The problem is taken from the “The Book Digital Design and Computer Architecture” Second Edition
David Money Harris Sarah L. Harris

looking where they are going. Football players are hustling between the athletic fields and the dining hall on Bravado Boulevard. They are tossing the ball back and forth and aren't looking where they are going either. Several serious injuries have already occurred at the intersection of these two roads, and the Dean of Students asks Ben Bitdiddle to install a traffic light before there are fatalities.

Ben decides to solve the problem with an FSM. He installs two traffic sensors, T_A and T_B, on Academic Ave. and Bravado Blvd., respectively. Each sensor indicates TRUE if students are present and FALSE if the street is empty. He also installs two traffic lights, L_A and L_B, to control traffic. Each light receives digital inputs specifying whether it should be green, yellow, or red. Hence, his FSM has two inputs, T_A and T_B, and two outputs, L_A and L_B. The intersection with lights and sensors is shown in Figure 3.23. Ben provides a clock with a 5-second period. On each clock tick (rising edge), the lights may change based on the traffic sensors. He also provides a reset button so that Physical Plant technicians can put the controller in a known initial state when they turn it on. Figure 3.24 shows a black box view of the state machine.

Ben's next step is to sketch the *state transition diagram*, shown in Figure 3.25, to indicate all the possible states of the system and the transitions between these states. When the system is reset, the lights are green on Academic Ave. and red on Bravado Blvd. Every 5 seconds, the controller examines the traffic pattern and decides what to do next. As long as

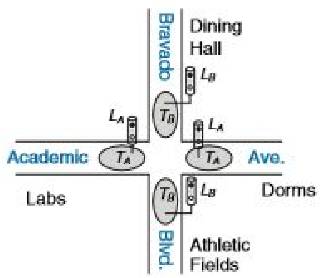

Figure 3.23 Campus map

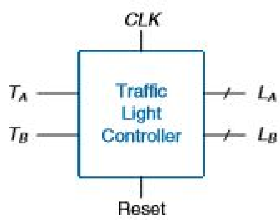

Figure 3.24 Black box view of finite state machine

looking where they are going. Football players are hustling between the athletic fields and the dining hall on Bravado Boulevard. They are tossing the ball back and forth and aren't looking where they are going either. Several serious injuries have already occurred at the intersection of these two roads, and the Dean of Students asks Ben Bitdiddle to install a traffic light before there are fatalities.

Ben decides to solve the problem with an FSM. He installs two traffic sensors, T_A and T_B, on Academic Ave. and Bravado Blvd., respectively. Each sensor indicates TRUE if students are present and FALSE if the street is empty. He also installs two traffic lights, L_A and L_B, to control traffic. Each light receives digital inputs specifying whether it should be green, yellow, or red. Hence, his FSM has two inputs, T_A and T_B, and two outputs, L_A and L_B. The intersection with lights and sensors is shown in Figure 3.23. Ben provides a clock with a 5-second period. On each clock tick (rising edge), the lights may change based on the traffic sensors. He also provides a reset button so that Physical Plant technicians can put the controller in a known initial state when they turn it on. Figure 3.24 shows a black box view of the state machine.

Ben's next step is to sketch the *state transition diagram*, shown in Figure 3.25, to indicate all the possible states of the system and the transitions between these states. When the system is reset, the lights are green on Academic Ave. and red on Bravado Blvd. Every 5 seconds, the controller examines the traffic pattern and decides what to do next. As long as

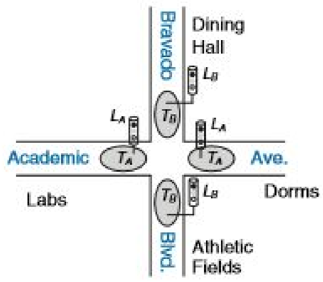

Figure 3.23 Campus map

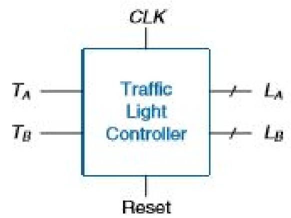

Figure 3.24 Black box view of finite state machine

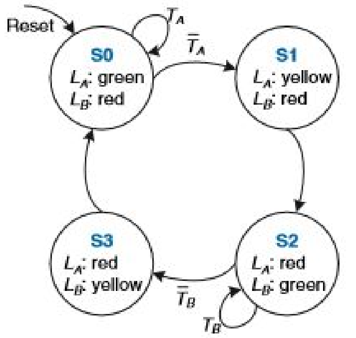

Figure 3.25 State transition diagram

traffic is present on Academic Ave., the lights do not change. When there is no longer traffic on Academic Ave., the light on Academic Ave. becomes yellow for 5 seconds before it turns red and Bravado Blvd.'s light turns green. Similarly, the Bravado Blvd. light remains green as long as traffic is present on the boulevard, then turns yellow and eventually red.

In a state transition diagram, circles represent states and arcs represent transitions between states. The transitions take place on the rising edge of the clock; we do not bother to show the clock on the diagram, because it is always present in a synchronous sequential circuit. Moreover, the clock simply controls when the transitions should occur, whereas the diagram indicates which transitions occur. The arc labeled Reset pointing from outer space into state S0 indicates that the system should enter that state upon reset, regardless of what previous state it was in. If a state has multiple arcs leaving it, the arcs are labeled to show what input triggers each transition. For example, when in state S0, the system will remain in that state if T_A is TRUE and move to S1 if T_A is FALSE. If a state has a single arc leaving it, that transition always occurs regardless of the inputs. For example, when in state S1, the system will always move to S2. The value that the outputs have while in a particular state are indicated in the state. For example, while in state S2, L_A is red and L_B is green.

Ben rewrites the state transition diagram as a *state transition table* (Table 3.1), which indicates, for each state and input, what the next state, S', should be. Note that the table uses don't care symbols (X) whenever the next state does not depend on a particular input. Also note that Reset is omitted from the table. Instead, we use resettable flip-flops that always go to state S0 on reset, independent of the inputs.

The state transition diagram is abstract in that it uses states labeled {S0, S1, S2, S3} and outputs labeled {red, yellow, green}. To build a real circuit, the states and outputs must be assigned *binary encodings*. Ben chooses the simple encodings given in Tables 3.2 and 3.3. Each state and each output is encoded with two bits: $S_{1:0}$, $L_{A1:0}$, and $L_{B1:0}$.

Notice that states are designated as S0, S1, etc. The subscripted versions, S_0, S_1, etc., refer to the state bits.

Table 3.4 State transition table with binary encodings

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0