Methodology for Designing FSMs

© Prof. Seda Ogrenci

Designing a Finite State machine can be described with the following steps:

1. Analyze the verbal problem statement
2. Create example input/output sequences that represent typical scenarios conforming to the stated behavior
3. Going through the sequences,
    a. Either create a new state and branch out for the corresponding input case
    b. Revisit an existing state that covers this input behavior
4. For each state, transitions for all possible input combinations must be accounted for
5. For each possible transition, the output response must be attached

Let us examine each step through an example. I will walk you through the design of a pattern detection FSM and we will review how to design a Moore and a Mealy style FSM.

Problem Statement: Design a Finite State Machine that receives a stream of bits and outputs '1' if it receives a 01 or a 10 pattern. The FSM should output '0' for all other cases.

STEP 1. Analysis: What should the FSM do for cases, such as "0 1 0" or "1 0 1"? Since there are no specific instructions to the contrary, we can assume that the FSM should interpret them as both patterns being detected (i.e., overlaps will be allowed).

STEP 2. Example input output/output sequence: Often times, there will not be any specific guidelines for creating a sample sequence. This is based on the designer's discretion. As a general rule, a sample sequence should incorporate all possible scenarios.

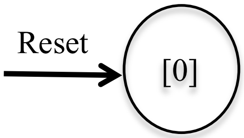
INPUT: 0 0 0 1 1 1 0 1 0 0

OUTPUT: 0 0 0 1 0 0 1 1 1 0

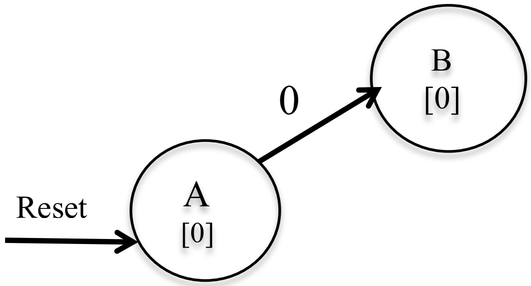
STEP 3. Creation of states: I will illustrate this steps twice, once for the Moore FSM and once for the Mealy FSM, so that you can observe how these two styles lead to two different implementations:

<u>**Moore Style FSM:**</u> In this implementation states are solely responsible for generating the outputs, hence, outputs will be associated with each current state.

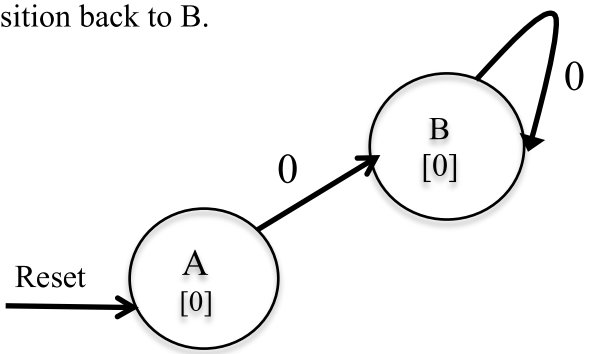
We always start by inserting an initial (Reset) state. While in this state, the detector should output '0':



Now let's step through the input sequence and determine what transitions will be needed from this state. The FIRST bit arrives and it is '0'. This might be the **first bit of a potential "01" pattern**. The FSM needs to **"remember" this scenario**, **by creating a new state**. We insert a second state and add a transition to this state for input '0'. The output to be generated by this new state should still be '0'.

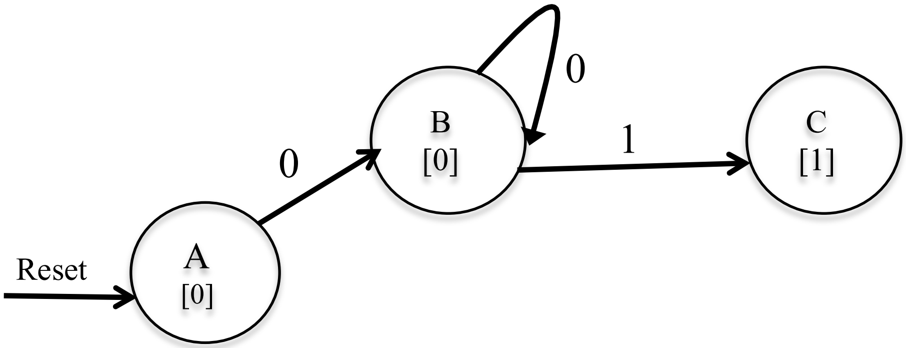


The SECOND input is '0', that means the 01 pattern did not occur. The very next bit might still be '1', hence the situation is identical to what was evaluated a clock cycle ago. Hence upon the encounter of this second '0' the FSM should remain in STATE B. We insert a self-looping transition back to B.

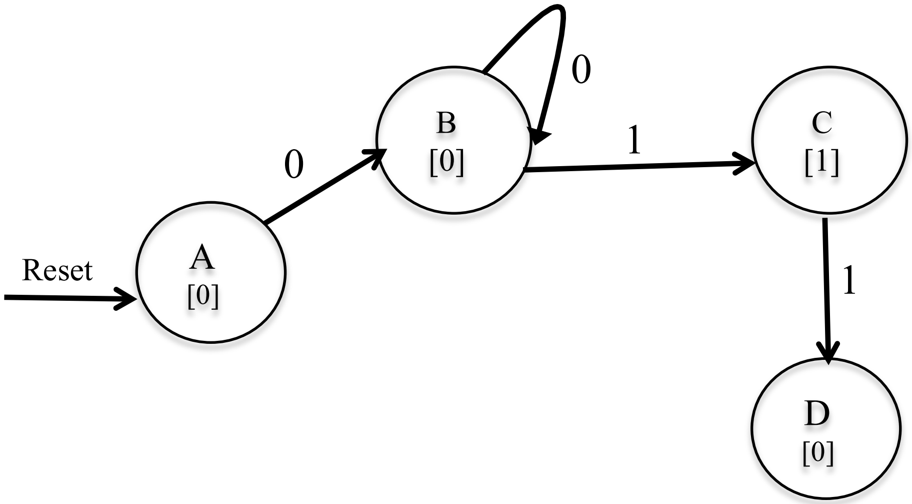


The THIRD input is '0'. Same reasoning applies. FSM should remain in STATE B, generating '0' output.

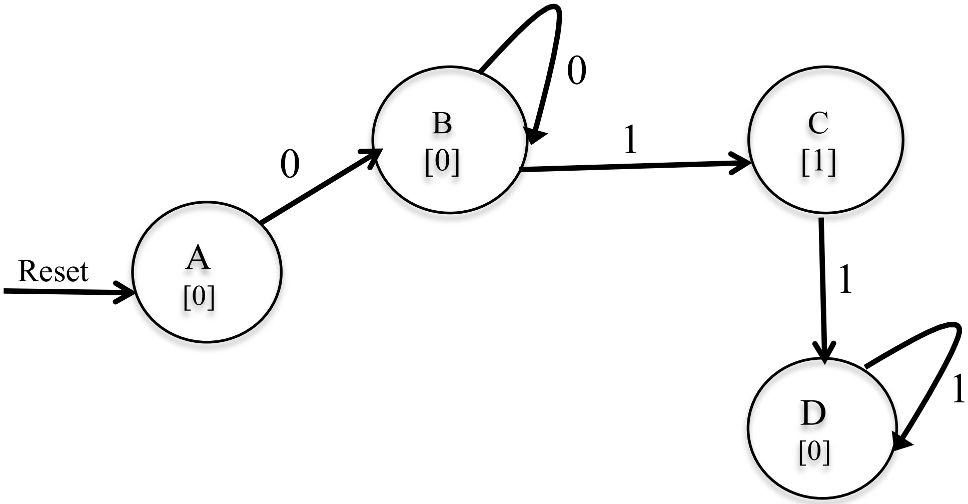
The FOURTH input is '1'. A "01" pattern will be detected. **This is a new event, i.e., the detection of "01" pattern. The FSM will remember this event by creating a new state** for this. We insert STATE C. When the FSM enters STATE C, the output should now be '1'.



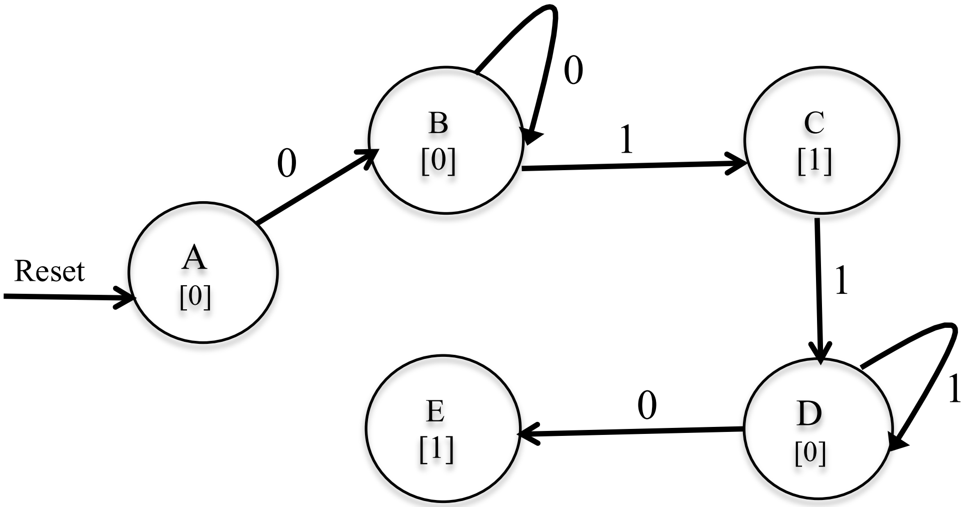
The FIFTH input is '1'. It may be **the precursor of a potential "10" pattern. This is a new event that FSM must remember**. Technically, there is no difference between receiving a '1' input after having received a '0' or a '1'. However, we need a state that should respond with a '0' output in this particular case. Even though, STATE C also signifies a '1' input being observed (albeit it generating a 01 pattern), C cannot cover this new role. Because STATE C is tasked with generating **a '1' output**. **Now, we need a state that should generate a '0'**. **Hence, we need a new state** to represent this case. A, B, C are not capable to representing it. So, we create STATE D and transition from C to D for input '1'. The output of STATE D should be '0'.



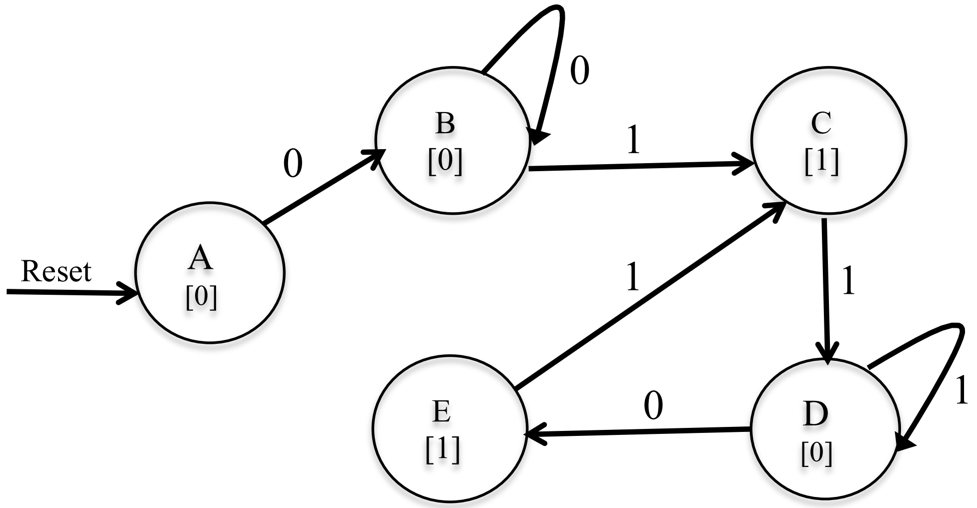
The SIXTH input is ‘1’, this maps to the same case as the previous one. A ‘1’ has been observed, it might be the start of a “10’ pattern. This still maps to STATE D. We insert a self-looping transition.



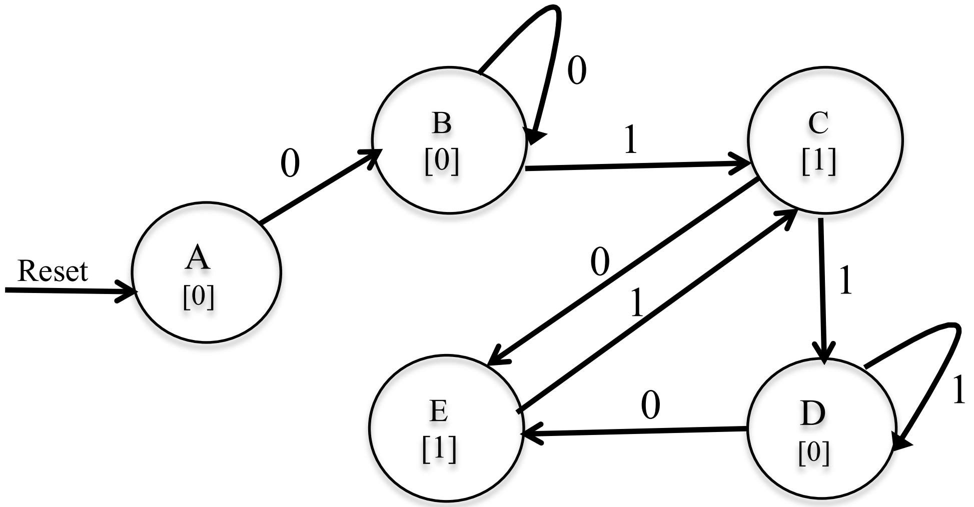
THE SEVENTH input is ‘0’, **this creates a “10” pattern. This is a new event that FSM must “remember” by inserting a new STATE E**. A transition from STATE D to STATE E will occur. STATE E should generate a ‘1’ output.



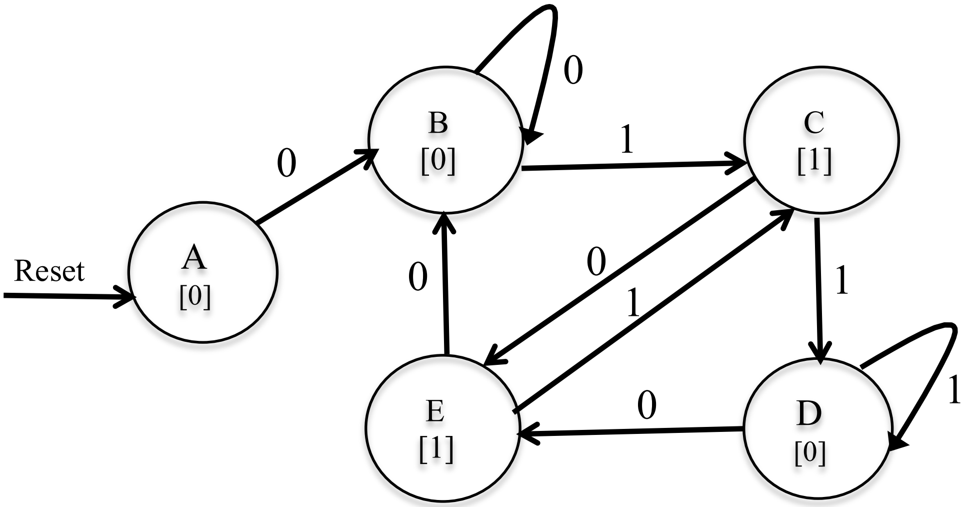
THE EIGHTH input is ‘1’, this creates a “01” pattern (with an overlap with the previous “10” pattern, which we agreed to allow). The FSM already has a state in place to “remember” this case, which is STATE C. Hence, there is no need for a new state. Only a new transition from STATE E to STATE C will be inserted for input ‘1’.



THE NINTH input is ‘0’, this creates another “10” pattern. The FSM should transition back to STATE E, which is in charge of representing the event of detecting “10”. A transition from STATE C to STATE E for input ‘0’ will be inserted.

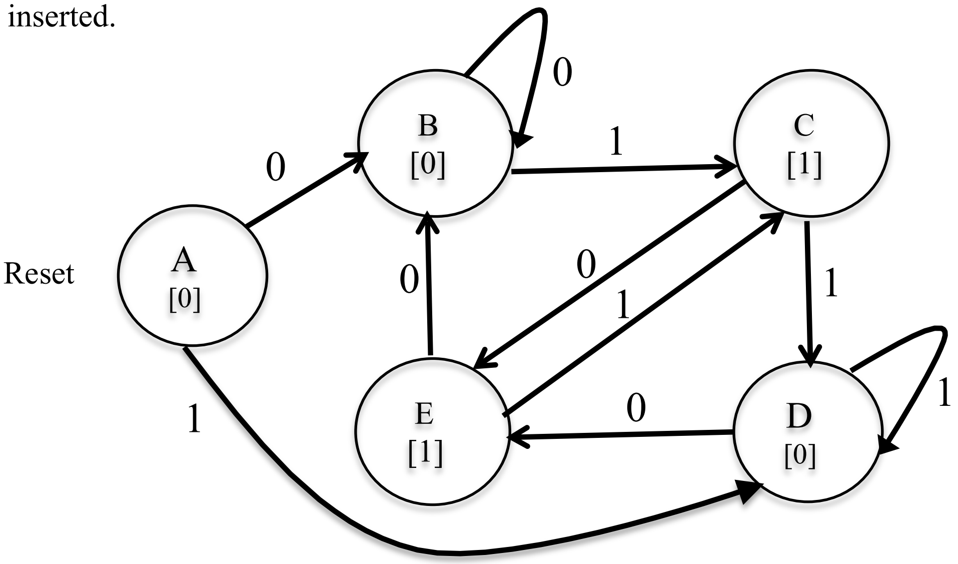


THE TENTH input is ‘0’. This might be the start of a new “01” pattern. Recall that, STATE B serves the purpose of helping the FSM remember this particular scenario. Hence, a transition from STATE E back to STATE B will be inserted.



We can observe that all possible scenarios have been accounted for and there are states in place to help the FSM “remember” all relevant events in this behavior. Now, we move onto Step 4 and complete missing transitions.

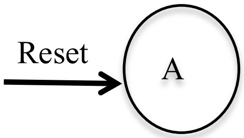
STEP 4. All incoming and outgoing transitions of all states must be accounted for: Since this FSM only has a single bit input. We need to make sure that each state has TWO outgoing transitions, one for input being ‘0’ and one for input being ‘1’. STATES B, C, D, and E are already complete in this regard. So, we check STATE A. Its outgoing transition for when the input is ‘1’ is missing. When the first bit arriving after Reset is 1, this is equivalent to the case of having encountered a ‘1’, which might potentially be the start of a “10” pattern. STATE D represents this case. Hence, a transition from A to D will be inserted.



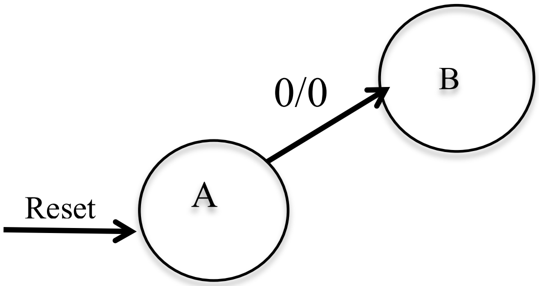
STEP 5. Review output response: Since, this is a Moore style FSM, we have accounted for all output responses by associating them with individual states

<u>**Mealy Style FSM:**</u> In this implementation the output behavior of the FSM will be associated with transitions. Thereby, they will become a function of both the inputs and the states, which the transitions are originating from.

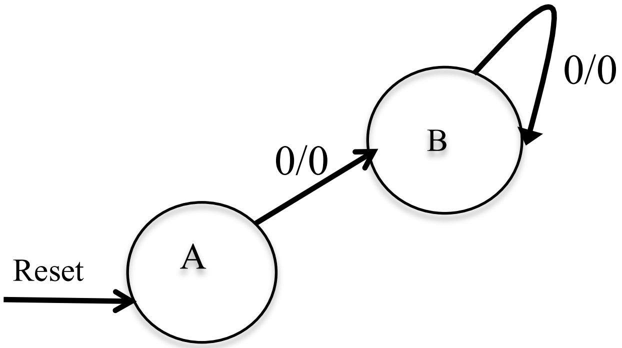
We always start by inserting an initial (Reset) state.



Now let's step through the input sequence again. The FIRST bit arrives and it is '0'. This might be the **first bit of a potential "01" pattern**. The FSM needs to **"remember" this scenario**, **by creating a new state**. We insert a second state B and add a transition to this state for input '0' generating an output '0'.

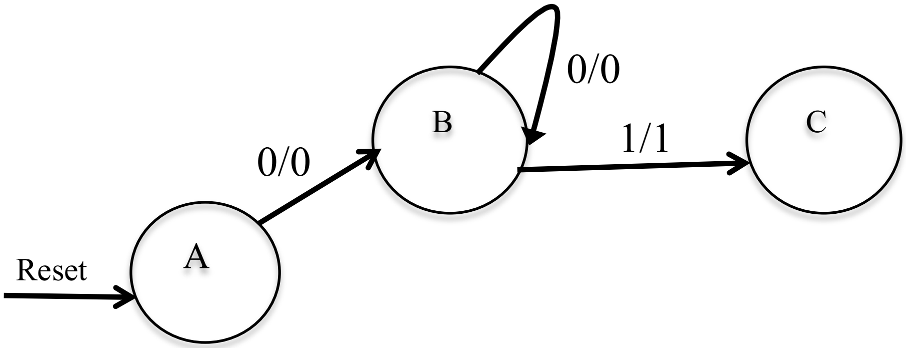


The SECOND input is '0', that means the 01 pattern did not occur. The very next bit might still be '1', hence the situation is identical to what was evaluated a clock cycle ago. Hence upon the encounter of this second '0' the FSM should remain in STATE B. We insert a self-looping transition back to B with an output of '0'.

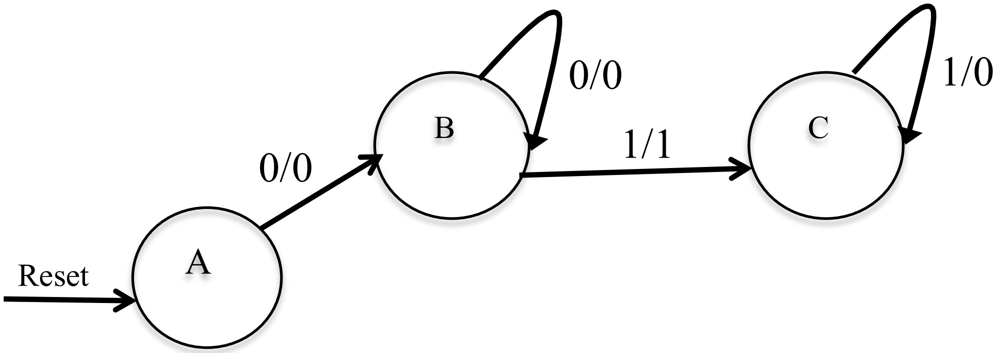


The THIRD input is '0'. Same reasoning applies. FSM should remain in STATE B.

The FOURTH input is ‘1’. A “01” pattern will be detected. **This is a new event, i.e., the detection of the “01” pattern. The FSM will remember this event by creating a new state** for this. We insert STATE C. The transition from B to C under the input ‘1’ will generate output ‘1’.

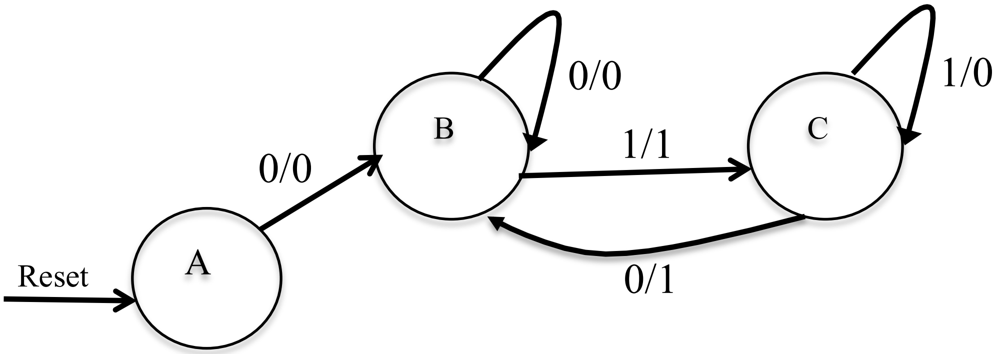


The FIFTH input is ‘1’. It may be **the precursor of a potential “10” pattern.** Yet at the same time, in its essence it is no different that having encountered the ‘1’ input in the previous cycle (FOURTH INPUT). Hence, we can still use STATE C to denote the arrival of this ‘1’ input. However, we need a new transition (a self-looping transitions to keep us in STATE C) to generate the ‘0’ output that we need now.



The SIXTH input is ‘1’, this maps to the same case as the previous one. A ‘1’ input has been observed, it might still be the start of a “10’ pattern. This still maps to STATE C.

THE SEVENTH input is ‘0’, **this creates a “10” pattern. While it hold special importance since it signals a pattern, as far as the most recent bit being observed, it is the same as having observed the very first ‘0’ (accounted for by STATE B). The differentiating factor will be a transition from C back to B to generate a ‘1’ output.**



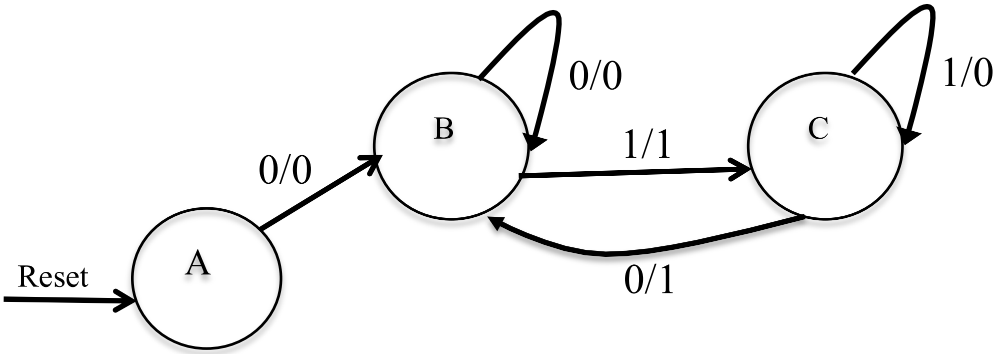
THE EIGHTH input is ‘1’, this creates a “01” pattern (with an overlap with the previous “10” pattern, which we agreed to allow). The FSM already has a transition (from B to C under input=’1’) in place to “remember” this case, bringing the FSM to STATE C.

THE NINTH input is ‘0’, this creates another “10” pattern. The FSM will utilize the transition from STATE C to STATE B under input=’0’ generating a ‘1’ output.

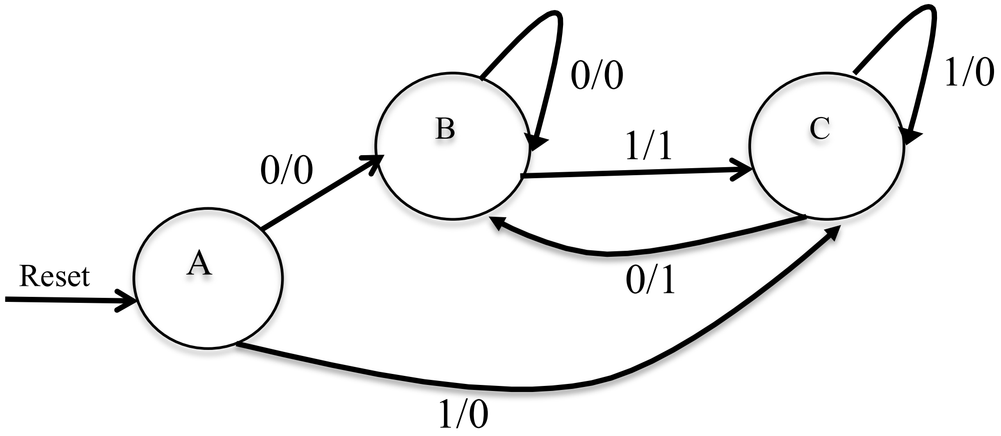
THE TENTH input is ‘0’. This might be the start of a new “01” pattern. Recall that, STATE B serves the purpose of helping the FSM remember this particular scenario. The self-looping transition at STATE B under input=’0’ will generate output ‘0’.

We can observe that all possible scenarios have been accounted for and there are states and transitions in place to help the FSM “remember” all relevant events in this behavior. Now, we move onto Step 4 and complete any missing transitions.

STEP 4. All incoming and outgoing transitions of all states must be accounted for: Since this FSM only has a single bit input. We need to make sure that each state has TWO outgoing transitions, one for input being ‘0’ and one for input being ‘1’. STATES B, C, D, and E are already complete in this regard. So, we check STATE A. Its outgoing transition for when the input is ‘1’ is missing. When the first bit arriving after Reset is 1, this is equivalent to the case of having encountered a ‘1’, which might potentially be the start of a “10” pattern. STATE D represents this case. Hence, a transition from A to D will be inserted.



STATES B and C have two outgoing transitions each already. The outgoing transition scenario for STATE A under input ‘1’ is missing. After the Reset, if the first bit to arrive is ‘1’, this will be the same as having observed ‘1’ under any other circumstance. Hence, this should trigger a transition to STATE C, generating an output ‘0’.



STEP 5. Review output response: Since, this is a Mealy style FSM, we have accounted for all output responses by associating them with transitions.