DSD Lab 1 (20-09-2021)

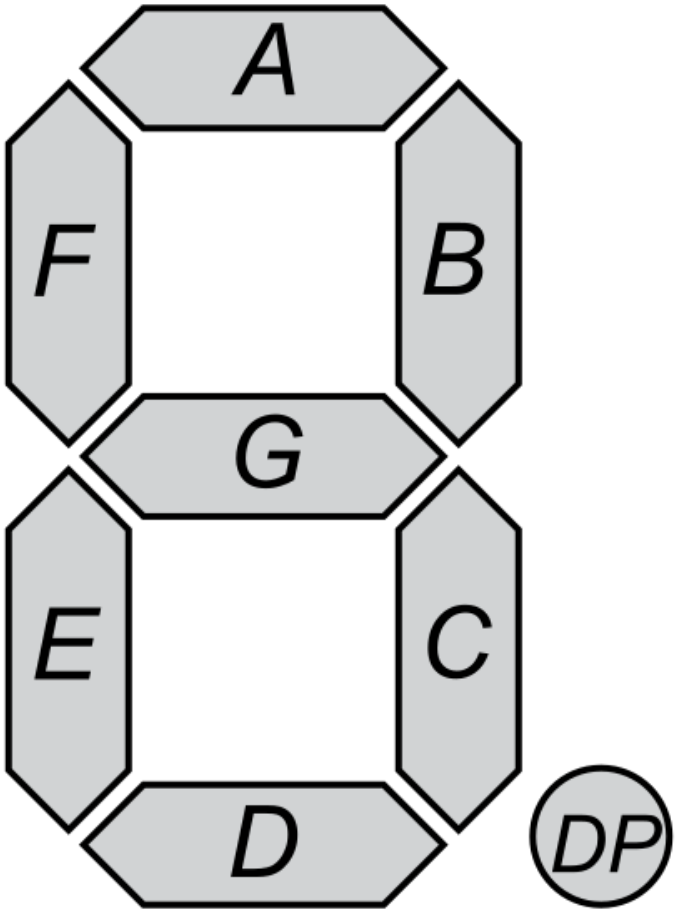
Submitted by:

R.S Muthukumar

201EC149

1. **BCD to seven segment decoder. Use the seven segment display in logisim for output.**

Ans)

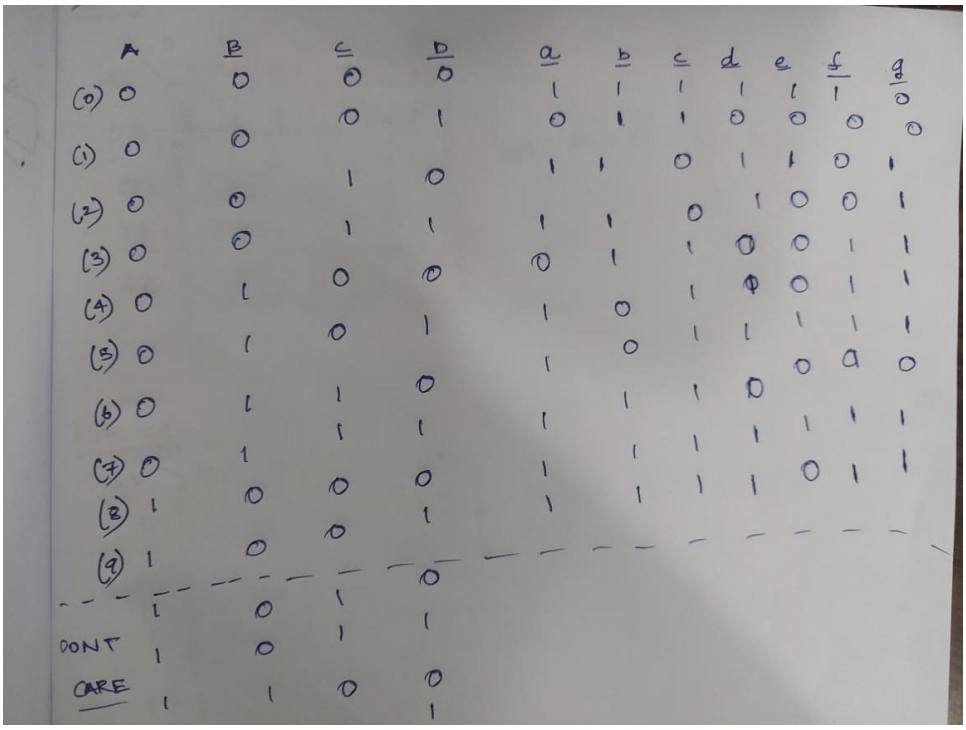


This is the picture of the seven segment display that we are going to find the system design for. As we can see, there are 7 different parts to this display, which light up to finally show us the number. Suppose the number if 2. In that case, only [A, B, G, E and D] need to light up. Rest need not light up. So, we need to design a circuit such that according to the input, specific segments of the 7-segment display light up.

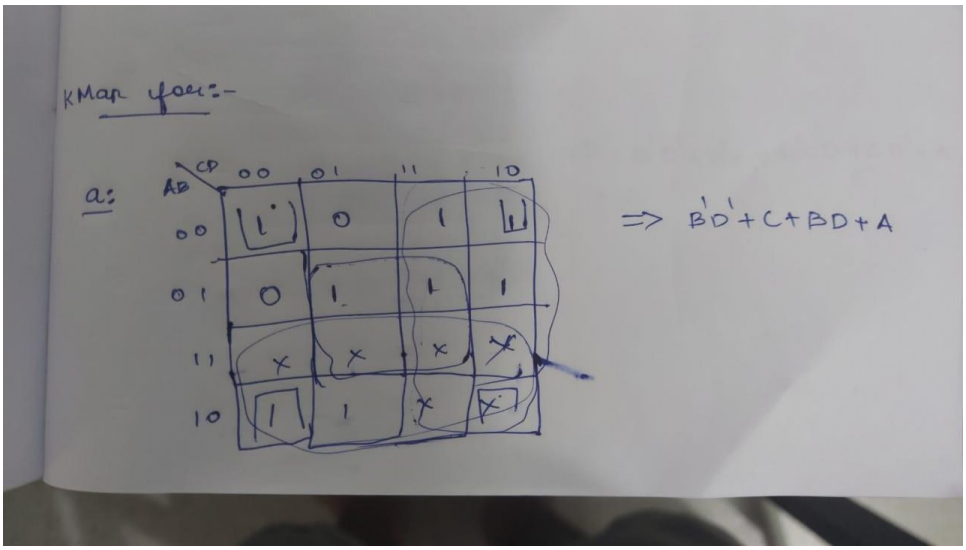
For each number, it is different segments that light up.

- 1: F, E
- 2: A, B, G, E, D
- 3: A, B, C, D, G
- 4: B, C, F, G
- 5: A, C, D, F, G
- 6: A, C, D, E, F, G
- 7: A, B, C
- 8: A, B, C, D, E, F, G
- 9: A, B, C, D, F, G
- 0: A, B, C, D, E, F

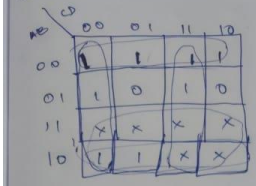
Now, since we have this data, we can write the following truth table:



And for each A, B, C …. G, we can draw a kmap as follows:

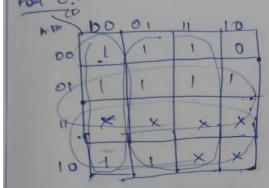


For b:-



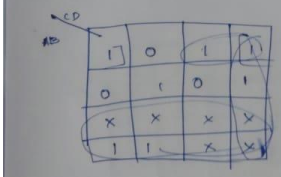
$\Rightarrow B' + C'D' + CD$

For c:-



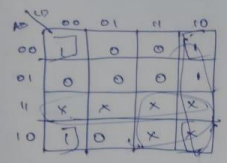
$\Rightarrow C' + D + B$

For d:-



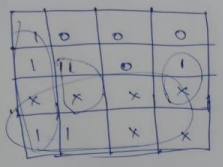
$\Rightarrow B'D' + BC + B'CD + CD' +$

For e:-



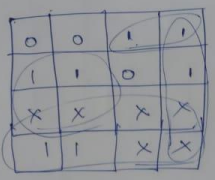
$\Rightarrow B'D' + CD'$

For f:-



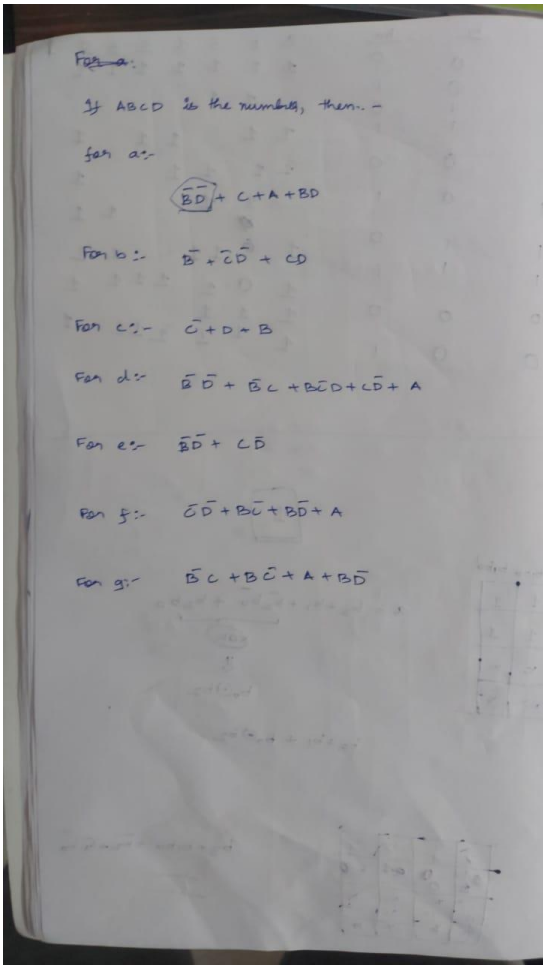
$\Rightarrow a'b' + Bc' + BD' + A$

For g:-



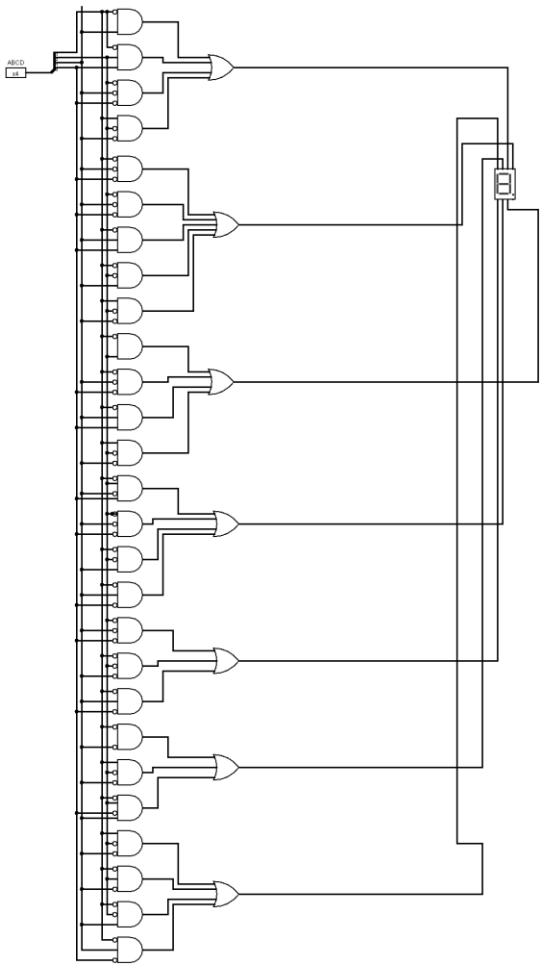
$\Rightarrow B'C + BC' + A + BD'$

And finally, to summarize everything:



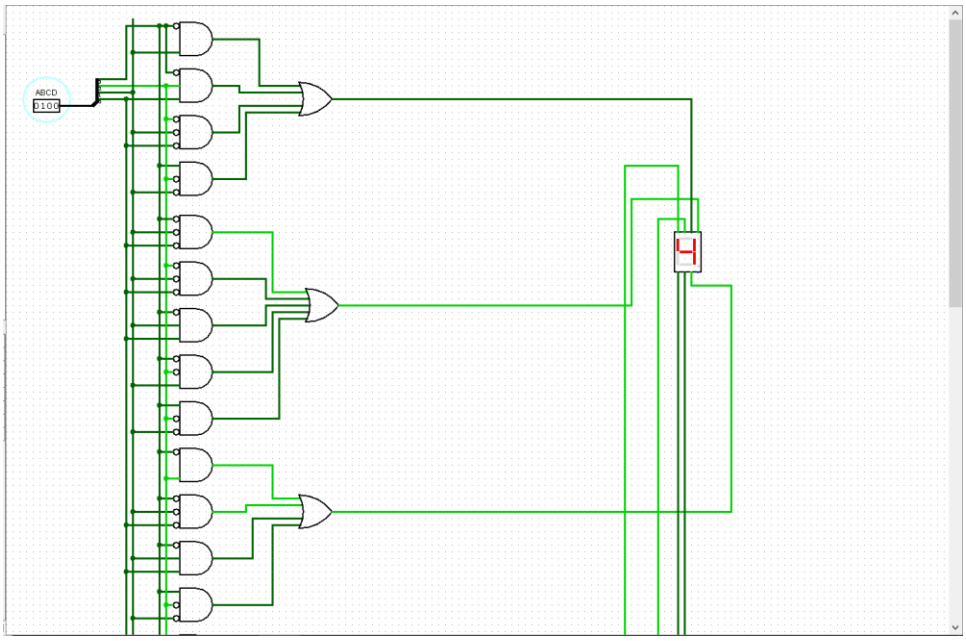
These are the gates that are required for each and every segment.

Circuit diagram from logisim:

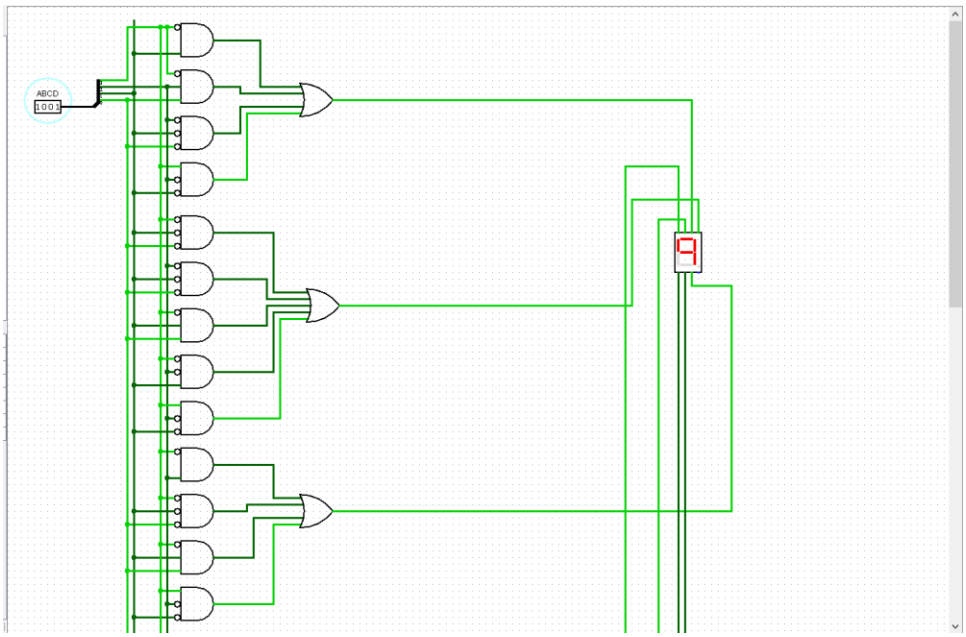


Results obtained:

Input: 4 [0100]



[Could not take complete screenshot because of size restrictions, but 0100 gives 4 as the output, which is what is expected.]



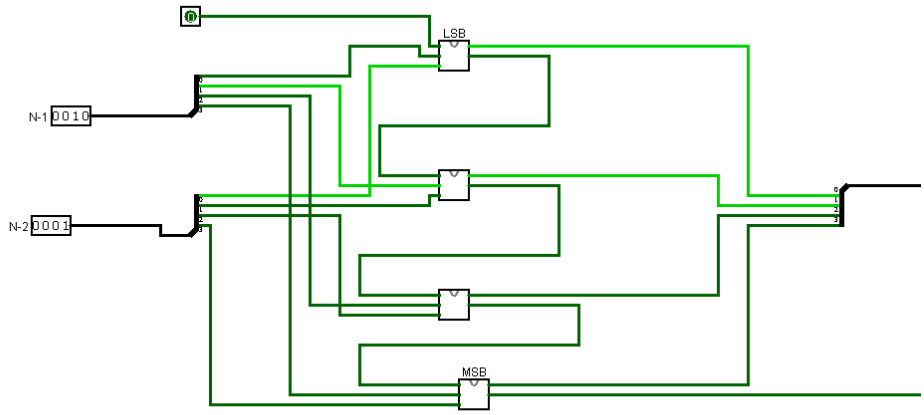
9 is the output when input is 1001, which is also correct

2. **Connect the BCD to seven segment decoder to the output of the 4 bit adder implemented in last class such that the inputs A and B satisfy A+B<=9.**

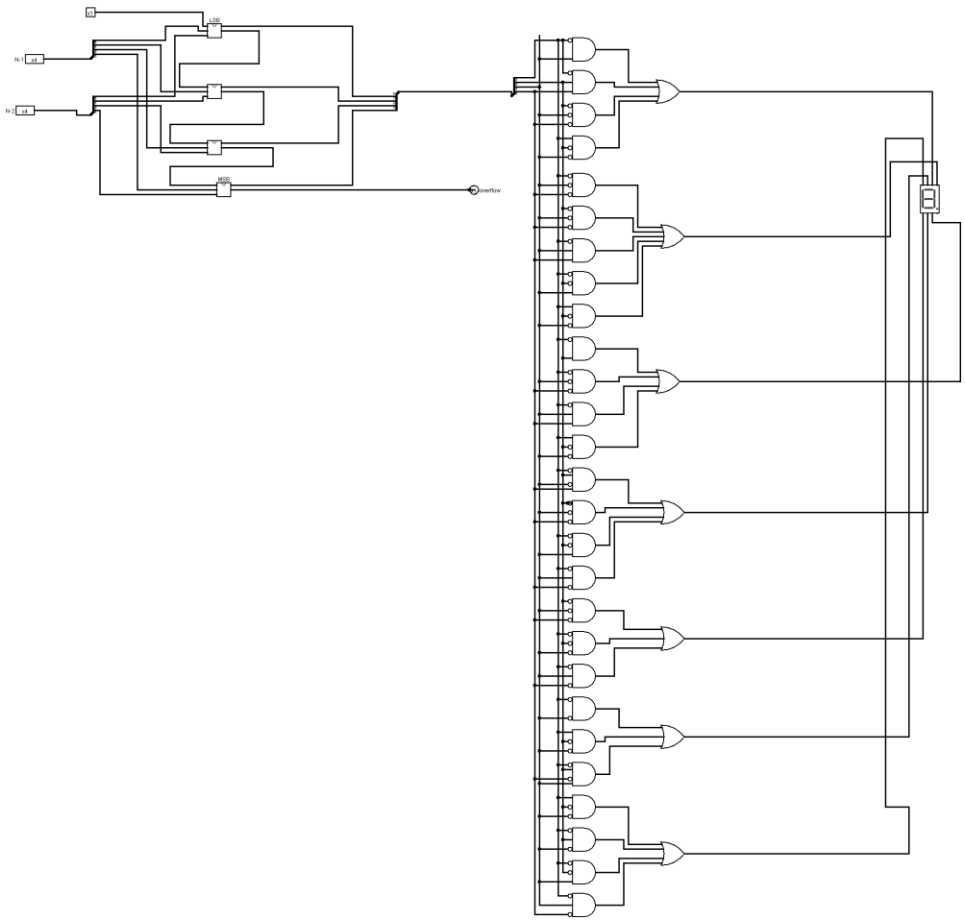
Ans)

In the previous question, we have already shown how to design a 7-segment decoder using logic games. So, here, we will be connecting the output of a 4 bit-adder, that takes two inputs A and B and returns back the sum. This is then connected to the 7-segment display, which displays the sum.

The 4 bit adder works on the principle of ripple effect, where after every bit is added, a carry over is sent to the next addition, which send one to the next and so on. So it is a combination of 4 1 bit adders. And produces a ripple effect.

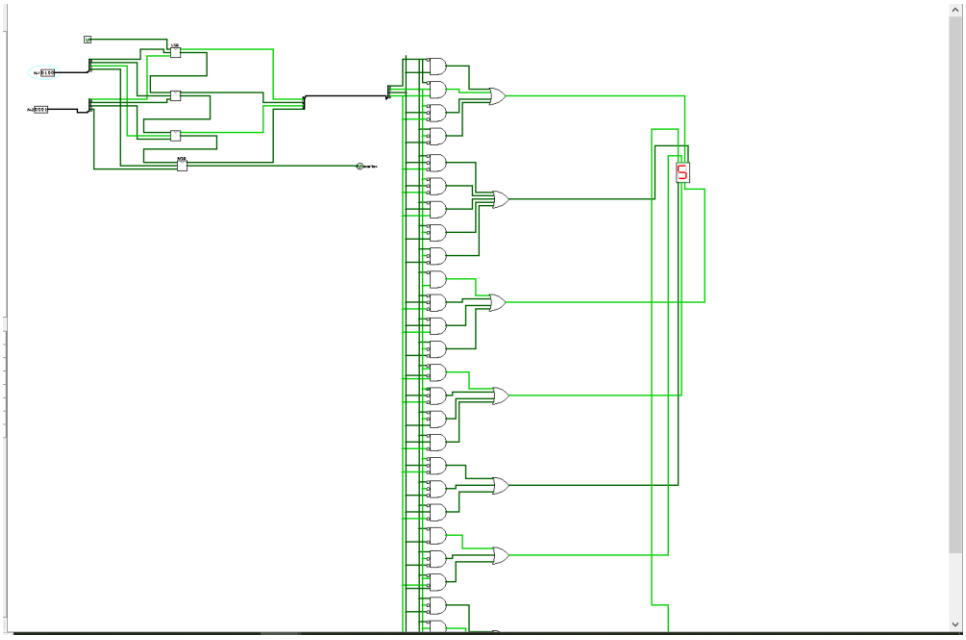


The 4 boxes are the 4 half single bit adders, which returns a carry and a sum. This output, which is received in the splitter, is then connected to the 7 segment display, to yield the following result:

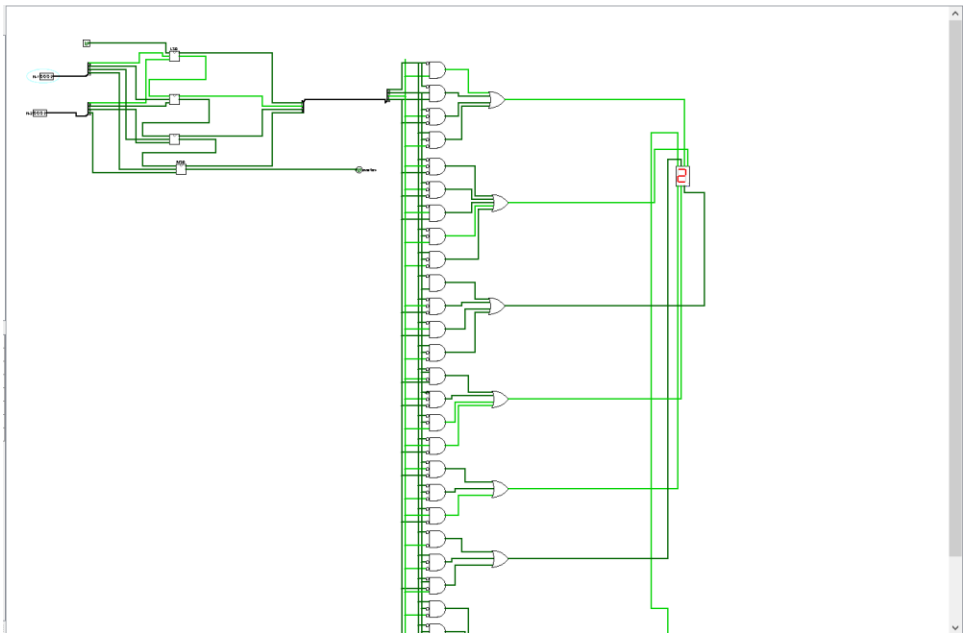

N
LSB
N-1
N-2
MSB
overflow

Results obtained:

Input: A = 4 [0100] B = 1[0001]



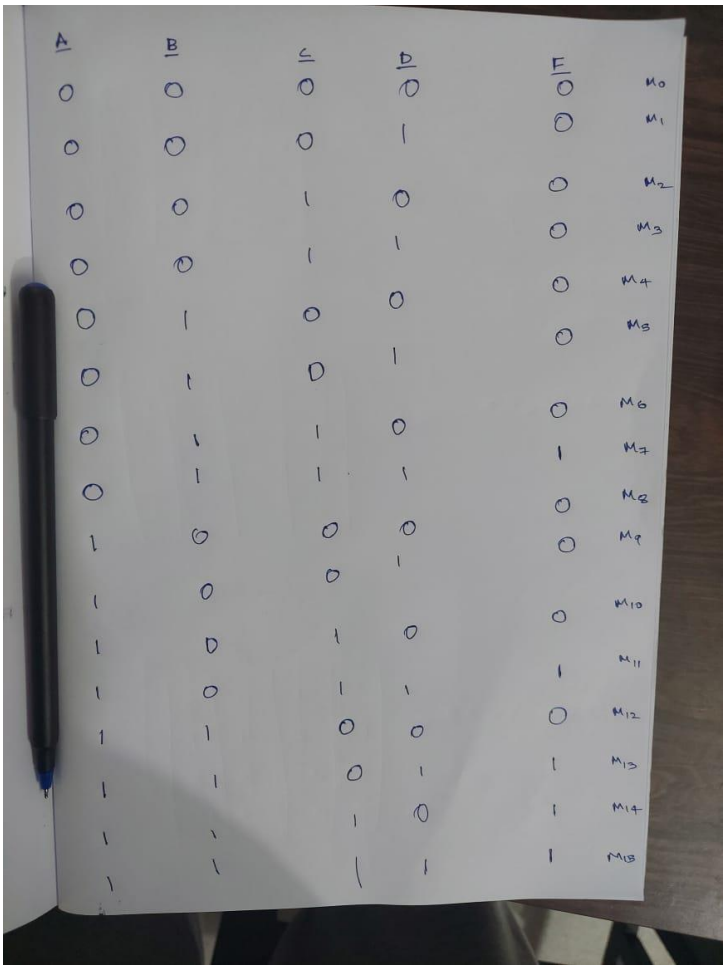
As we can see, 5 is the output we get which is correct, because 4+1 is 5 and hence both our adder as well as our 7 segment display are working properly.



When the inputs are 1 and 1 [0001] and [0001], then the output is 2, which is also correct.

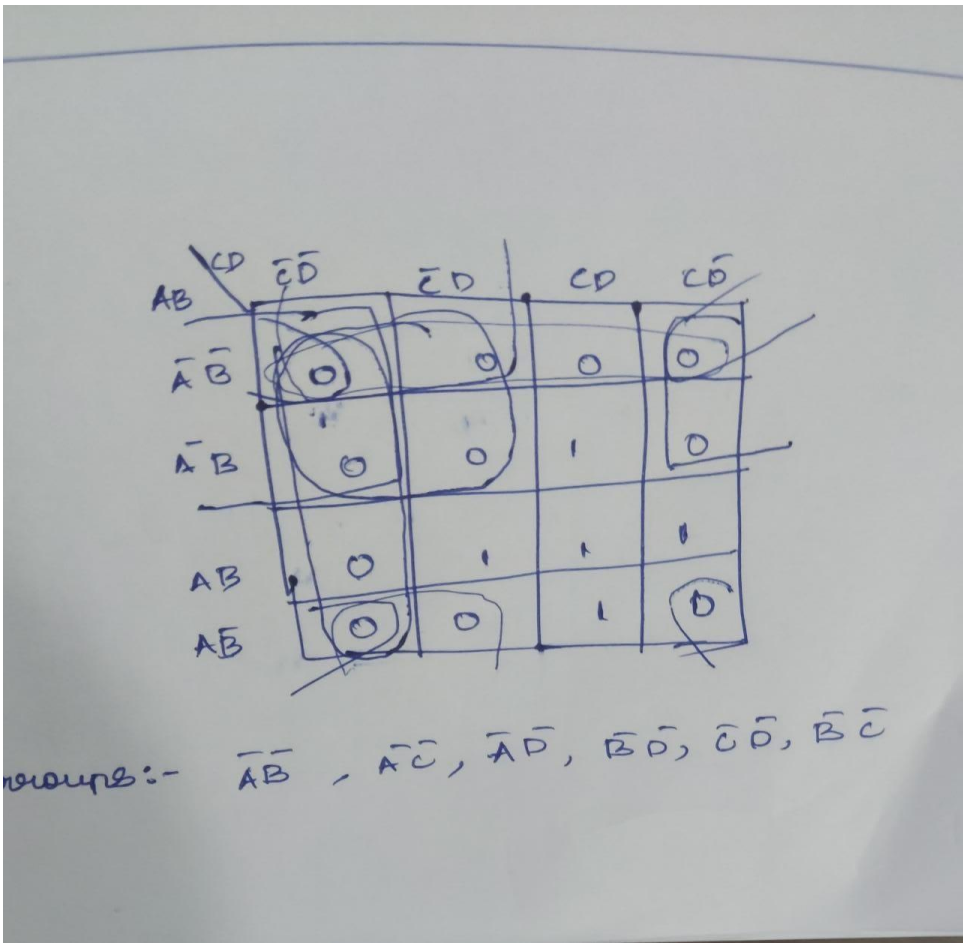
3. **A four variable logic function that is equal to 1 if any three or all four of its variables are equal to 1 is called a majority function. Design a minimum cost POS circuit using NOR gates that implements this majority function.**

Ans) The truth table for this will be as follows:

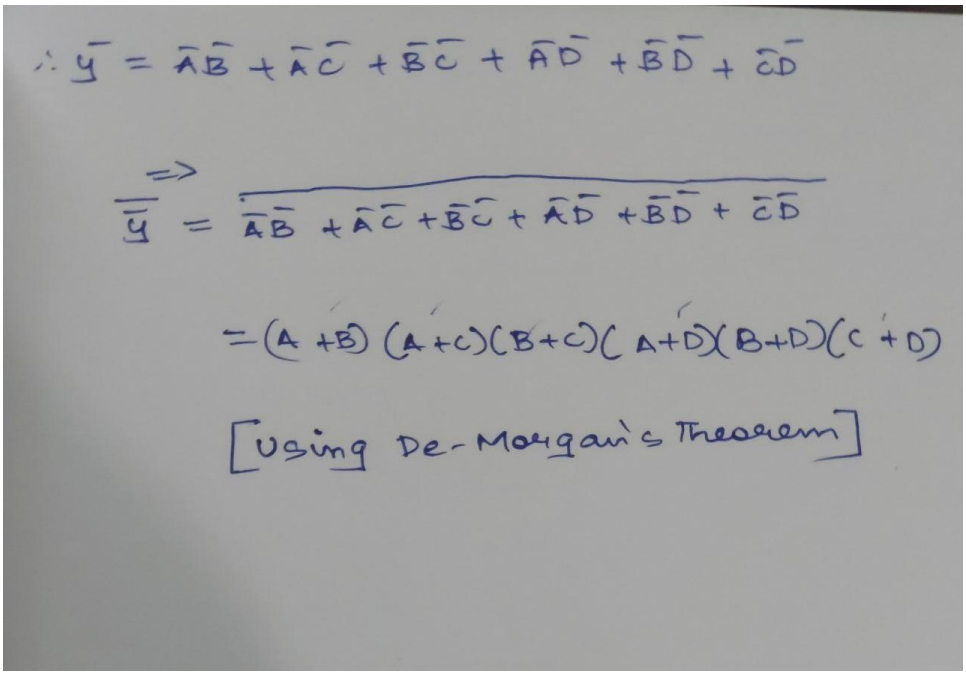


We are concerned about the POS form of this expression, so we make a K-map and do the following

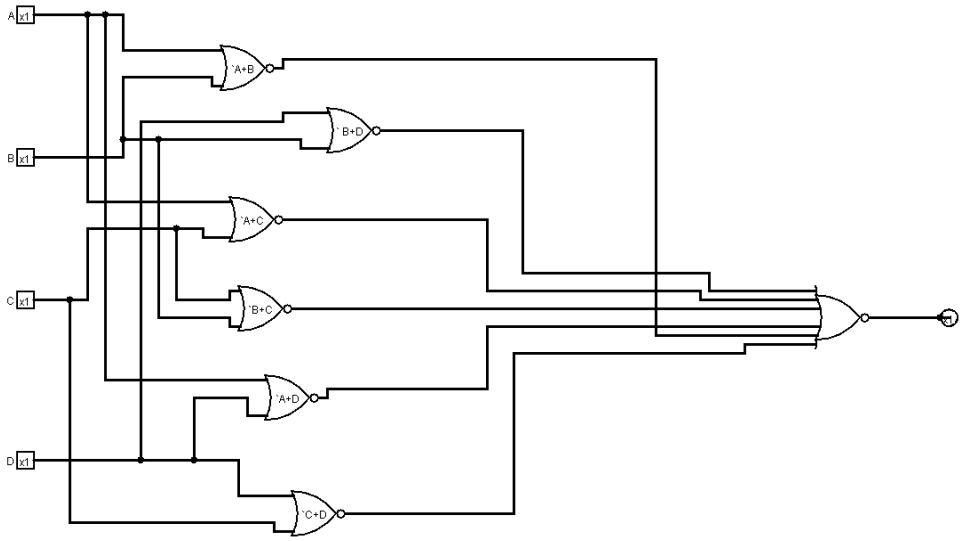
As we can see, M0, M1, M2, M3, M4, M5, M6, M8, M9, M10, M12 are all 0s. So if we draw the Kmap for this:



We can write the groups as:



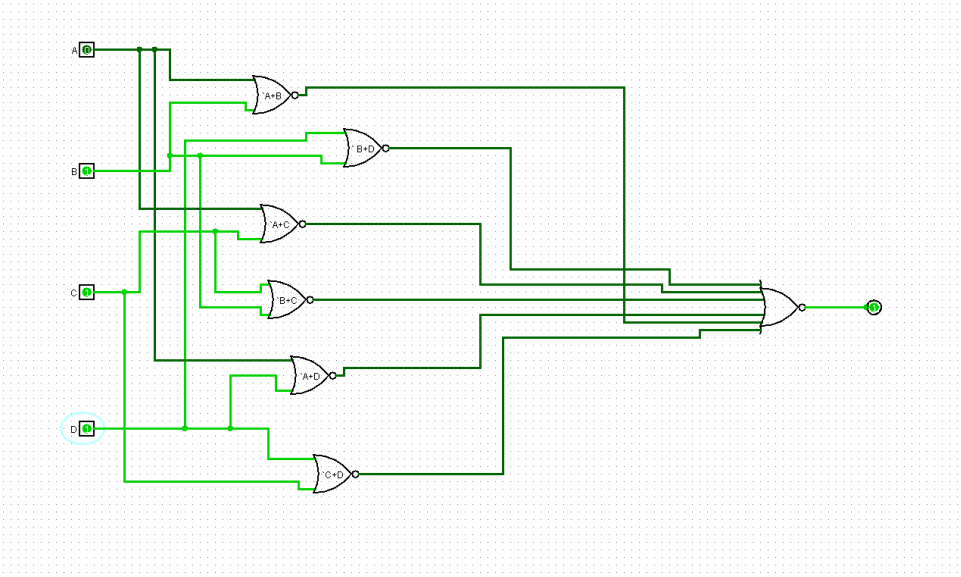
Now, we can draw the circuit using this in logisim as follows:



Results obtained:

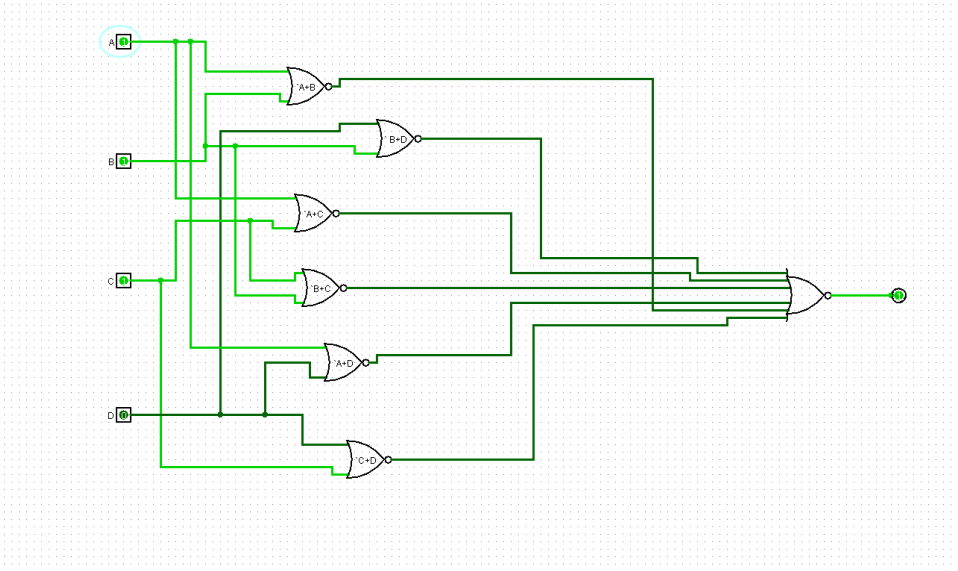
Input: A =0, B=1, C=1, D=1

Output: 1, which is correct since majority is 1.



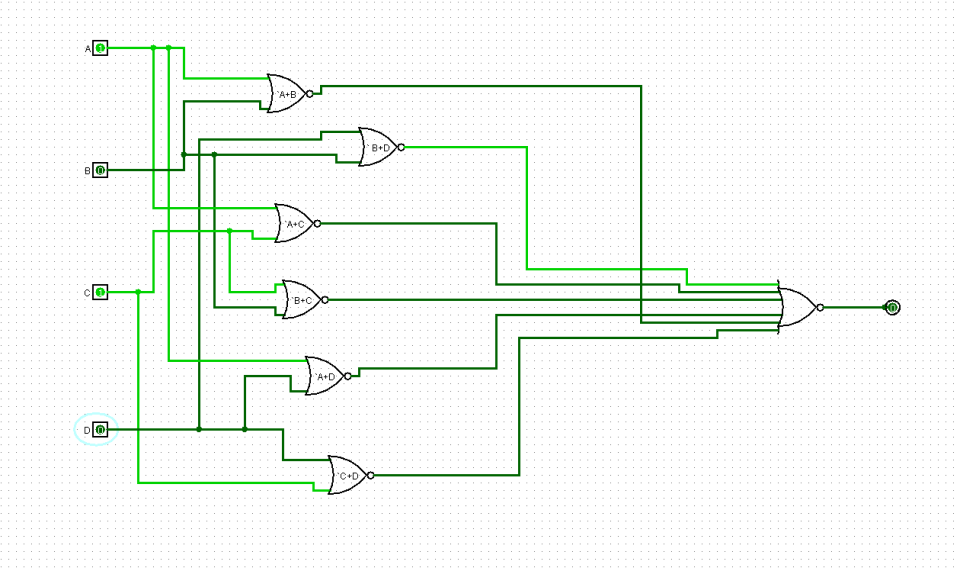
Input: A =1, B=1, C=1, D=0

Output: 1, which is correct since majority is 1.



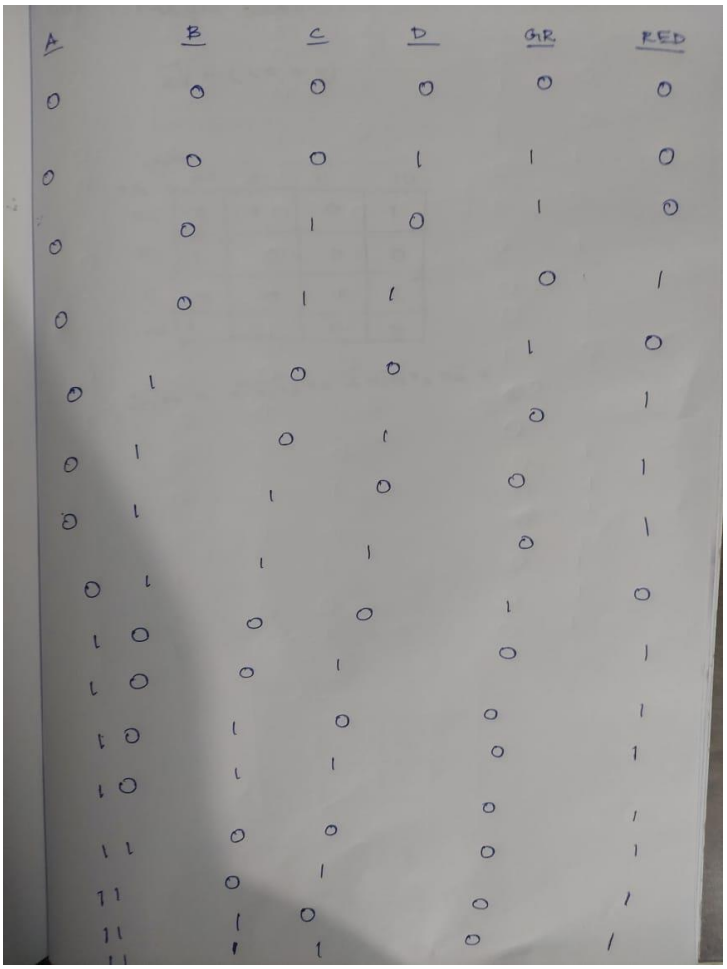
Input: A =1, B=0, C=1, D=0

Output: 0, which is correct since there is no majority of 1.

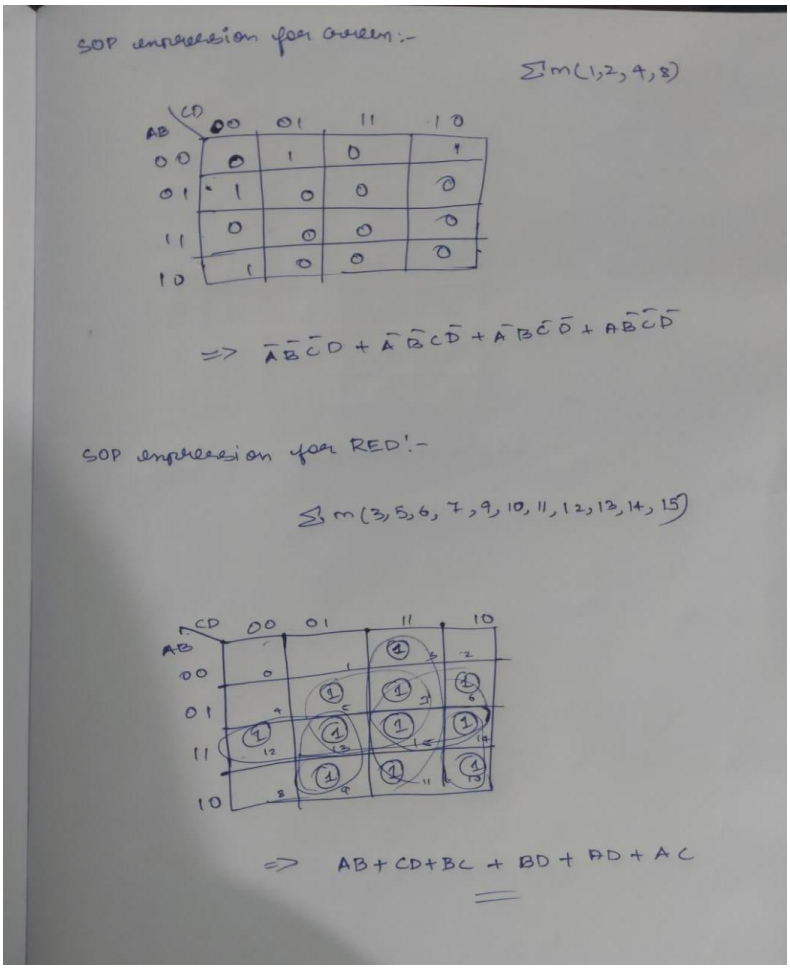


4. **A given system has four sensors that produce an output of 0 or 1. The system operates properly when exactly one of the sensors has its output equal to 1. An alarm must be raised when two or more sensors have the output of 1. Design the simplest circuit that can be used to raise the alarm. Use a red LED to indicate ALARM and a green LED to indicate the system operated properly.**

Ans) The truth table for this will be as follows: Since we have two outputs, we will need two K maps. Let the outputs be RED and GREEN. There are 4 inputs.

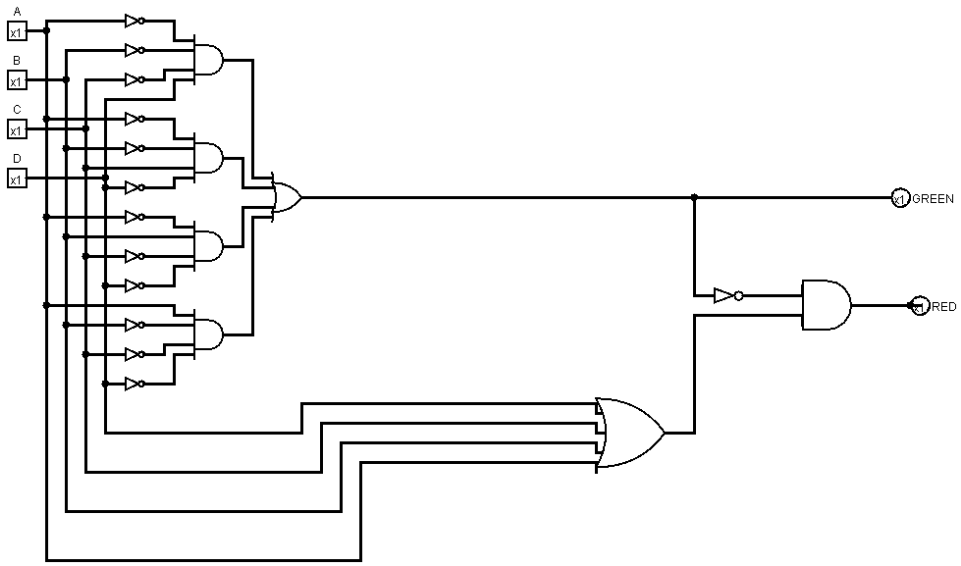


Now, let us draw the Kmap for GR and RED, which are our outputs

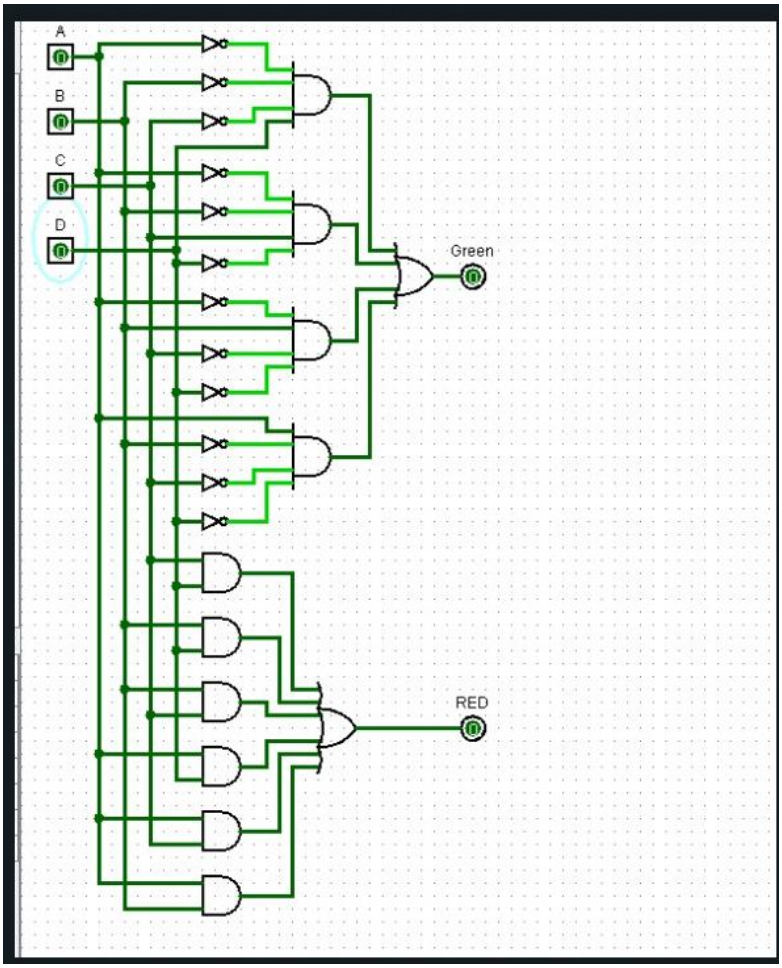


Now, using the minimized expresssion, we can design the circuit as follows:

The most important point to be noted in the circuit is that, the whole RED LED can be written as a combination of the output of GREEN LED. So, the RED's output is the complimentary of GREEN's output **except** when the inputs are all zero. To check this, we send it through an OR gate, then through an AND gate, hence minimizing the number of gates and cost.



If we plot just the kmap logic of this circuit, we will get the following:



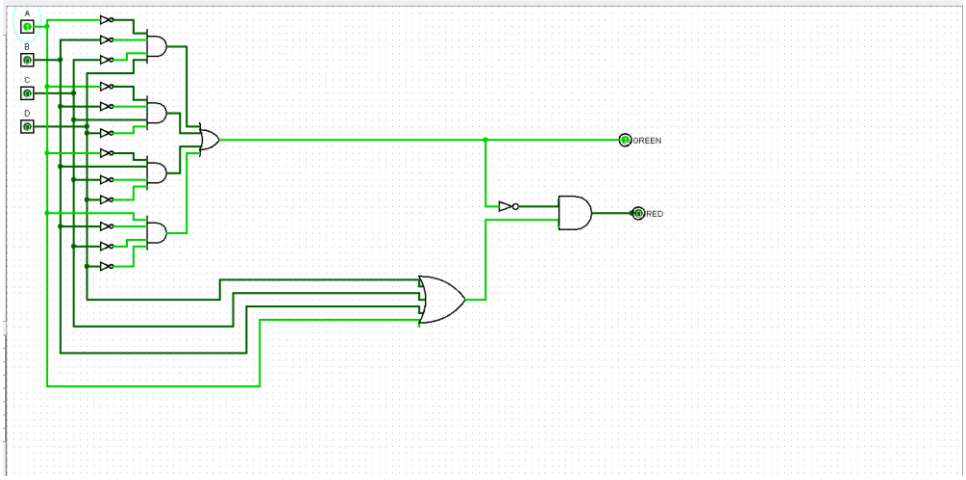
As we can see, this circuit has far too many gates and hence is very cost innefficiecnt. So, we discard this circuit and use the one that was described above, since it is very efficient.

Result

Input: 1000
Output: Green: 1

RED: 0

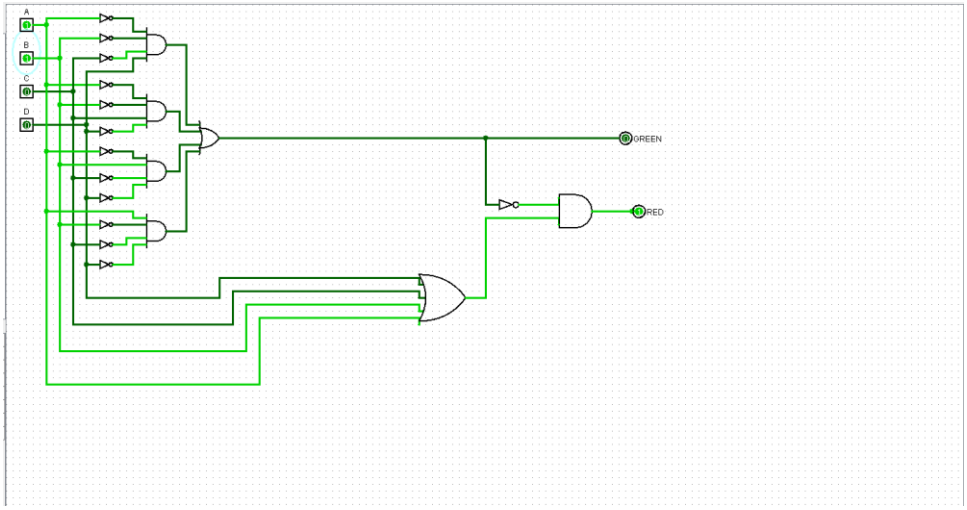
Which is correct because if there is only one 1, then it should correspond only to green.



Input: 1100
Output: RED: 1

GREEN: 0

2 sensors have hence green should be low and RED should be high.


A
B
C
D
GREEN
RED