

- Courseware (/courses/UTAustinX/UT.6.01x/1T2014/courseware)
- Course Info (/courses/UTAustinX/UT.6.01x/1T2014/info)
- Discussion (/courses/UTAustinX/UT.6.01x/1T2014/discussion/forum)
- Progress (/courses/UTAustinX/UT.6.01x/1T2014/progress)
- Questions (/courses/UTAustinX/UT.6.01x/1T2014/a3da417940af4ec49a9c02b3eae3460b/)
- Syllabus (/courses/UTAustinX/UT.6.01x/1T2014/a827a8b3cc204927b6efaa49580170d1/)

VIDEO 10.5. TRAFFIC LIGHT DEMO

Help

	0:29 / 0:29	1.0x			
--	-------------	------	--	--	--

Example 10.1. Design a traffic light controller for the intersection of two equally busy one-way streets. The goal is to maximize traffic flow, minimize waiting time at a red light, and avoid accidents.

Solution: The intersection has two one-ways roads with the same amount of traffic: North and East, as shown in Figure 10.6. Controlling traffic is a good example because we all know what is supposed to happen at the intersection of two busy one-way streets. We begin the design defining what constitutes a state. In this system, a state describes which road has authority to cross the intersection. The basic idea, of course, is to prevent southbound cars to enter the intersection at the same time as westbound cars. In this system, the light pattern defines which road has right of way over the other. Since an output pattern to the lights is necessary to remain in a state, we will solve this system with a Moore FSM. It will have two inputs (car sensors on North and East roads) and six outputs (one for each light in the traffic signal.) The six

PE1=0, PE0=0 means no cars exist on either road

PE1=0, PE0=1 means there are cars on the East road

PE1=1, PE0=0 means there are cars on the North road

PE1=1, PE0=1 means there are cars on both roads

The next step in designing the FSM is to create some states. Again, the Moore implementation was chosen because the output pattern (which lights are on) defines which state we are in. Each state is given a symbolic name:

goN, PB5-0 = 100001 makes it green on North and red on East

waitN, PB5-0 = 100010 makes it yellow on North and red on East

goE, PB5-0 = 001100 makes it red on North and green on East

waitE, PB5-0 = 010100 makes it red on North and yellow on East

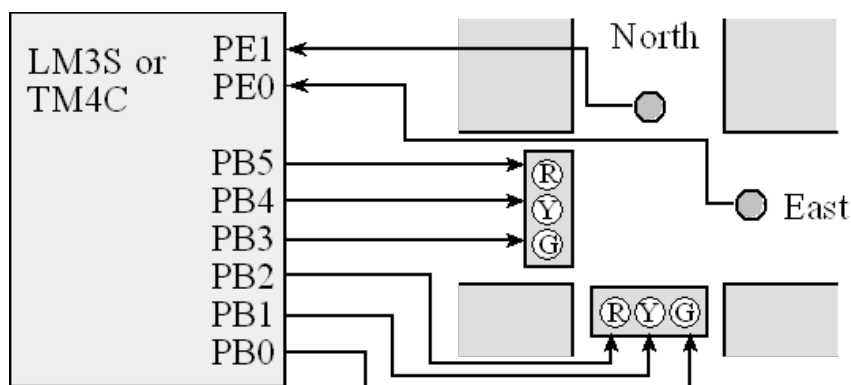


Figure 10.6. Traffic light interface with two sensors and 6 lights.

The output pattern for each state is drawn inside the state circle. The time to wait for each state is also included. How the machine operates will be dictated by the input-dependent state transitions. We create decision rules defining what to do for each possible input and for each state. For this design we can list heuristics describing how the traffic light is to operate:

If no cars are coming, stay in a green state, but which one doesn't matter.

To change from green to red, implement a yellow light of exactly 5 seconds.

Green lights will last at least 30 seconds.

If cars are only coming in one direction, move to and stay green in that direction.

If cars are coming in both directions, cycle through all four states.

Before we draw the state graph, we need to decide on the sequence of operations.

1. Initialize timer and direction registers
2. Specify initial state
3. Perform FSM controller
 - a) Output to traffic lights, which depends on the state
 - b) Delay, which depends on the state

c) Input from sensors

d) Change states, which depends on the state and the input

We implement the heuristics by defining the state transitions, as illustrated in Figure 10.7. Instead of using a graph to define the finite state machine, we could have used a table, as shown in Table 10.2.

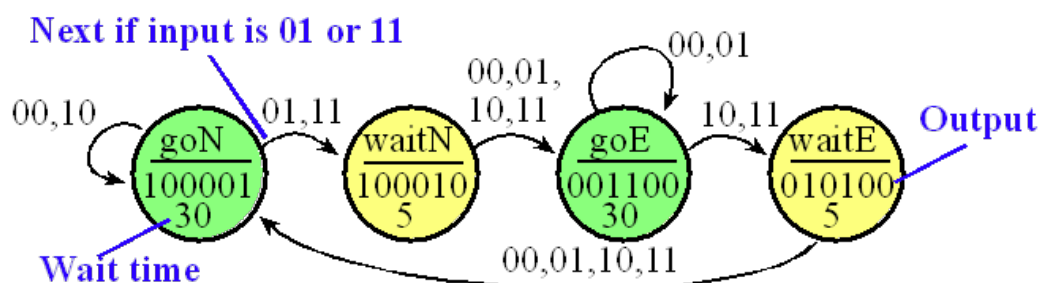


Figure 10.7. Graphical form of a Moore FSM that implements a traffic light.

A **state transition table** has exactly the same information as the state transition graph, but in tabular form. The first column specifies the state number, which we will number sequentially from 0. Each state has a descriptive name. The "Lights" column defines the output patterns for six traffic lights. The "Time" column is the time to wait with this output. The last four columns will be the next states for each possible input pattern.

Num	Name	Lights	Time	In=0	In=1	In=2	In=3
0	goN	100001	30	goN	waitN	goN	waitN
1	waitN	100010	5	goE	goE	goE	goE
2	goE	001100	30	goE	goE	waitE	waitE
3	waitE	010100	5	goN	goN	goN	goN


Table 10.2. Tabular form of a Moore FSM that implements a traffic light.



and artificial intelligence.

Video 10.5. Traffic Light Demo | 10.4 Finite S...

 <https://courses.edx.org/courses/UTAustinX/UT...>
(<https://twitter.com/edXOnline>)

 (<https://plus.google.com/108235383044095082735/posts>)

 (<http://youtube.com/user/edxonline>)
© 2014 edX, some rights reserved.

Terms of Service and Honor Code -
Privacy Policy (<https://www.edx.org/edx-privacy-policy>)

Help