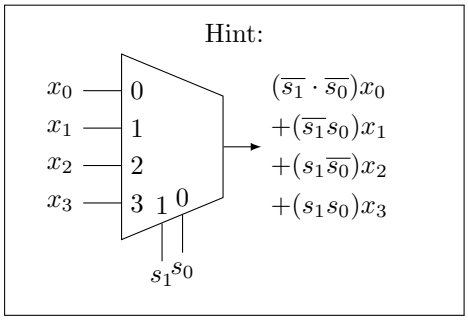
2. The Product of Sums requires 3 gates while the sum of products requires 4. In particular, note that the product of sums has a term d which does not require a gate to implement.



3. Consider the following function shown below.

1. Write the switching expression for the multiplexer circuit below. The multiplexers have binary select inputs. The expression need not be simplified.
2. Determine the prime implicants and the essential prime implicants for this expression.
3. Find the minimal sum-of-product for the switching expression f.
4. Does this function have a unique minimal sum-of-products? If not, list any other minimal sum-of-products expressions.





Solution

Note the new variables g and h defined in the circuit diagram above. From the 2-input multiplexer, we get

$$g = \overline{b} \cdot 1 + b \cdot 0 = \overline{b}.$$

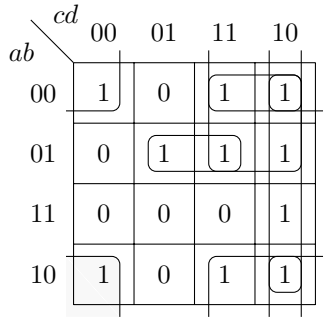
Simplifying the AND-OR network, we get

$$\begin{aligned} h &= \overline{a}b + \overline{a} \cdot \overline{b} + a\overline{b} \\ &= \overline{a}b + \overline{b} \\ &= \overline{a} + \overline{b} = \overline{ab} \end{aligned}$$

Using the hint, we can write the output of the 4-input multiplexer as:

$$f(a,b,c,d) = \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{a}b\overline{c}d + c\overline{d} + (\overline{a} + \overline{b})cd$$

Now, we can use f to draw the Karnaugh map and list the prime implicants:



Prime Implicants:

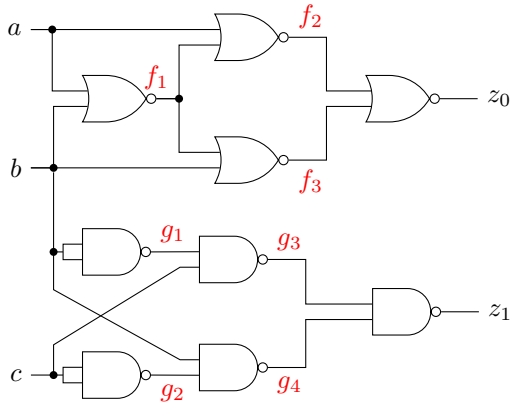
X0X0	Essential
0X1X	Not Essential
X01X	Essential
XX10	Essential
01X1	Essential

The minimal sum of products is given by the essential prime implicants alone:

$$f(a,b,c,d) = \overline{b} \cdot \overline{d} + \overline{b}c + c\overline{d} + \overline{a}bd.$$

Since the essential prime implicants cover all the minterms, the minimal sum of products expression is unique.

4. Analyze the NAND-NOR network shown in the figure below. Obtain switching expressions for the outputs z_0 and z_1.



Solution

Find z_0	Find z_1
$f_1 = \overline{a+b}$	$g_1 = \overline{bb}$
$\quad = \bar{a}\bar{b}$	$\quad = \bar{b}$
$f_2 = \bar{a}\bar{f}_1$	$g_2 = \bar{c}$
$\quad = \bar{a}\overline{\bar{a}\bar{b}}$	$g_3 = \overline{g_1 c}$
$\quad = \bar{a}(a+b)$	$\quad = \bar{g}_1 + \bar{c}$
$\quad = \bar{a}b$	$\quad = b + \bar{c}$
$f_3 = a\bar{b}$	$g_4 = \overline{g_2 b}$
$z_0 = \bar{f}_2\bar{f}_3$	$\quad = \bar{b} + c$
$\quad = \overline{\bar{a}b}\ \overline{a\bar{b}}$	$z_1 = \bar{g}_3 + \bar{g}_4$
$\quad = (a+\bar{b})(\bar{a}+b)$	$\quad = \bar{b}c + b\bar{c}$
$\quad = ab + \bar{a}\bar{b}$	$\quad = b \oplus c$
$\quad = \overline{a \oplus b}$	

5. Design a combinational circuit that converts a 3-bit sign-and-magnitude number, a, into a 3-bit one's complement number, b. You are allowed to use any combination of the following blocks:

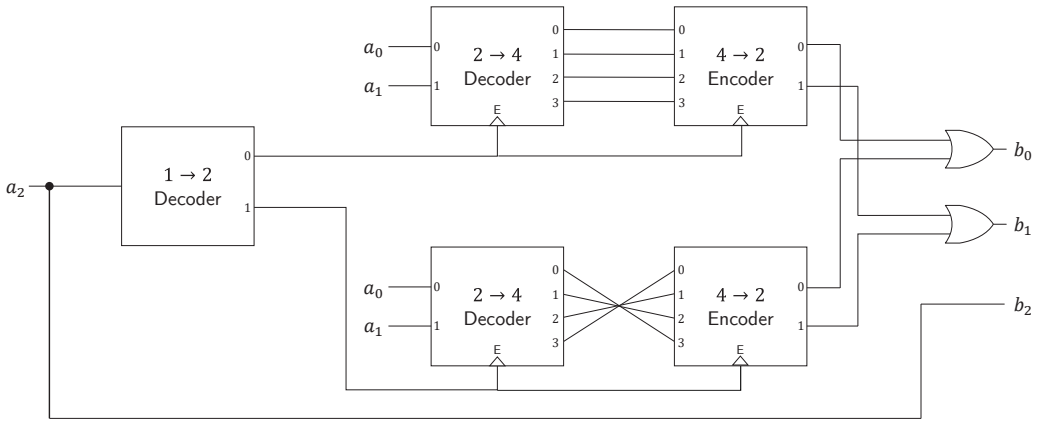
- Decoders: $1 \to 2$ and $2 \to 4$ decoders
- Encoders: $2 \to 1$ and $4 \to 2$ encoders
- Logic gates: Use either OR gates or AND gates, **but not both**

Every block or wire must be clearly labeled.

Decimal	Sign-Magnitude	One's Complement
-3	111	100
-2	110	101
-1	101	110
-0	100	111
0	000	000
1	001	001
2	010	010
3	011	011

Solution

Observe that if the input is positive, then b is the same as a. If the input is negative, then the bits are flipped. This flipping operation can be easily done by reordering the wire connections between a decoder and an encoder. The circuit is given below.



Note that we can remove the $1 \to 2$ decoder and directly use a and $\bar{a}$ as inputs to the enable ports of the $2 \to 4$ decoders.

6. Compute $z = (a - b) + (c - d)$ in 2's complement.

1. Fill up the table given below.
2. How many bits should z have to represent the correct result?
3. Perform calculations on bit-vectors representing a, b, c and d (all in 2's complement) and show every step of your work. z should be given in **both** decimal representation and 2's complement. **If you want**, you can extend the table below.

Solution

(a)

	Decimal	Sign-Magnitude	2's Complement
a	-13	1 1101	1 0011
b	-10	1 1010	1 0110
c	-6	1110	1010
d	82	0101 0010	0101 0010

(b)

z is 8 bits.

(c)

				a	$-$	b			
-13		1	1	1	1	0	0	1	1
10	+	0	0	0	0	1	0	1	0
-3		1	1	1	1	1	1	0	1

				c	$-$	d			
-6		1	1	1	1	1	0	1	0
-82	+	1	0	1	0	1	1	1	0
-88		1	0	1	0	1	0	0	0

	z							
-3	1	1	1	1	1	1	0	1
-88	1	0	1	0	1	0	0	0
-91	1	0	1	0	0	1	0	1

$z = (-91)_{10} = (10100101)_2$

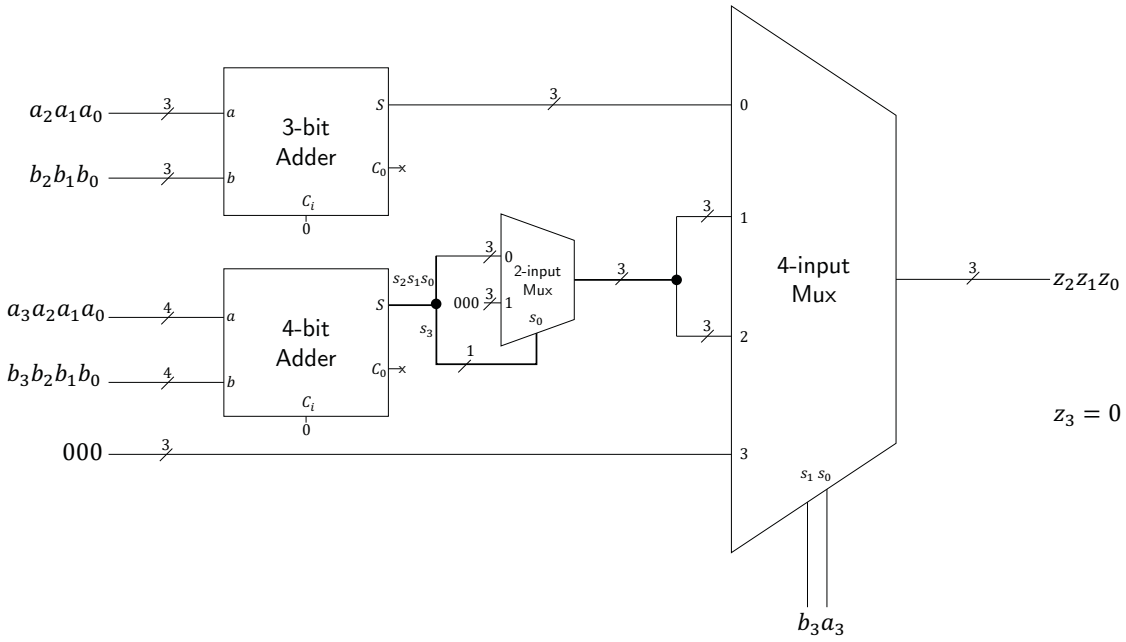
7. Design a combinational network that has 4-bit inputs a and b, and a four-bit output z. All inputs and outputs are given in two's complement representation. The function of the system is

$$z = \max\{a + b, 0\} \mod 8$$

For example, if $a = 3$ and $b = 7$, then $z = 2$. If $a = 3$ and $b = -7$, then $z = 0$.
You are allowed to use only magnitude comparators, adders, and/or multiplexers. Every block and wire must be clearly labeled.

Solution

Observe that z is always a non-negative number. Thus, $z_3 = 0$. To find $z_2z_1z_0$, we implement the circuit below.



The logic behind this circuit is as follows:

- If both numbers are negative, i.e., $a_3 = b_3 = 1$, then z is zero.
- If both numbers are positive, i.e., $a_3 = b_3 = 0$, then we may have an overflow, e.g., adding four-bits of $a = 3$ to four-bits of $b = 7$ will give 1010 which is -6 in 2's complement, **if we take the four-bits as an output**. To fix this, we only need to look at the last three bits of the sum term, or use a three-bit adder that adds $a_2a_1a_0$ to $b_2b_1b_0$.

- If one number is negative and the other is positive, i.e., $a_3 = \bar{b}_3$, then we will not have the overflow problem, and the output of a four-bit adder, $S = s_3 s_2 s_1 s_0$, is always correct. In that case, if $s_3 = 1$, then the sum is negative, and thus we need to select a three-bit zero signal as the output. If $s_3 = 0$, then the number is positive, and thus we select $s_3 s_2 s_1$.

In other words, using the sign bits and the multiplexer, we know when the output of an adder is correct and when the output of that adder needs to be corrected due to the overflow problem. Specifically, we have an overflow when the signs are the same. If both are negative, we do not need an adder in the first place since $z = 0$. If both are positive, then we have two options. The first one is to use a three-bit adder, where we drop the most significant bits of a and b and make the output of that adder equals to $z_2 z_1 z_0$. The second option is to use a four-bit adder, but only take three bits from the output of the adder (by dropping the most significant bit). If the signs are different, we won't have an overflow, so we can directly use a four-bit adder. In this case, the output of the adder will determine whether the number is positive or negative (by looking at the most significant bit of the output).