# Problem 1

Design a 2-bit standard up counter. That is, a state machine that cycles through the states 00, 01, 10, 11.

# Design Process

Generally speaking, the process of sequential design begins with the creation of a state diagram that describes the state machine given in the specification. We have already seen how to create a state diagram from the information that is contained in this specification. However, since the number of states in this state machine is known, and the order of states is easily determined from the specification, the designer can proceed to the state table without creating a diagram.

**Step 1: Arrange the present states.**

Since a state table is a form of a truth table, we can begin by organizing the present state information. This step is quite simple, as we just have to put the information into the section labeled as “Present State.” Conceivably, the information could be arranged in any order, but we want to work with this as a truth table as much as possible. We might show the information as indicated below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Present State | | Next State | | D inputs | |
| A | B | A+ | B+ | DA | DB |
| 0 | 0 |  |  |  |  |
| 0 | 1 |  |  |  |  |
| 1 | 0 |  |  |  |  |
| 1 | 1 |  |  |  |  |

**Step 2: Arrange the next states.**

Since the specification is clear on how the states progress, we can fill in the known next state information into that section of the state table.

In looking at each present state (AB) and the next state that follows it (A+B+), the progression of states as described in the specification (00 – 01 – 10 – 11) should be apparent. In each case, when a given state is considered as the present state, then the state immediately following it must be the next state of the present state in question.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Present State | | Next State | | D inputs | |
| A | B | A+ | B+ | DA | DB |
| 0 | 0 | 0 | 1 |  |  |
| 0 | 1 | 1 | 0 |  |  |
| 1 | 0 | 1 | 1 |  |  |
| 1 | 1 | 0 | 0 |  |  |

Since the excitation information depends on knowing exactly what the present and next states are in all cases, it is very important to make sure that the specification (or state diagram) is interpreted correctly.

**Step 3: Obtain the excitation information.**

So that we can easily see which transitions are contained in the table, let’s add some shading to the state table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Present State | | Next State | | D inputs | |
| A | B | A+ | B+ | DA | DB |
| 0 | 0 | 0 | 1 |  |  |
| 0 | 1 | 1 | 0 |  |  |
| 1 | 0 | 1 | 1 |  |  |
| 1 | 1 | 0 | 0 |  |  |

All of the information connected with state A (or with flip-flop A) has been shaded dark gray, while all of the information connected with state B (or with flip-flop B) has been shaded light gray. When a particular transition is present for a given state, (A → A+, for example) the excitation information that causes the desired transition for that particular flip-flop will be recorded in the correct row and column.

Recall the excitation information for the D flip-flop:

|  |  |  |
| --- | --- | --- |
| Q | Q+ | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Even with this excitation information, we can concentrate less on the transitions and more on the final states, since we know that the next state of the D flip-flop following the clock pulse must be the same as the D flip-flop input prior to the clock pulse. That means that we can fill in the information for inputs DA and DB using the information provided in next states A+ and B+.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Present State | | Next State | | D inputs | |
| A | B | A+ | B+ | DA | DB |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |

**Step 4: Obtain the logic equations.**

Now that we have the excitation information for each flip-flop input, we can derive the equations that implement the counter. Taking the columns that correspond to the flip-flop inputs to be functions of the bits of the present state, we can create Karnaugh maps, just as we would for functions in any other truth table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DA |  | B | |  | DB |  | B | |
|  |  | 0 | 1 |  |  |  | 0 | 1 |
| A | 0 | 0 | 1 |  | A | 0 | 1 | 0 |
| 1 | 1 | 0 |  | 1 | 1 | 0 |

DA = A’B + AB’ = A ⊕ B

DB = B’

## END OF PROBLEM 1