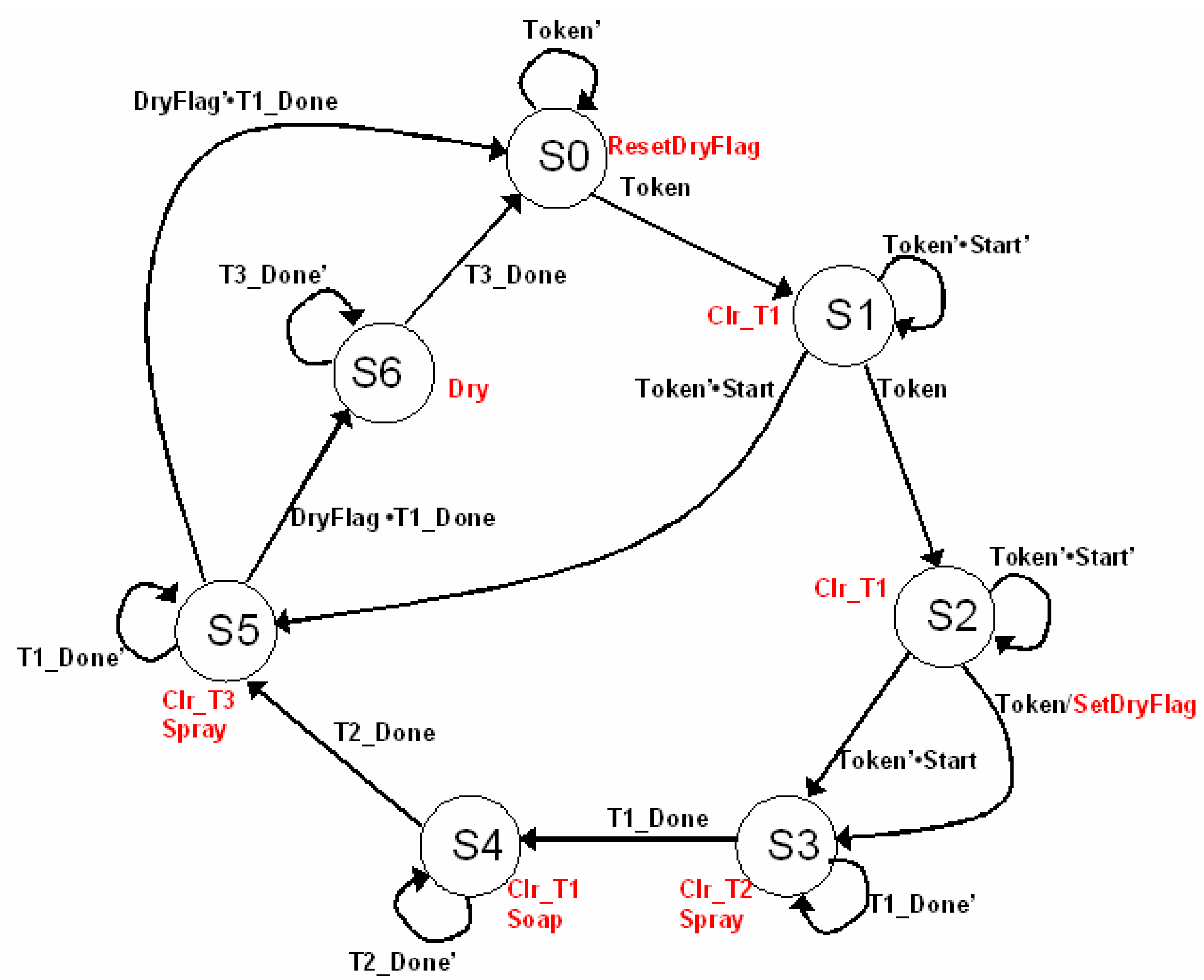
ECEn/CS 224
Chapter 16 Homework Solutions

16.1 Draw the state graph for a car wash controller which has 3 different kinds of washes - a simple rinse-only wash, a rinse-soap-rinse wash, and a rinse-soap-rinse-blow dry wash. The first two of these wash types are the same as those in this chapter. State any assumptions you make as you create the state graph.

Method 1

One way to solve this problem is to assume we have a separate piece of logic that can tell us if we will need to dry after the last rinse cycle. A state diagram which takes advantage of this logic is shown below. A list of our assumptions follows.



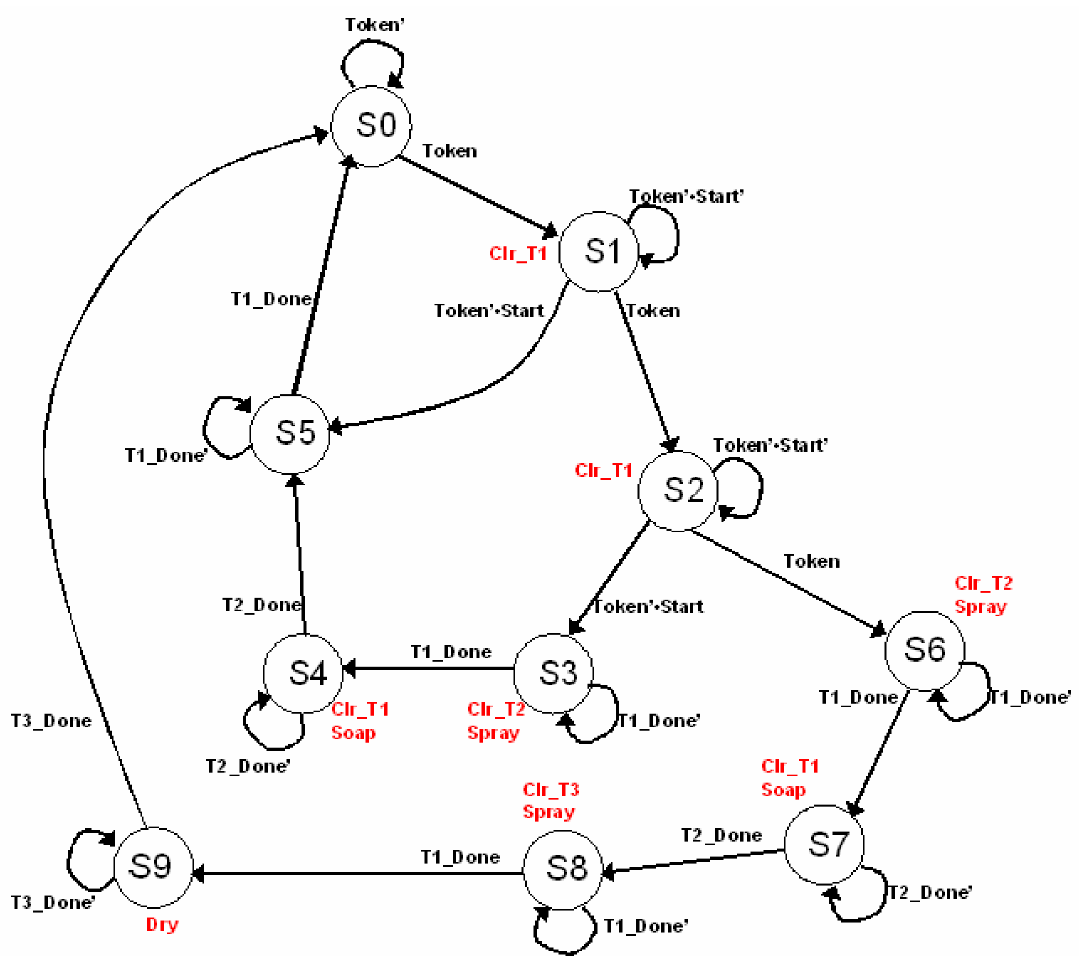
Inputs: Token, Start, T1_Done, T2_Done, T3_Done, DryFlag
Outputs: Clr_T1, Clr_T2, Clr_T3, Spray, Soap, Dry, SetDryFlag, ResetDryFlag

Assumptions:

1) The Spray signal turns on the Spray function
2) The Soap signal turns on the Soap function
3) The Dry output signal turns on the Blow Dry function
4) Timer 1 (T1) is used to time the Spray function
5) Timer 2 (T2) is used to time the Soap function
6) Timer 3 (T3) is used to time the Blow Dry Function
7) The SetDryFlag signal sets a register which indicates if we will need to Blow Dry at the end of the wash
8) The ResetDryFlag signal resets the register which indicates if we will need to Blow Dry at the end of the wash

Method 2

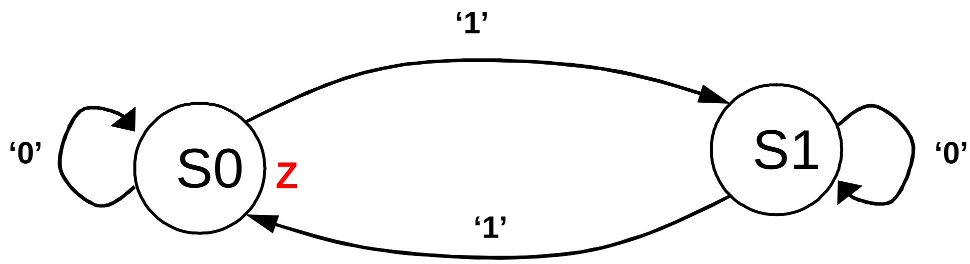
A second approach is to use a separate line of states for the third was type, which requires the blow dry at the end of the wash. A state diagram for this method is shown below. A list of our assumptions follows.



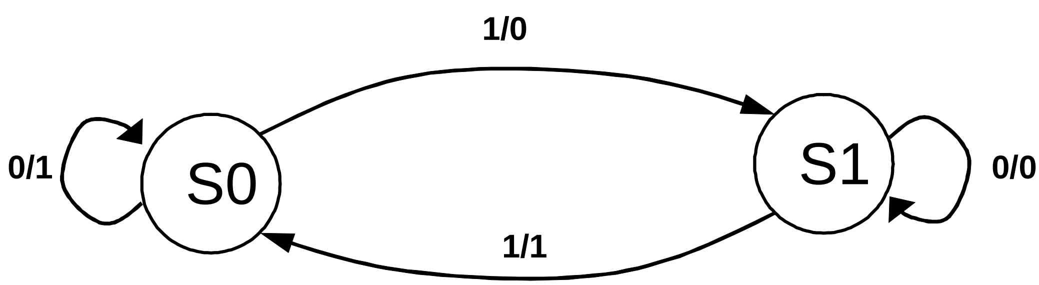
Inputs: Token, Start, T1_Done, T2_Done, T3_Done
Outputs: Clr_T1, Clr_T2, Clr_T3, Spray, Soap, Dry

Assumptions:
1) The Spray signal turns on the Spray function
2) The Soap signal turns on the Soap function
3) The Dry output signal turns on the Blow Dry function
4) Timer 1 (T1) is used to time the Spray function
5) Timer 2 (T2) is used to time the Soap function
6) Timer 3 (T3) is used to time the Blow Dry Function

16.2 Draw the state graph for a state machine which has a single input. Have it output a signal which indicates whether an even number of 1's has been seen this far in the input stream (no 1's seen so far counts as even). Use a Moore output. Use a minimum number of states.

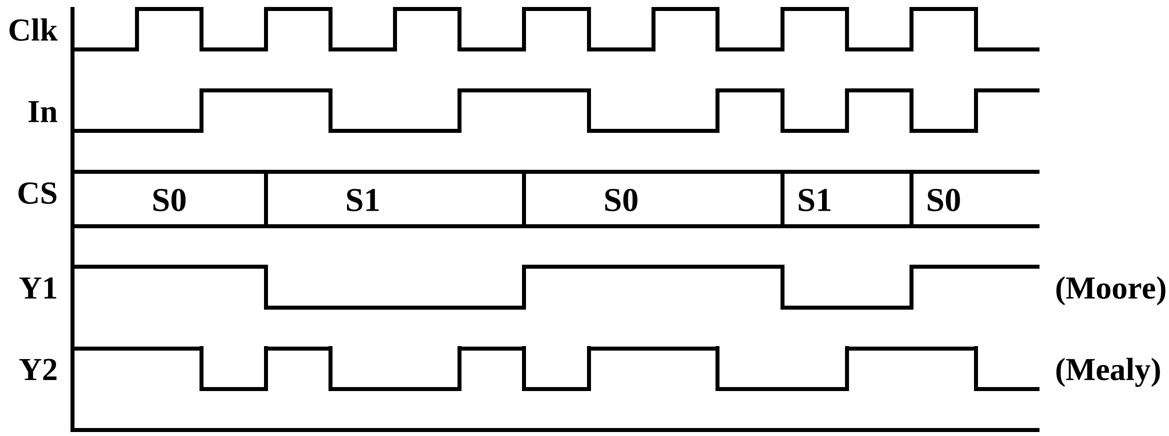


16.3 Repeat the previous problem, but with a Mealy output. Use a minimum number of states.



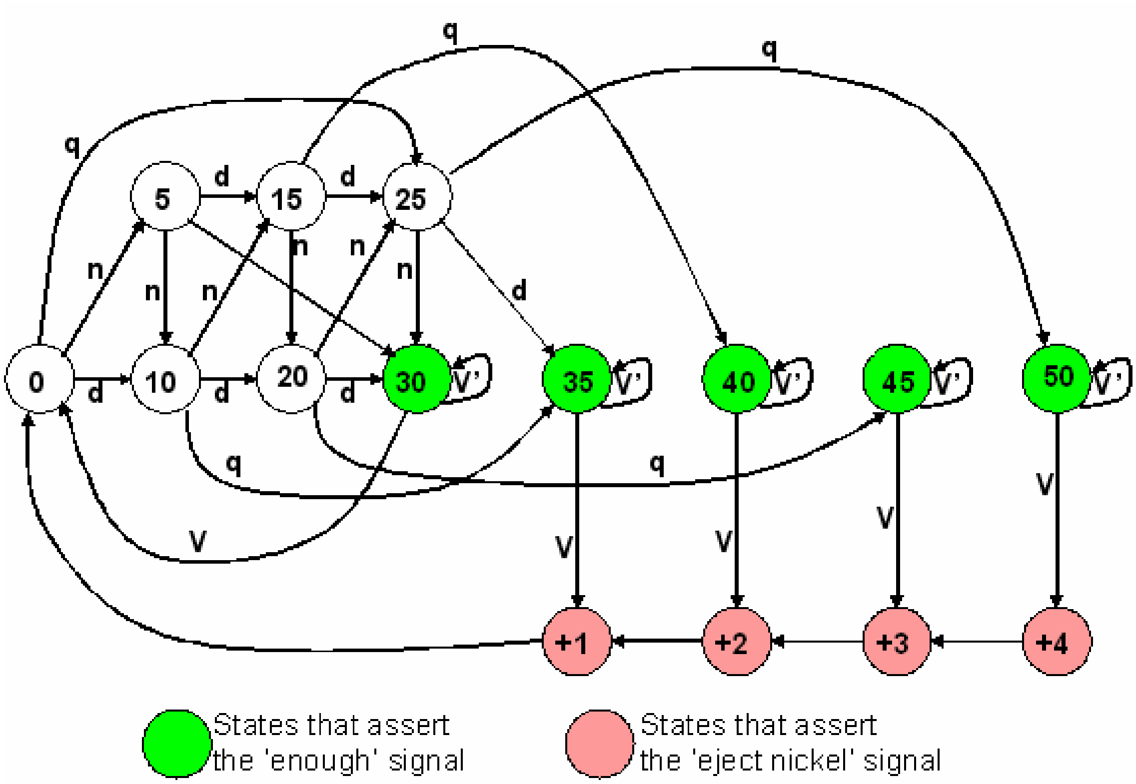
16.4 Draw a timing diagram showing how the previous two state machines behave differently in response to a sequence of inputs.

A sample diagram is shown below, assuming rising edge triggered flip flops.



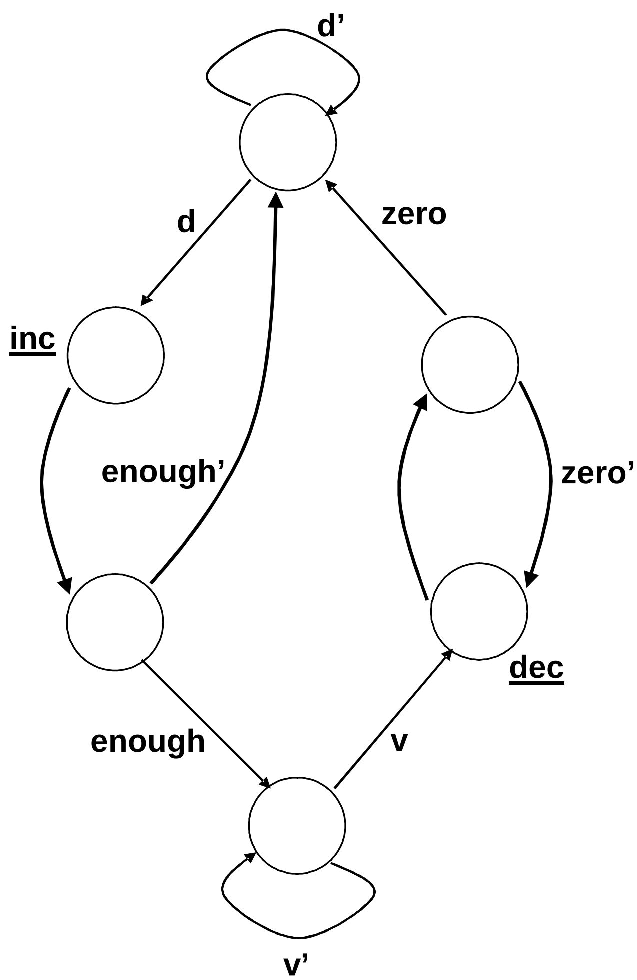
16.5 See the text for a complete problem description.

The state diagram below is one possible solution to this problem. Alternate solutions exist. The states S0, S5, S10, S15, S20, and S25 each have a path back to themselves which is not shown. These paths are taken when there are no coin inputs (i.e. n'•d'•q').

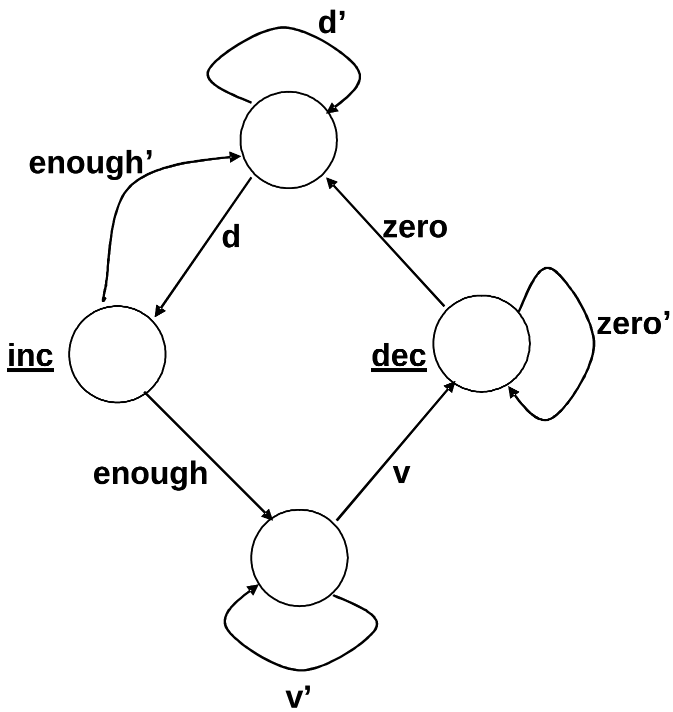


16.6 See the text for a complete problem description.

One relevant piece of information not stated in the problem description is whether or not the accumulator provided is synchronous or asynchronous. Most likely it is synchronous, running of the same clock as our state machine. If it synchronous then we need more states (or Mealy outputs) in our state machine since we cannot increment and check the **enough** signal in the same state, since the increment will not occur until the next clock edge. A state diagram, assuming a synchronous accumulator, is shown below. Other solutions exist.



An asynchronous accumulator would require that the state machine logic driving the **inc** and **dec** inputs be hazard free. This requires careful choice of state encoding and hazard free logic design. A state diagram assuming an asynchronous accumulator is shown below.


d’
enough’
d
zero
zero’
inc
dec
enough
v
v’