PROJECT 10

Contents

P 10.0: Logisim

Intro

In this mini-project I will try to make a simple representation of digit numbers 0-9, by a given binary input.

The binary input number, hence, will be between 0000-1001. Thus, 4 variables, named, w, x, y, z can be used to make the Boolean functions for the combinational circuit.

7 bars

Before proceeding in making the combinational circuit, it is first important to know how many outputs there will be given w, x, y, z.

Thus, the main thinking is how can we represents the digits 0-9 with minimum possible bars but still comprehensible to read them.

Just like the analogy of LCD digital clocks/watches, digits can be shaped more "rectangularly" as follows:

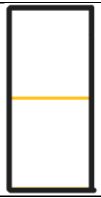
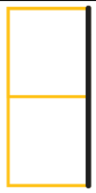
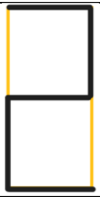
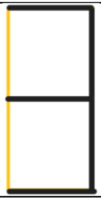
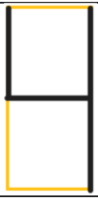
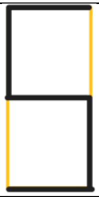
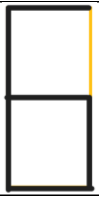
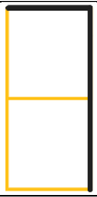
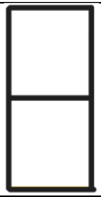
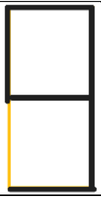
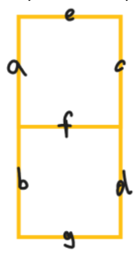
0	1	2	3	4	5	6	7	8	9
									

Table 1: screenshots taken from Microsoft Whiteboard

It can be noticed that 7 bars are indeed a good solution.

Therefore:

- there will be 4 inputs $w, x, y, z \in \{0,1\}$, that represent a binary number $wxyz$,
- 7 outputs, named, a, b, c, d, e, f, g in the following format:



Boolean functions

Now for each output we can create a Boolean function. Since the there are 4 inputs a 4D Karnaugh map can be used for assisting in simplification and optimization of Boolean functions.

The source of the 4D K-map is mentioned here:
https://www.tutorialspoint.com/digital_circuits/images/4_variable_k_map.jpg

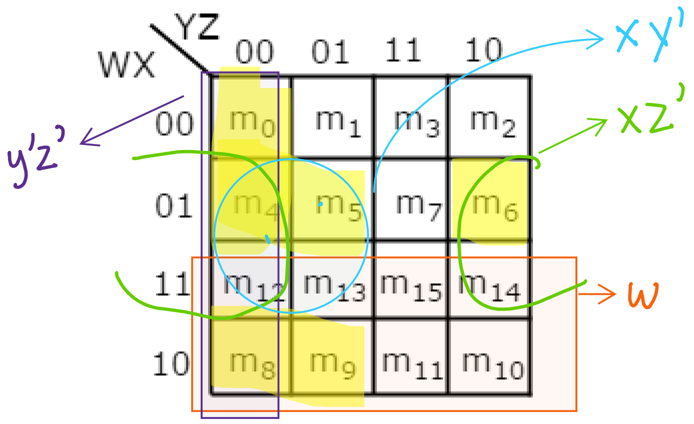
Note that for numbers between 10 and 15, inclusively, are IRRELEVANT for the design of Boolean functions, since those will be assumed to be INVALID inputs.

We assume the input is always between 0-9, inclusively, 0000-1001.

However, these IRRELEVANT numbers may be used in making groups in K-maps larger, which aid in smaller Boolean functions, thus more optimized.

Output a

Looking at Table 1, the bar a would be on (so $a = 1$) if the inputs are $0,4,5,6,8 \; or \; 9$.



Therefore, the Boolean function is:

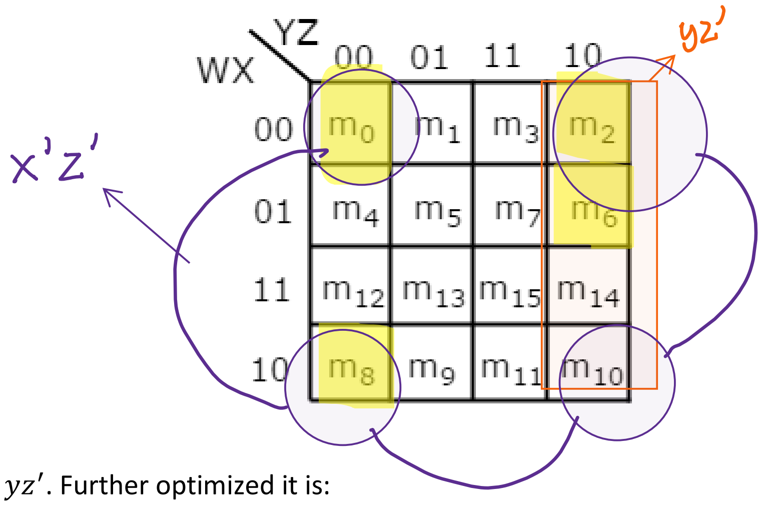
$$a = w + y'z' + xy' + xz'$$

Further optimized:

$$a = w + y'z' + x(y' + z')$$

Output b

Bar b is set when inputs are $0,2,6,8$.

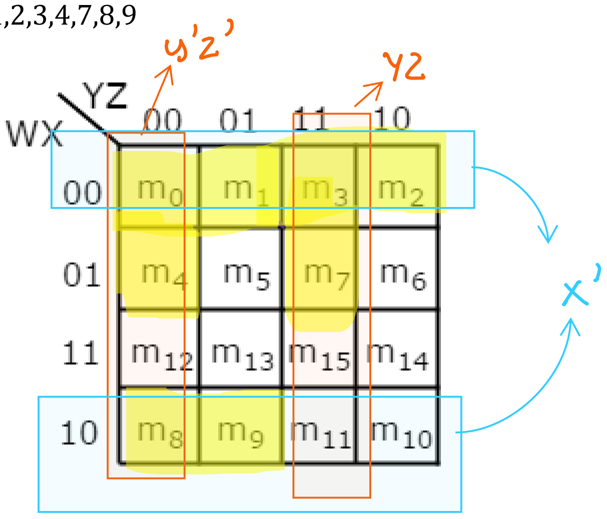


So, $b = x'z' + yz'$. Further optimized it is:

$$b = (x' + y)z'$$

Output c

Bar c is set when inputs are 0,1,2,3,4,7,8,9

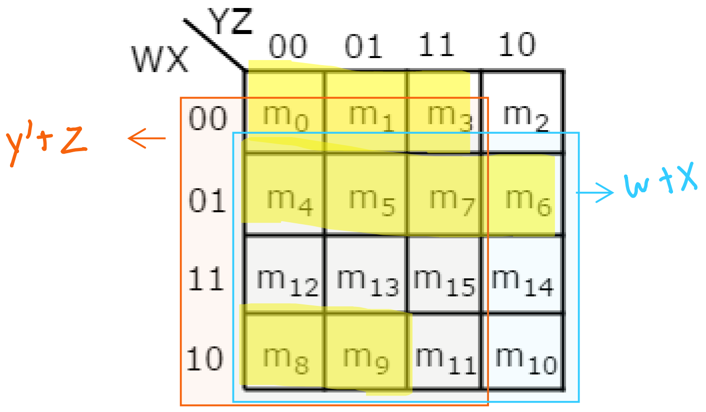


So,

$$c = x' + yz + y'z'$$

Output d

Bar d is set when inputs are 0,1,3,4,5,6,7,8,9

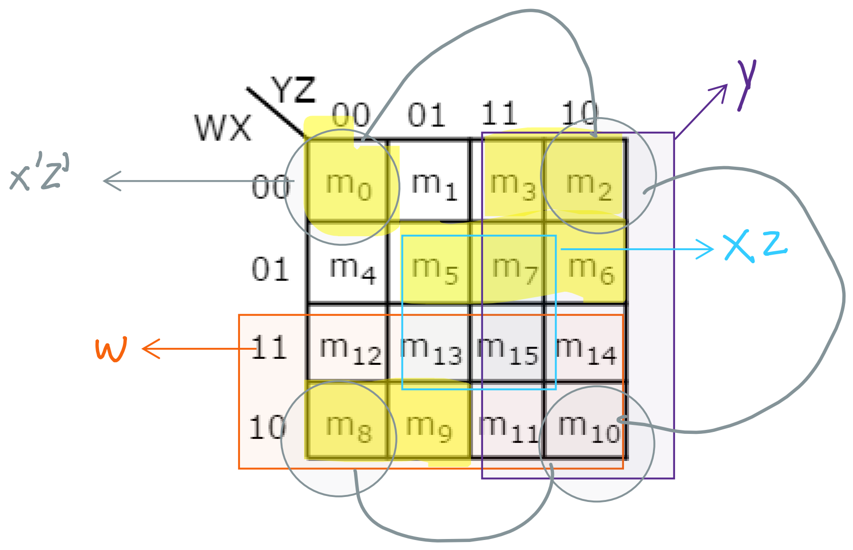


So,

$$d = w + x + y' + z$$

Output e

Bar e is set when inputs are 0,2,3,5,6,7,8,9

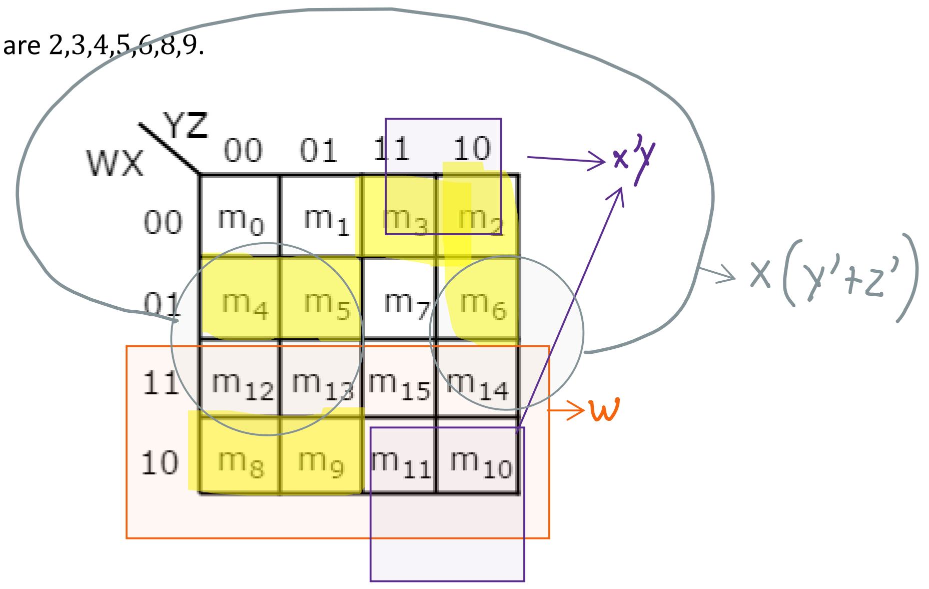


So,

$$e = w + y + xz + x'z'$$

Output f

Bar f is set when inputs are 2,3,4,5,6,8,9.



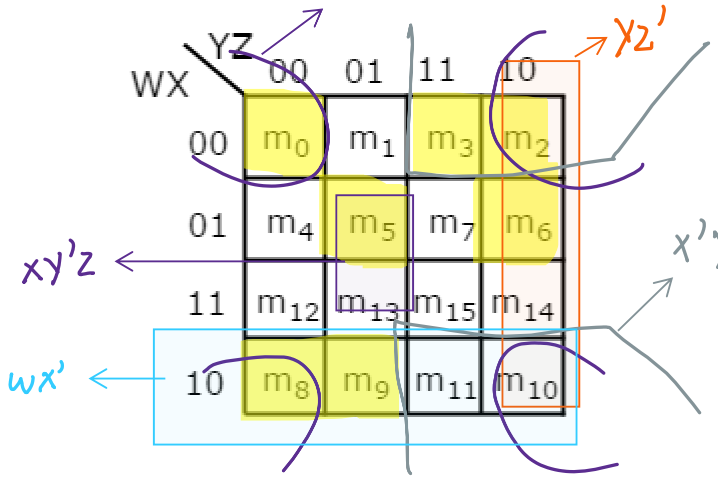
So,

$$f = w + x'y + x(y' + z')$$

Output g

Bar g is set when inputs are 0,2,3,5,6,8,9.

$x'z'$



So,

$$g = wx' + x'z' + x'y + yz' + xy'z$$

Further optimized:

$$g = wx' + x'y + x'z' + yz' + xy'z$$

$$g = x'(w + y) + (x' + y)z' + xy'z$$

Further optimized via substitution:

$$g = x'(w + y) + b + xy'z$$

Overall optimization

Note that some Boolean functions, such as $y' + z'$, which are part of Boolean functions of a and f, are repeated. This implies that we do not need to make duplicate/redundant use of logic gates.

This allows to further reduce the count of overall logic gates.

However, note that through the use of K-Maps the solution has been optimized at a significant level. Although it may not be the best optimal solution with the fewest logic gates. The optimization procedures have been applied to the extent of intuitive capabilities as much as possible.

Logisim design

The design is planned as follows:

- Make a combinational circuit with 4 inputs and 7 outputs, with logic gates representing Boolean functions as represented above
- Use the combinational circuit to show digits using LED lights
- Use the combinational circuit to show digits using the 7-segment display
  - I noticed I could use the 7-segment display as well

You may see the file with ".circ" extension, using Logisim application (".exe" file included) in the 'section Logisim' folder.

Demos

You may access files with prefix name “1” and “2” in the ‘demos’ folder.

P 10.0: Raspberry Pi

Intro

In this mini-project I will try to make a simple representation of digit numbers 0-9 on the breadboard using 7 output pins, each pin corresponding to a bar of a 7-segment display. GPIO pins of Raspberry Pi – 4B are used.

There will be a basic interaction with left and right buttons that allow to decrement and increment the value of digit displayed. Using the same code, first using a set of LED lights were used to show the digits in a creative manner. Then a 7 – segment display was used to show the same.

GPIO pins used

There are various pins that can be selected. The ones that were chosen are shown in the picture below:

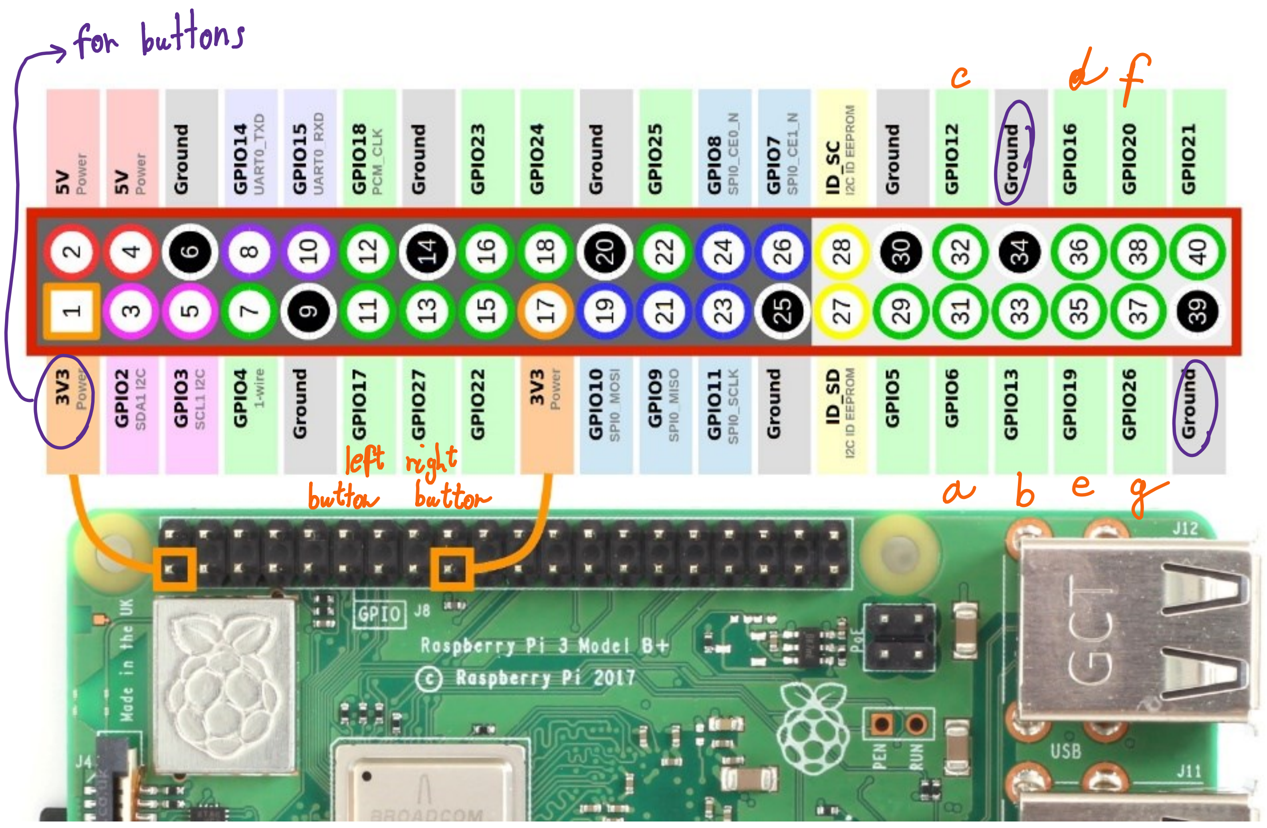
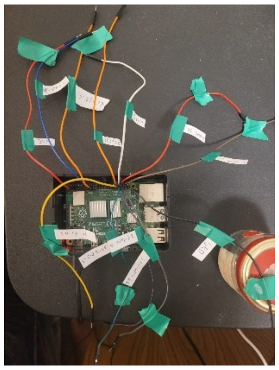


Figure 1: GPIO pins used. Image from https://www.raspberrypi-spy.co.uk/wp-content/uploads/2012/06/Raspberry-Pi-GPIO-Header-with-Photo.png

Pins labelled a, b, c, d, e, f, g will be used as OUTPUT pins and each pin will be connected to the shorter side, anode (+), of the LED lights. Whereas the Ground pin will be connected to the longer side, cathode(-).

Whereas for 7 – segment display the 3rd pin (in the middle of 5 pins on each side) will be connected to ground pin. The output pins will be connected to the other respective pins. Note that the DP pin will not be connected to any since decimal point will not be used.

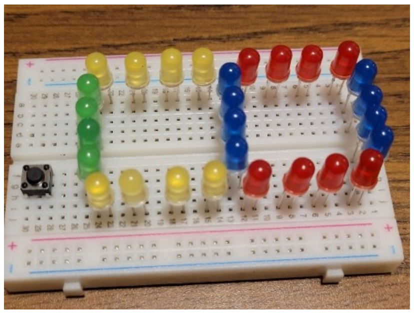
Breadboard setup



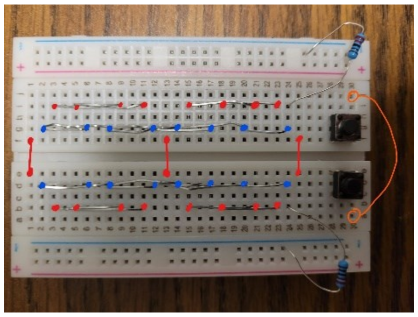
The subfolder 'pics' has images of the setup of breadboard.

Picture '1' shows the GPIO, 3v3 and ground pins used.

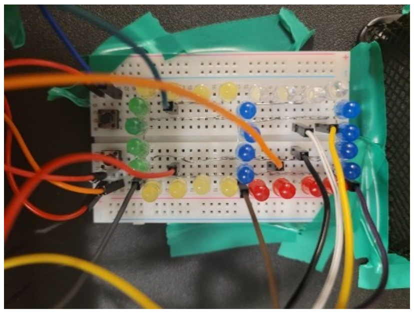
LED lights



Initially LED lights were placed, as shown in pictures '2a' and '2b', to see how the complete setup should be. Cathode /Anode positions LED lights were memorized. Then they were removed.

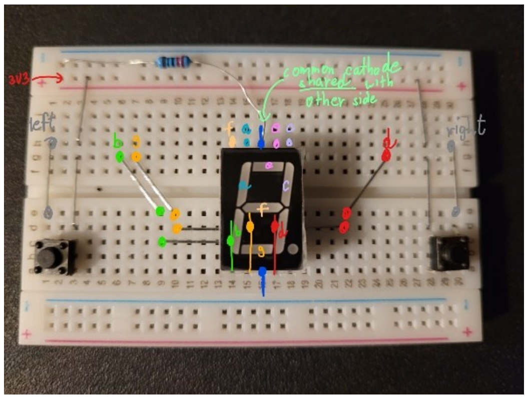


Then using staples the cathode (-) positions of LED lights were connected to a common negative (-) line on each half of breadboard. Each respective set of LED lights' anodes (+) were connected together as well so that later the corresponding output pin needs to be inserted in one place only. Pictures '3' shows this set up of positive(+) and negative(-) lines.



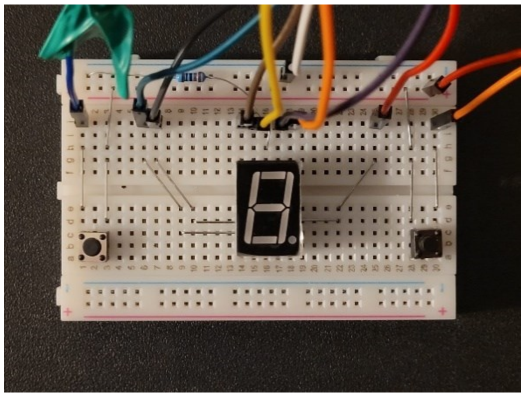
Then in picture '4' the complete LED lights setup is done. Left and right buttons are connected with common 3v3 line and respectively to GPIO 17 and 27. However note that resistors (220R) were removed since during the tests they were not allowing all LED lights to be on (too much resistance).

7 – segment display



In picture ‘5’ the wiring was setup using staples. Note that to avoid clutter of wires the lower part of 7 – segment display lines were connected on the upper half of breadboard.

The same applies for left and right buttons.



Finally then in picture ‘6’ the pin wires were connected.

This time one resistor (220R) was used.

Tests

Quick test has been done to ensure all the used input/output GPIO pins were working properly. In ‘section Raspberry Pi\test’ you may access the quick video test and the code as well.

Demos

You may access files with prefix name “3” and “4” in the ‘demos’ folder.