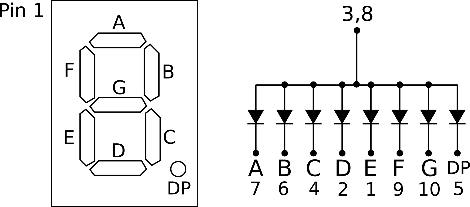
1. This circuit takes 4-bit binary input and has a 7-bit binary output. From the basic analysis, it seems like it has a 7-segment display. This circuit is known as **BCD *(Binary Coded Decimals)* to 7-segment display.** It lights up each LED segment whenever the output is ‘1’ and it gives out the decimal value. BCD: 4-bits to 16-bits (24 = 16), used for the 10 digits *0 to 9* to be displayed.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Display | D | C | B | A | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |



Example: Decimal value 0 has output abcdef ‘HIGH’.

HIGH = LED ONE

LOW = LED OFF

Although the output is recorded in binary, the 7-segment display shows decimal value.

1. Circuit A is a JK Flip Flop. It functions like a SR Flip Flop except when all the inputs are HIGH.

SR Flip Flop is best expressed using **OR gates** whereas JK Flip Flop is best described using **AND gate**

* DFF has one input (data) and 2 possible combo

When a positive rising edge, the value remembered by the FF becomes the value of the D input

* JKFF has 2 inputs and 4 possible combo

When positive rising edge:

- the value remembered by the FF toggles when J = 1 & K = 1.

- the value remains the same when J & K = 0.

- the value equals to K or J when J =1 & K = 0 or K = 1 & J = 0

* TFF functions with JKFF with same inputs

When the positive rising edge, the value remembered by FF either toggles or remains the same depending on whether the T inputs is 1 or 0

Truth Table for SR Flip Flop Truth table for JK Flip Flop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| clk | S | R | Output | |
| 0 | 0 | 0 | Memory | |
| 1 | 0 | 0 | Memory | |
| 1 | 1 | 0 | *Q* = 1 | = 0 |
| 1 | 0 | 1 | *Q* = 0 | = 1 |
| 1 | 1 | 1 | Not in use | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| clk | J | K | Output | |
| 0 | 0 | 0 | Memory | |
| 1 | 0 | 0 | Memory | |
| 1 | 1 | 0 | *Q* = 1 | = 0 |
| 1 | 0 | 1 | *Q* = 0 | = 1 |
| 1 | 1 | 1 | Toggles | |

*Notice the first 4 inputs to be similar! The difference occurs when all inputs are HIGH. JK Flip Flop, toggles its output*

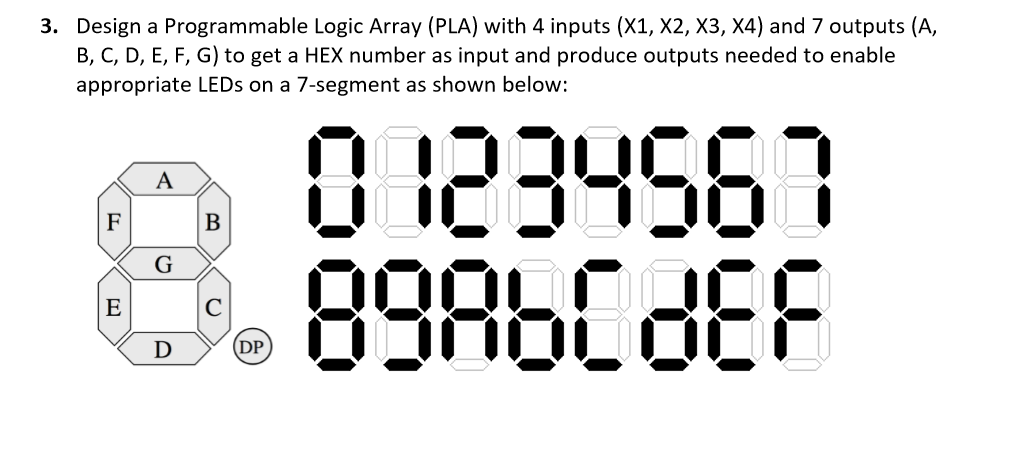
Truth Table for T Flip Flop

|  |  |  |
| --- | --- | --- |
| clk | T | Output |
| 0 | X | Memory |
| 1 | 0 | Memory |
| 1 | 1 | Toggles |

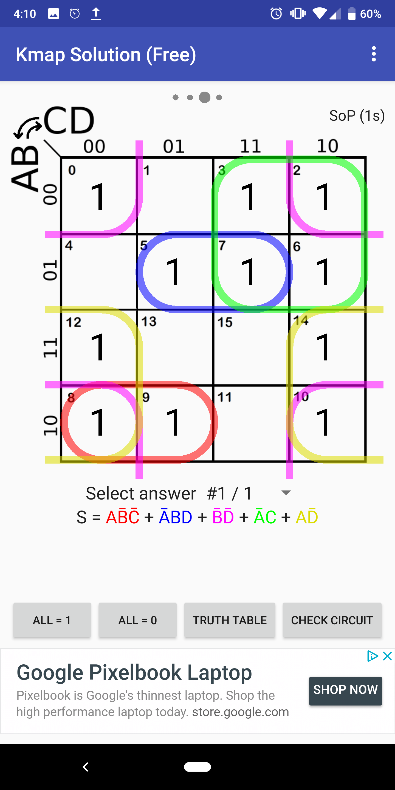
TFF uses a **XOR gate**

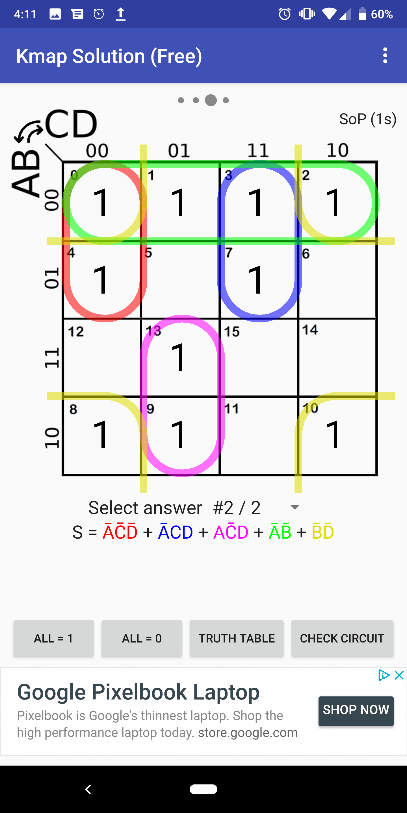
It is implemented using a JKFF – or at least it functions similarly as it.

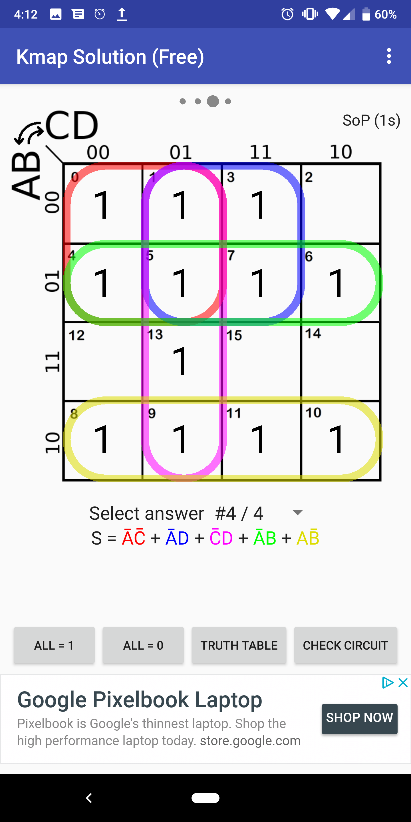
The output will depend on previous states.

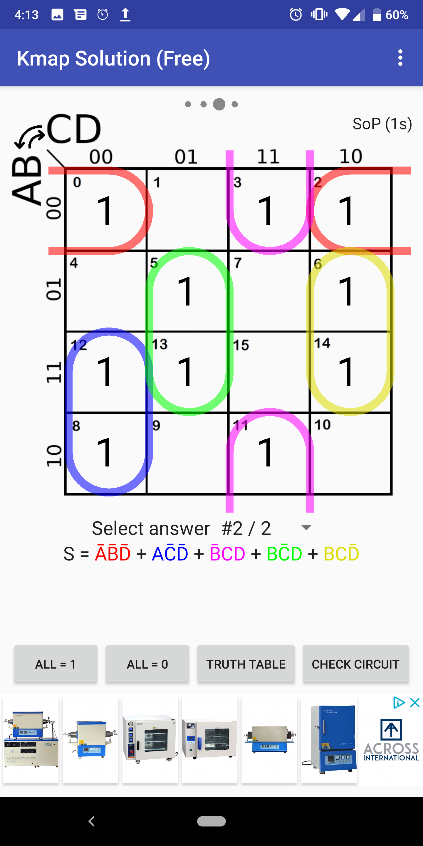
1. Truth table for the 7 segment display

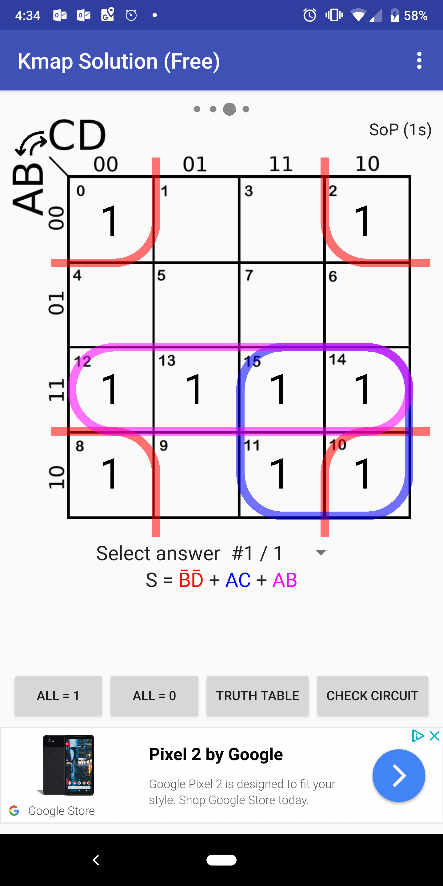
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Display | X1 | X2 | X3 | X4 | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| A | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| b | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| C | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| d | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| E | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| F | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

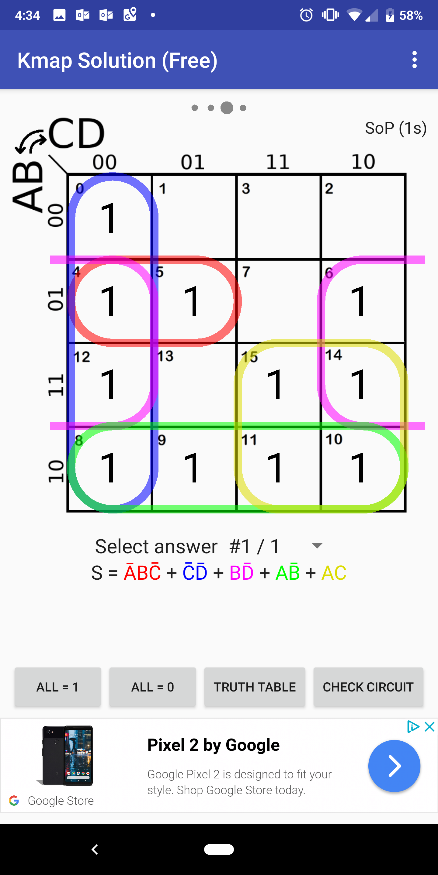


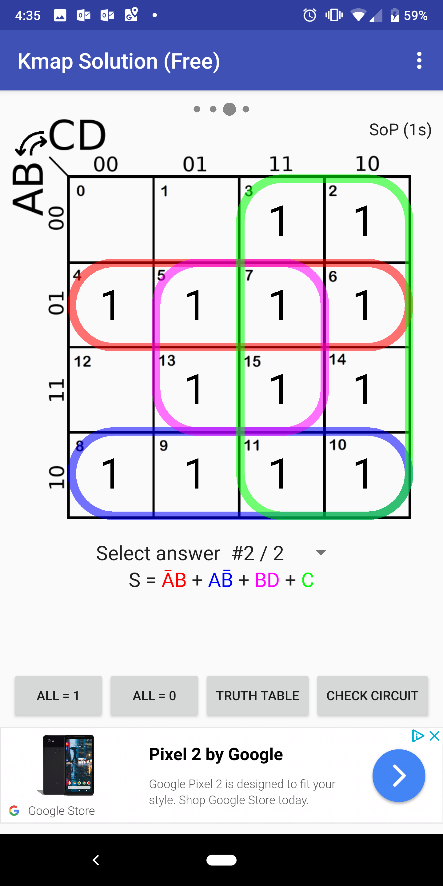


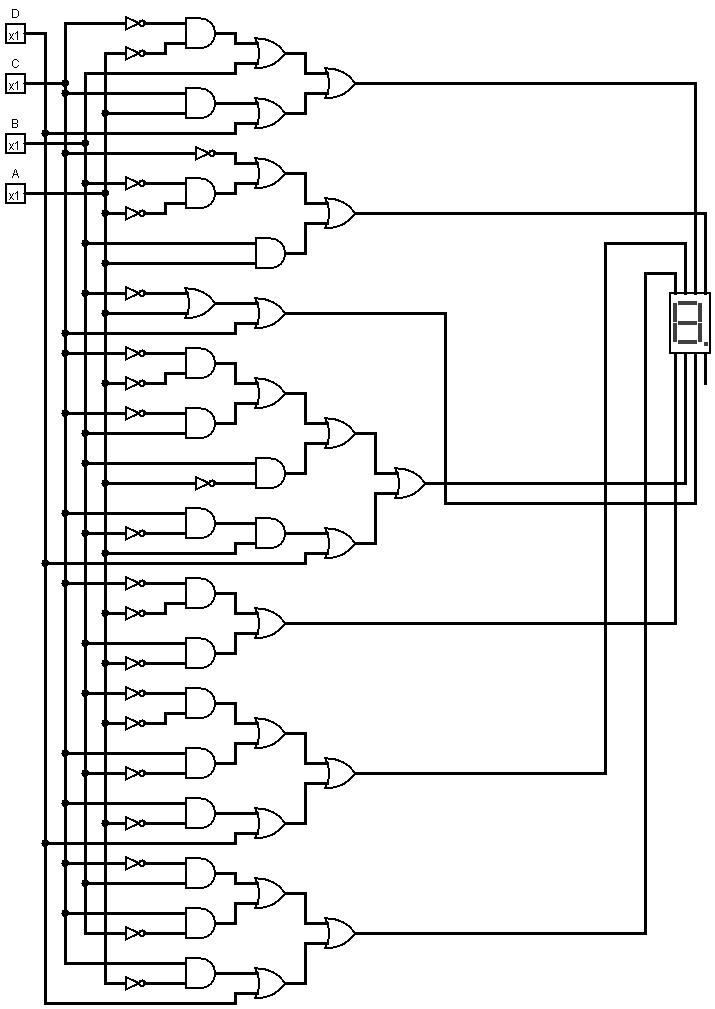












This was the closest PLA Design I could find that is an replica of each Boolean expression.