string 1010 is successfully extracted, the final "tail" indicator is asserted, indicating the completion of the data package reception.

## Exercises

**Problem 7.1.** Design a 101 sequence detector utilizing both Mealy and Moore FSMs. As depicted in Figure 7.11, the sequence detector will examine a string of 0's and 1's provided to the input "x" and produce an output "y=1" only when the input sequence concludes with the pattern 101.

For instance, Figure 7.32 visually demonstrates the operation of the sequence detector. It processes serial data on the "x" input in a left-to-right fashion, handling each digit per clock cycle. Notably, there can be an overlap between digits in two consecutive instances of the desired 101 string. For example, in the second and third occurrences of the 101 string, the last digit "1" from the second occurrence serves as the first digit "1" of the third instance.

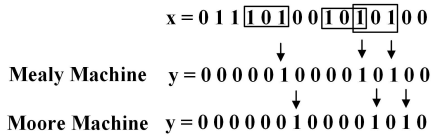


**FIGURE 7.32**
101 Sequence Detection Utilizing Mealy and Moore FSM

Perform the following four steps for both Mealy FSM and Moore FSM designs:
1) Build a state graph.
2) Generate the state table and encode it into a transition table.
3) Create block diagrams illustrating the expected hardware results, depicting the sequential registers and combinational circuits for both output and next state. Evaluate the combinational circuits using K-maps to optimize the logic.
4) Analyze the simulation results using the input stimulus shown in Figure 7.33. The input signals "x" and the asynchronous reset "rst" are provided. Draw the current state "cur_state" and the next state "nxt_state", as well as the output "y".

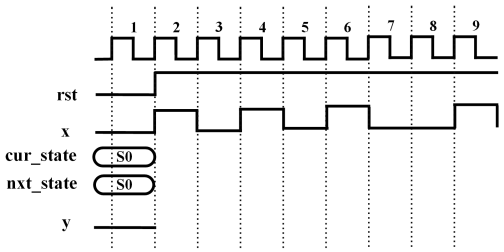


**FIGURE 7.33**
Simulation Results Analysis

## PBL 17: Sequence Detector

1) Design a 101 sequence detector with Verilog HDL, utilizing both Mealy and Moore FSMs. Table 7.2 provides an overview of the design's IOs. In addition to the input "x" and output "y", the state machine necessitates an asynchronous reset for initializing the state register and a positive edge-triggered clock for sampling each state.

**TABLE 7.2**
Sequence Detector IOs Description

| Name | Direction | Bit Width | Description |
|---|---|---|---|
| clk | Input | 1 | Clock, rising edge trigger |
| rst | Input | 1 | Asynchronous reset, 0 valid |
| x | Input | 1 | Digit string input |
| y | Output | 1 | Output 1 when 101 string detected |

2) Create a testbench to simulate the design, following the input stimulus depicted in Figure 7.33. Compare the simulation outcomes from ModelSim with the analysis results from Problem 7.1.
3) Execute synthesis and implementation in Vivado. Compare the RTL Analysis results obtained in Vivado with the hardware diagrams generated in Problem 7.1.