## **Switching States**

Since we have eight states that we need to switch between each other, it is crucial that we extract the appropriate expressions for each flip flop.

As stated previously we are using three D flip flops which can be denoted by, A, B, and C.

We recall that we also have the following inputs:

The regular switch inputs:

The previous switch states (covered in separate module):

The timer variable: T (4 second timer)

### **The Boot State (000)**

When we are in the boot state, we only care for the regular switch inputs as they decide whether we go into the lock state or directly to the first sequence. We denote its state table below:

Table 1 - Boot State Table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **A** | **B** | **C** |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | Else | | | | 0 | 0 | 1 |

From this table we are able to extract the following expressions:

Since we are choosing to focus on the transition of the boot state we are obliged to AND these expressions with the current state expressions. Thus we obtain:

(A’B’C’)

(A’B’C’)

(A’B’C’)

### **The Sequence Detector (111)**

For the sequence detector its transitions purely rely on the previous switch state inputs. Its state table is constructed as follows:

Table 2 - Sequence Detector State Table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **A** | **B** | **C** |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

Since we are also confining our transitions within this state we must AND every single expression with it as well:

(ABC)

(ABC)

(ABC)

### **Sequence One (001)**

The only transition in sequence one is its transition to the sequence detector. This event is triggered when all the input switches are 0. When this happens the T timer goes high for 4 seconds, when it goes to 0 again, this is when we enter the sequence detector.

Table 3 - Sequence One State Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **T** | **A** | **B** | **C** |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |  | 1 | 1 | 1 |

**(A’B’C)**

**(A’B’C)**

**(A’B’C)**

### **Sequence Two (010)**

Sequence Two behaves exactly like sequence one and its equations are the same, except when we account for the transition focus.

Table 4 - Sequence Two State Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **T** | **A** | **B** | **C** |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |  | 1 | 1 | 1 |

**(A’BC’)**

**(A’BC’)**

**(A’BC’)**

### **Sequence Four (100)**

Similarly for sequence four, the equations are the same as sequence one and two.

Table 5 - Sequence Four State Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **T** | **A** | **B** | **C** |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |  | 1 | 1 | 1 |

**(AB’C’)**

**(AB’C’)**

**(AB’C’)**

### **Sequence Three (011)**

Although sequence three inherently behaves like the three previous states, it has an extra transition. This transition is to the special trick state. It occurs when two or more switches are on.

In order to study and extract the appropriate expressions for this state we need to cover all the permutations while ignoring T. We need to be careful that when there are less than two lamps on, we must stay in sequence three. Only when more than two lamps are on do we transition to the special state.

Table 6 - Sequence Three State Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **T** | **A** | **B** | **C** |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |  | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | X | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | X | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | X | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | X | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | X | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | X | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | X | 1 | 0 | 1 |

**(AB’C’)** + ( + + + + +

**(AB’C’)** + ( + + + + +

**(A’BC)**

### **Special Trick (101)**

To construct the table for this state we must denote new inputs which come from the special trick module itself.

The enable inputs:

These inputs determine whether a switch is disabled or not. We enter and leave this state with these inputs always as (1111). We are not allowed to leave the special trick state unless the switch is reenabled.

Thus when we enter this state, we go back to sequence three only when all the enable inputs are back to normal and the timer cool down ended.

Table 7 - Special Trick State Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current** | | | **Inputs** | | | | | **Next** | | |
| **A** | **B** | **C** |  |  |  |  | **T** | **A** | **B** | **C** |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |  | 0 | 1 | 1 |

)’**(AB’C)**

**(AB’C)**

**(AB’C)**

### **Multiplexing**

We now have to combine all these formulas into one, however we notice that since we decided to focus the transitions on each state, we can simply use an 8:1 MUX like with the output multiplexing to select which equation should be applied to the flip flops.