# [CS M51A F18] Solution to Practice Homework 6

Solutions posted: 12/6/18

---

## Problem 1

We would like to put together a 12-input decoder using 4-input decoders. Let us consider the following two methods.

1. Using a tree structure, how many levels would we need? How many 4-input decoders are used in total?

   ***Solution*** At each level, we can deal with 4 inputs since our building block is 4-input decoders. Since we need to deal with 12 inputs total, this leads to $\frac{12}{4} = 3$ levels.

   The total number of 4-input decoders is:

   $$1 + 2^4 + (2^4)^2 = 273$$

2. The textbook has an example of using a coincident structure which divides the $n$-inputs into two groups using $\frac{n}{2}$-input decoders. This can be expanded to the case where we divide the $n$ inputs into $k$ groups, using $r$-input decoders. In this case, what would be the value of $k$ in terms of $n$ and $r$? How many AND gates would we need?

   ***Solution*** The textbook example divides the inputs into 2 groups and combines 2 decoder outputs, one from each, into one 2-input AND gate. Therefore, it is possible to divide the inputs into $k$ groups and combine $k$ decoder inputs into one $k$-input AND gate each. Each divided group now needs to accomodate for $\frac{n}{k}$ inputs, which can be done with $r = \frac{n}{k}$-input decoders.

   As $r = \frac{n}{k}$, $k = \frac{n}{r}$, and AND gates are needed for every combination of $r$-input decoder outputs, which equals $(2^r)^k = 2^{rk} = 2^n$.

   Another way of calculating the number of AND gates would be to consider what the output of the AND gates correspond to. Since each output of the AND gate is an output for the overall $n$-input decoder, the number of AND gates must be equal to the number of outputs, which is $2^n$.

3. Now, for our coincident 12-input decoder, how many 4-input decoders do we need? How many AND gates? How many inputs do we need for the AND gates?

   ***Solution*** We plug in $n = 12$, $r = 4$ and $k = 3$ in the equations we derived in part 2. We need 3 4-input decoders, $2^{12} = 4096$ AND gates, where each AND gate has 3 inputs.

## Problem 2

We want to use multiplexers to implement the function $f(a, b, c, d) = \sum m(1, 2, 4, 7, 10, 11, 14)$ using the following methods.

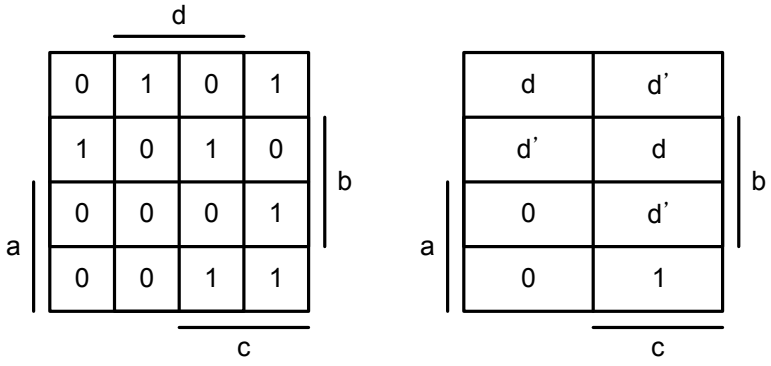
For both cases, show your work, either the minterm expressions or K-map implementations. Assume that complements to inputs are available.

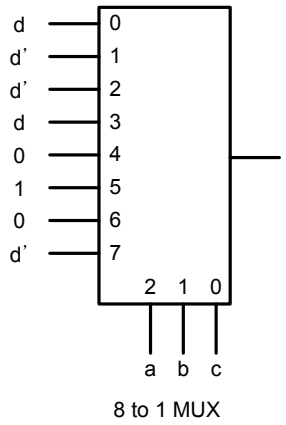
1. Implement the function using 8-to-1 multiplexers. Use $a, b, c$ as the selection bits.

***Solution***

$$\begin{aligned} f(a,b,c,d) &= \sum(1,2,4,7,10,11,14) \\ &= a'b'c'd + a'b'cd' + a'bc'd' + a'bcd + ab'cd' + ab'cd + abcd' \\ &= m_0(a,b,c)d + m_1(a,b,c)d' + m_2(a,b,c)d' + m_3(a,b,c)d + m_5(a,b,c)(d'+d) + m_7(a,b,c)d' \end{aligned}$$

Or if we use K-maps, we get:



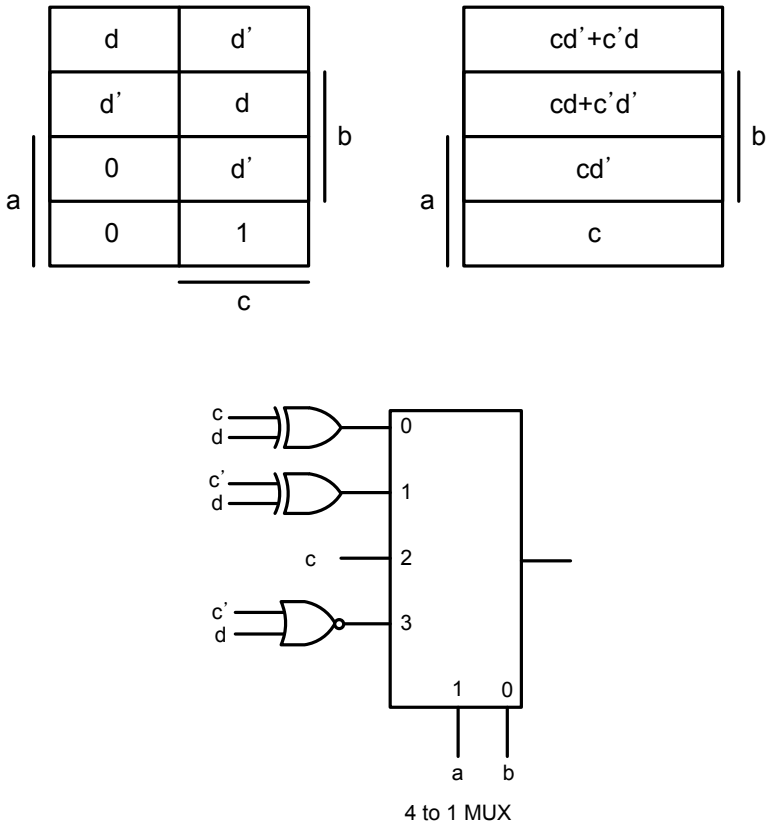




2. Repeat the same process using 4-to-1 multiplexers. Use $a, b$ as the selection bits, and for the additional logic needed, only use XOR gates and NOR gates.

***Solution***

$$\begin{aligned} f(a,b,c,d) &= \sum(1,2,4,7,10,11,14) \\ &= a'b'c'd + a'b'cd' + a'bc'd' + a'bcd + ab'cd' + ab'cd + abcd' \\ &= m_0(a,b)(c'd + cd') + m_1(a,b)(c'd' + cd) + m_2(a,b)(cd' + cd) + m_3(a,b)cd' \\ &= m_0(a,b)(c \oplus d) + m_1(a,b)(c' \oplus d) + m_2(a,b)c + m_3(a,b)(c'+d)' \end{aligned}$$

With K-maps, we get:




Note that for input at 1, the input can also be $c \oplus d'$.

## Problem 3

Complete the following table for the given representations in each part.

| | Signed integer $x$ (in decimal) | Representation $x_R$ (in decimal) | Bit-vector $\underline{x}$ |
|---|---|---|---|
| a | -25 | | |
| b | | 37 | |
| c | | | 110110 |

The 'Signed integer' column shows the actual value of the signed integer, and the 'Representation' column holds the value of the bit-vector representation. Assume that any bit vector that is shorter than the given bit length has leading 0s.

1. Two's complement, $n = 7$ bits

   ***Solution***

   For a: Since $25 = 16 + 8 + 1 = 11001_{(2)}$ and $n = 7$, we can get

   $$\begin{aligned} \underline{x} &= x' + 1 = 1100110 + 1 = 1100111_{(2)} \\ x_R &= 1100111_{(2)} = 64 + 32 + 4 + 2 + 1 = 103 \end{aligned}$$

   For b: $\underline{x} = 37 = 32 + 4 + 1 = 100101_{(2)}$ and $x$ is also the same value.

   For c: $x = x_R = 0110110_{(2)} = 54$

| | Signed integer $x$ (in decimal) | Representation $x_R$ (in decimal) | Bit-vector $\underline{x}$ |
|---|---|---|---|
| a | -25 | 103 | 1100111 |
| b | 37 | 37 | 0100101 |
| c | 54 | 54 | 110110 |

2. Two's complement, $n = 6$ bits

***Solution***

For a: Since $25 = 16 + 8 + 1 = 11001_{(2)}$ and $n = 6$, we can get

$$\begin{aligned} \underline{x} &= x' + 1 = 100110 + 1 = 100111_{(2)} \\ x_R &= 100111_{(2)} = 32 + 4 + 2 + 1 = 39 \end{aligned}$$

For b:

$$\begin{aligned} \underline{x} &= 37 = 32 + 4 + 1 = 100101_{(2)} \\ x &= -32 + 4 + 1 = -27 \end{aligned}$$

For c:

$$\begin{aligned} x &= -32 + 16 + 4 + 2 = -10 \\ x_R &= 110110_{(2)} = 32 + 16 + 4 + 2 = 54 \end{aligned}$$

| | Signed integer $x$ (in decimal) | Representation $x_R$ (in decimal) | Bit-vector $\underline{x}$ |
|---|---|---|---|
| a | -25 | 39 | 100111 |
| b | -27 | 37 | 100101 |
| c | -10 | 54 | 110110 |

3. Ones' complement, $n = 6$ bits

***Solution***

For a: Since $25 = 16 + 8 + 1 = 11001_{(2)}$ and $n = 6$, we can get

$$\begin{aligned} \underline{x} &= x' + 1 = 100110 = 100110_{(2)} \\ x_R &= 100110_{(2)} = 32 + 4 + 2 = 38 \end{aligned}$$

For b:

$$\begin{aligned} \underline{x} &= 37 = 32 + 4 + 1 = 100101_{(2)} \\ x &= (-32 + 4 + 1) + 1 = -26 \end{aligned}$$

For c:

$$\begin{aligned} x_R &= 110110_{(2)} = 32 + 16 + 4 + 2 = 54 \\ x &= (-32 + 16 + 4 + 2) + 1 = -9 \end{aligned}$$

| | Signed integer $x$ (in decimal) | Representation $x_R$ (in decimal) | Bit-vector $\underline{x}$ |
|---|---|---|---|
| a | -25 | 38 | 100110 |
| b | -26 | 37 | 100101 |
| c | -9 | 54 | 110110 |

## Problem 4

We would like to identify the working signals of an arithmetic unit at work, namely the two's-complement arithmetic unit shown in the textbook at Figure 10.12 (page 297). For the system, fill in the table below according to the given computation in each part. $c_0$, $K_x$ and $K_y$ are control signals, and $z$, $c_{out}$, $ovf$, $zero$ and $sgn$ are result signals.

1. $z = x + y$

   ***Solution*** For addition, $c_0 = K_x = K_y = 0$.

| $x$ | $y$ | $c_0$ | $K_x$ | $K_y$ | $z$ | $c_{out}$ | $ovf$ | $zero$ | $sgn$ |
|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 00000000 | 0 | 0 | 0 | 00000000 | 0 | 0 | 1 | 0 |
| 10101010 | 10100101 | 0 | 0 | 0 | 01001111 | 1 | 1 | 0 | 0 |
| 10110110 | 00110011 | 0 | 0 | 0 | 11101001 | 0 | 0 | 0 | 1 |

2. $z = x - y$

   ***Solution*** For subtraction of two's complement numbers, $c_0 = 1$, $K_x = 0$, $K_y = 1$.

| $x$ | $y$ | $c_0$ | $K_x$ | $K_y$ | $z$ | $c_{out}$ | $ovf$ | $zero$ | $sgn$ |
|---|---|---|---|---|---|---|---|---|---|
| 00100010 | 00100010 | 1 | 0 | 1 | 00000000 | 1 | 0 | 1 | 0 |
| 10010010 | 01000011 | 1 | 0 | 1 | 01001111 | 1 | 1 | 0 | 0 |
| 00010011 | 00110011 | 1 | 0 | 1 | 11100000 | 0 | 0 | 0 | 1 |

## Problem 5

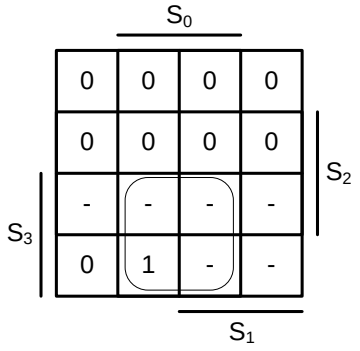
Using a modulo-16 binary counter with parallel inputs explained in the textbook (p. 321) along with logic gates, we would like to implement the following counters.

For each design, minimize the gate logic for the input signals and load signal as much as possible using K-maps.

1. A modulo-10 counter.

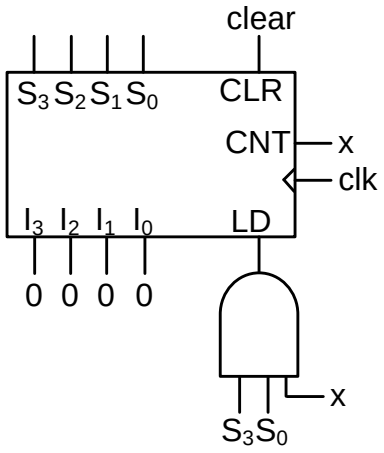
   ***Solution*** For a modulo-10 counter, we need to make a jump from 9 to 0, and the values from 10 to 15 are don't-cares as they are never reached.

   The K-map for the LD signal is:




   where we can get $LD = xS_3S_0$.

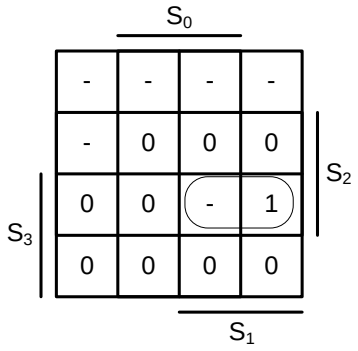
   The counter implementation is:




2. A 5-to-14 counter.

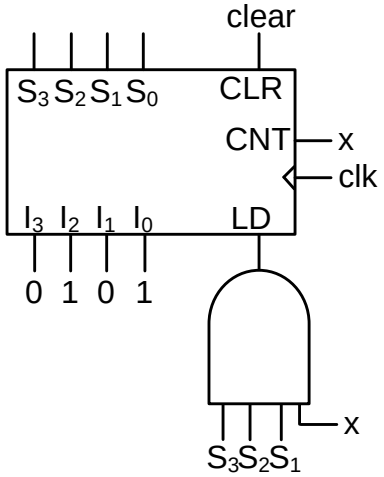
   ***Solution*** A 5-to-14 counter starts from 5 and jumps from 14 to 5. Values from 0 to 4 and 15 are don't-cares as they are never reached.

   The K-map for the LD signal is:




   where we can get $LD = xS_3S_2S_1$.

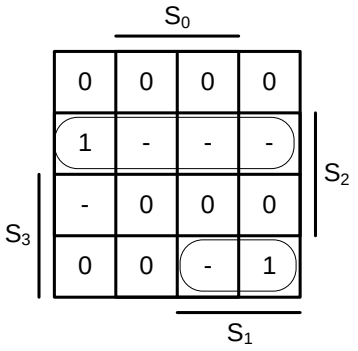
   The counter implementation is:




3. A counter which counts in the following sequence: 0, 1, 2, 3, 4, 8, 9, 10, 13, 14, 15

***Solution*** This counter makes two jumps, from 4 to 8, and 10 to 13. Values 5, 6, 7, 11, and 12 are don't-cares as they are never reached.

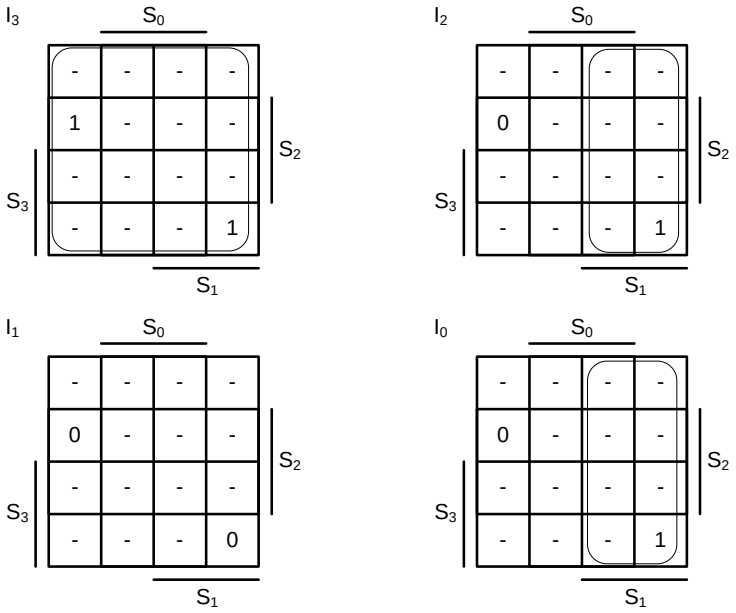
The K-map for the LD signal is:




where we can get $LD = x(S_3'S_2 + S_3S_2'S_1)$.

The values of $I_0$ to $I_3$ need to be: $I = \begin{cases} 1000 & \text{if } S = 4, \\ 1101 & \text{if } S = 10, \\ DC & \text{otherwise} \end{cases}$

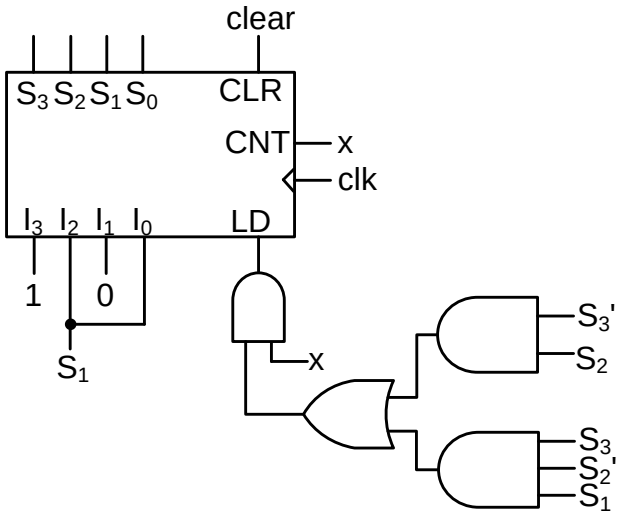
The K-maps are as follows:




and we can get

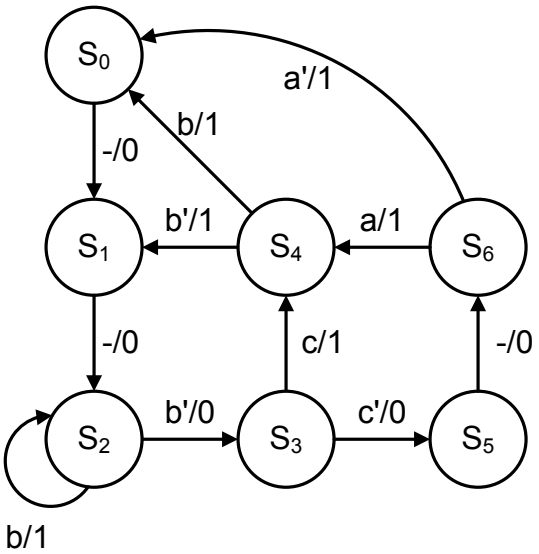
$$\begin{aligned} I_3 &= 1 \\ I_2 &= S_1 \\ I_1 &= 0 \\ I_0 &= S_1 \end{aligned}$$

The counter implementation is:




**Problem 6**

Implement a sequential system using the standard modulo-16 binary counter with parallel input. The system has three binary inputs $a$, $b$, $c$ and one binary output $z$. The transition and the output functions are specified by the state diagram shown below.




1. Setting $LD = CNT'$, obtain the expression for the input $CNT$ in terms of states ($S_0$ to $S_6$) and input signals.

   ***Solution*** Looking at the state diagram, we find all cases where the state index increments by 1. This gives us:

   $$CNT = S_0 + S_1 + S_2 b' + S_3 c + S_5$$

2. Find all load input combinations that we need for this system. For each combination, write the input conditions in terms of states ($S_0$ to $S_6$) and input signals. Simplify the conditions as much as possible.

   ***Solution*** We utilize parallel loads when the transition is not an increment, i. e. cannot be implemented by setting $CNT = 1$. The states are $S_0$, $S_1$, $S_2$, $S_4$ and $S_5$. The expressions we get directly are shown:

$$I_3I_2I_1I_0 = \begin{cases} 0000 & \text{if} \quad S_4b + S_6a' \\ 0001 & \text{if} \quad S_4b' \\ 0010 & \text{if} \quad S_2b \\ 0100 & \text{if} \quad S_6a \\ 0101 & \text{if} \quad S_3c' \end{cases}$$

To simplify the expressions, the input can be removed for conditions that are tied to a certain state, since the $LD$ signal is only invoked when $CNT = 0$. For instance, when we are at $S_2$ and $LD = 1$, $b$ will never be 0 because if $b = 0$, then $S_2b' = 1$ and $CNT$ must be 1 also. For $S_4$ and $S_6$, we cannot remove the input variable as the load value is affected by the input signal. For example, at $S_4$, if $b = 0$ then $I_0 = 1$, if $b = 1$ then $I_0 = 0$.

With the simplifications in place, we get:

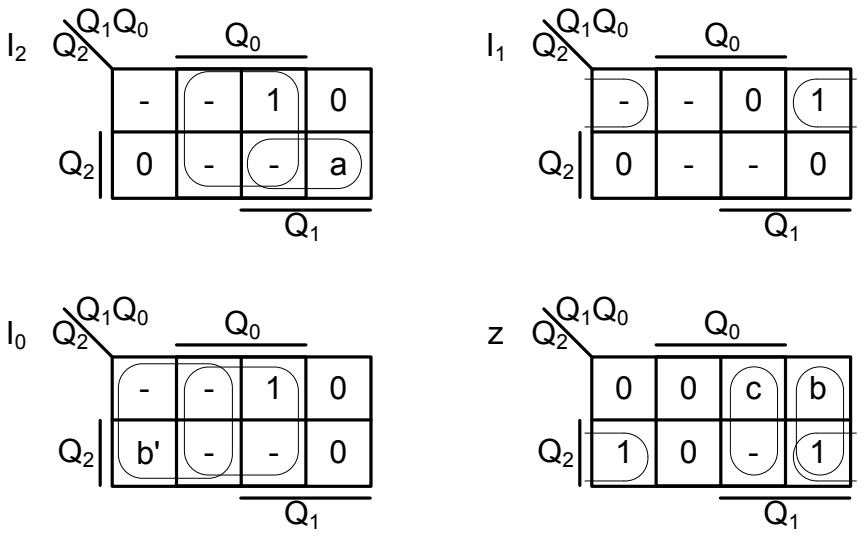
$$I_3I_2I_1I_0 = \begin{cases} 0000 & \text{if} \quad S_4b + S_6a' \\ 0001 & \text{if} \quad S_4b' \\ 0010 & \text{if} \quad S_2 \\ 0100 & \text{if} \quad S_6a \\ 0101 & \text{if} \quad S_3 \end{cases}$$

3. Using K-maps, obtain the minimal expressions for $I_3$ to $I_0$ and the output $z$, in terms of counter outputs ($Q_3$ to $Q_0$) and input signals.

***Solution*** We take each $I_i$ as a separate output and build the K-maps with respect to the state values. For example, looking at the answer from the previous section, $I_2$ needs to be 0 for $S_4b + S_6a'$, $S_4b'$ and $S_2$, and it needs to be 1 for $S_6a$ and $S_3$. The state number indicates the square on the K-map, and any variables become the values inside the squares. In the index 6 square for $I_2$, the output needs to be 0 for $a' = 1$ (or $a = 0$) and 1 for $a = 1$, therefore the value that goes here is $a$.

Similar rules apply for the output $z$. We can find an output condition for each state in the state diagram, and write a value in the K-map for each one. For example, $S_0$'s output is 0 regardless of the input, so we put a 0 in the square for index 0 in the K-map. At $S_3$, the output is 0 when $c' = 1$ (or $c = 0$) and 1 when $c = 1$. So we write $c$ in the square for index 3.

Following these principles, we can obtain the K-maps below. We do not need a K-map for $I_3$ since it is always 0.




Regarding covers that contain variables, we can consider them as special covers that will only cover that variable. They should never cover any 0 squares, and any 1 squares that they cover are considered incomplete,

since they will only be valid in cases when the variable is 1. Therefore, all constant 1 squares need to have a purely constant 1 cover to be completely covered.

For instance, in the K-map for $z$, we will need a separate constant 1 cover for the square at index 6 even if the value at index 4 were to be 0. This is because the $b$ cover is only partially covering index 6. And in the same K-map, if index 0 were to be 1, we still cannot have a cover like $Q_0$ in the final expression, which is a constant 1 cover going over a variable square. Since variable squares can be 0 or 1 as the input changes, any constant cover should never cover any variables.

The final expressions for each load input and the output is:

$$\begin{aligned} I_3 &= 0 \\ I_2 &= Q_0 + Q_2 Q_1 a \\ I_1 &= {Q_2}' {Q_0}' \\ I_0 &= Q_0 + {Q_1}' {b}' \\ z &= Q_2 {Q_0}' + Q_1 {Q_0}' b + Q_1 Q_0 c \end{aligned}$$