|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Present State | | | | Input | Next State | | | | FF Input Equation | | | |
| At | Bt | Ct | Dt | X | At+1 | Bt+1 | Ct+1 | Dt+1 | TA | TB | TC | TD |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | x | x | x | x | x | x | x | x |
| 1 | 0 | 1 | 0 | 1 | x | x | x | x | x | x | x | x |
| 1 | 0 | 1 | 1 | 0 | x | x | x | x | x | x | x | x |
| 1 | 0 | 1 | 1 | 1 | x | x | x | x | x | x | x | x |

**State Diagram:**

**State Assignment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | Binary Value Assignment | | | |
| C | 0 | 0 | 0 | 0 |
| S | 0 | 0 | 0 | 1 |
| E | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| - | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| - | 1 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 1 |

**4 bit Asynchronous Up/Down Counter:**

This counter will loop through the states from C (0000) to 7(1001) and vice versa upon the switch condition. When the switch is 1 (up situation), the counter will traverse through a loop as follows :

When the switch is 0 (down condition), the loop will traverse in reverse direction.

We have used 4 T Flip-Flops in the up/down counter. There is one switch to give inputs to the flip flops and one clock to give pulses after certain time periods.

**ROM implementation using 16:1 multiplexer:**

We have used 3 16:1 multiplexers as a system to store the data input combination from the 4 bit asynchronous up/down counter. This is a trick similar to using ROM to store number of words for certain input lines. Even though a ROM is *not a combinational logic circuit*. A combinational logic circuit does not have memory, while a ROM, by its very name, does.

Read-only memories allows to read stored memory, but not modify the memory. The idea is to initialize the memory once, and then use it as a kind of lookup table from that point onward.

Once stored in the ROM, the binary data can be read, but cannot be modified (under normal operating conditions) . A ROM implements (i.e. stores) the truth table of a function (or of several functions).

We have used 16:1 multiplexers so that we can accommodate the combination of our 10 states from the circuit diagram. And also these multiplexers have 4 select bits that match with the 4 bit up/down counter from where the combinations are coming in.

MUX truth table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Select bits | | | | Input | Output |
| S3 | S2 | S3 | S4 | X | F |
| 0 | 0 | 0 | 0 | 0 | I0 =0 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | I1 =0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | I2 =0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | I3 =0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | I4 =1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | I5 =1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | I6 =1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | I7 =1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | I8 =1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | I9 =1 |
| 1 | 0 | 0 | 1 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Select bits | | | | Input | Output |
| S3 | S2 | S3 | S4 | X | F |
| 0 | 0 | 0 | 0 | 0 | I0 =0 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | I1 =0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | I2 =1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | I3 =1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | I4 =0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | I5 =0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | I6 =1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | I7 =0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | I8 =1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | I9 =1 |
| 1 | 0 | 0 | 1 | 1 |

Truth table of MUX 1 Truth table of MUX 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Select bits | | | | Input | Output |
| S3 | S2 | S3 | S4 | X | F |
| 0 | 0 | 0 | 0 | 0 | I0 =0 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | I1 =1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | I2 =0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | I3 =1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | I4 =0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | I5 =1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | I6 =0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | I7 =1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | I8 =0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | I9 =1 |
| 1 | 0 | 0 | 1 | 1 |

Truth Table of MUX 3

From these MUX, the outputs are connected to the previous combinational circuit we prepared. So we replace the switches of combinational circuit by the output wires of Mux. Note that, we had a 3 bit combinational circuit and also now, we have 3 outputs from the MUXs so our combinational part will work fine.