# Now that we have a counter, let’s do something with it.

# Problem 3

Design the set of equations that will drive a seven-segment LED display to show the numeric equivalent of the value of the counter’s present state. Consider state A to be the bit of higher significance.



From left to right, the seven-segment LED display should take on the appearance of the above schematic representations for 00, 01, 10 and 11, respectively. As with the counter, the display should show the number zero in the state following the one where it shows the number three.

Let’s look at a modified version of the state table. The design that we are about to perform doesn’t depend on a specific type of flip-flop, so we are going to leave out the excitation information for now.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Present State | | Present Output | | | | | | |
| A | B | a | b | c | d | e | f | g |
| 0 | 0 |  |  |  |  |  |  |  |
| 0 | 1 |  |  |  |  |  |  |  |
| 1 | 0 |  |  |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |  |  |

The columns for the seven-segment display are listed under a heading for the “Present Output.” This underscores an important distinction in what happens in any sequential circuit. As we saw in the previous examples, the next state results when we apply the present state to the excitation equations and clock the circuit. The desired next state occurs **after** the clock pulse. We will define the present output in terms of the present state. In this case, the outputs **a** through **g** must be such that they cause the proper digit **during** the appropriate clock period.

Don’t get confused and think that the segments must indicate what the circuit will do after the clock pulse. The present output indicates what the circuit is doing **now** because of the present state. The next state indicates what the circuit will do **following the clock pulse** because of the present state. Even though both are defined in terms of the present state, they represent different times in the operation of the circuit.

To complete the columns for the present output, we have to know what the seven-segment display should show during each present state. Remember, to light a segment, that segment’s input must be logic-0. If the segment is to remain unlit, the segment’s input must be logic 1. Here are the display diagrams again:



To form the digit 0, light segments a, b, c, d, e, and f. To form the digit 1, light segments b and c. To form the digit 2, light segments a, b, d, e, and g. To form the digit 3, light segments a, b, c, d, and g. This gives us the following completed truth table:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Present State | | Present Output | | | | | | |
| A | B | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

Maybe it’s trivial to form 2-variable Karnaugh maps to express the segment driver equations in terms of state variables A and B. Let’s do it anyway:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a |  | B | |  | b |  | B | |  | c |  | B | |  | d |  | B | |
|  |  | 0 | 1 |  |  |  | 0 | 1 |  |  |  | 0 | 1 |  |  |  | 0 | 1 |
| A | 0 | 0 | 1 |  | A | 0 | 0 | 0 |  | A | 0 | 0 | 0 |  | A | 0 | 0 | 1 |
| 1 | 0 | 0 |  | 1 | 0 | 0 |  | 1 | 1 | 0 |  | 1 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| e |  | B | |  | f |  | B | |  | g |  | B | |
|  |  | 0 | 1 |  |  |  | 0 | 1 |  |  |  | 0 | 1 |
| A | 0 | 0 | 1 |  | A | 0 | 0 | 1 |  | A | 0 | 1 | 1 |
| 1 | 0 | 1 |  | 1 | 1 | 1 |  | 1 | 0 | 0 |

We obtain the following set of equations:

a = A’B

b = 0

c = AB’

d = A’B

e = B

f = A + B

g = A’

Since we don’t ever want to see the dot, we can connect it to logic-1. Since a segment must “receive” logic-0 in order to be lit, tying the dot to logic-1 means that it will never be lit.

We can implement these equations in conjunction with the 2-bit counter that we designed in Problem 1. Again, these equations aren’t dependent on a particular type of flip-flop. They only require that the state sequence obtained by the counter match that of the truth table we made above.

**END OF PROBLEM 3**