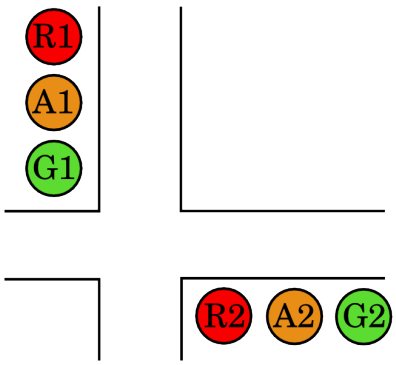
Lecture 10: A Design Example - Traffic Lights

In this lecture we will work through a design example from problem statement to digital circuits.

The Problem

The traffic department is trying out a new system of traffic lights based on the usual European model. We have to design a synchronous digital circuit, a Moore machine, which operates this new type of traffic light at two types of road crossing.

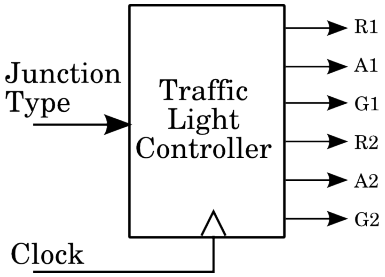


Quiet Junction

Red	Green
Red	Amber
Green	Red
Amber	Red

Busy Junction

Red	Green
Red	Amber
Red	Red
Green	Red
Amber	Red
Red	Red

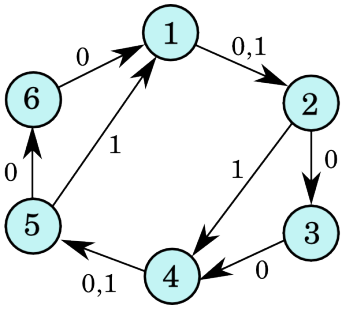


There are six lights to operate. The Red, Amber, and Green lights in the North-South direction will be designated as R1, A1, G1. Similarly, the lights in the East-West direction will be called R2, A2, and G2. When the digital signals are in the Logic-1 state they turn their respective lights on, otherwise the lights are off. A digital clock signal will be supplied and at each clock pulse the lights should change according the schedule given above. The design of the circuit that produces the clock pulses at appropriate times will not be considered here. There are two types of road crossing: quiet crossings that use a simple sequence, and busy crossings require a longer (delayed green) sequence. Some junctions may use the busy sequence during the day and the quiet sequence at night. One digital input signal called J (for junction type) will indicate whether the road crossing is considered quiet. J=0 denotes a busy junction and J=1 a quiet one. Thus, we have a one-input, six-output synchronous system to design.

Step 1: Formalise the problem and decide how many states you need.

Most problems are first specified in a loose verbal form which must be made more rigorous. A good first step in this direction is to determine the number of states required. Sometime the determination of the minimum number of states may be very difficult. However, our problem is simple enough to determine the states easily. Looking at the original specification, we see that there are six states (light patterns) for the busy intersection, and four states in the quiet junction. However we do not need ten states because all four states required for the quiet junction are also used in the busy junction. We need only six states. Let us number them 1 to 6 as shown in the table. Two states (3 and 6) have exactly the same traffic light outputs. Could they be merged as one state? The answer is no, unfortunately, because the state after 3 is 4 while the state after 6 is 1.

State	R1	A1	G1	R2	A2	G2
1	1	0	0	0	0	1
2	1	0	0	0	1	0
3	1	0	0	1	0	0
4	0	0	1	1	0	0
5	0	1	0	1	0	0
6	1	0	0	1	0	0



Step 2. Construct the state transition diagram (ignore outputs)

The state transition diagram is very simple. For the busy junction the states just cycle through in order. For the quiet junction states 3 and 6 are skipped. The state transition diagram is a finite state machine model and now represents a formal specification of what the circuit is required to do.

Step 3: Select the type and number of flip-flops for the circuit.

Since the number of states is equal to six, the minimum number of flip-flops, which can support six states, is three. The maximum number of flip-flops one may use is six (one flip-flop per state), though this implementation would clearly be wasteful and so we will use three D-type flip-flops. There will be two unused states.

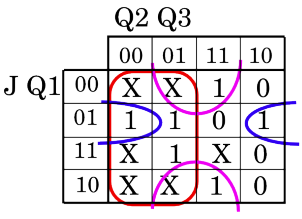
Step 4: Assign state numbers to flip-flop outputs and construct the transition table.

There are some heuristic rules for assigning states to flip-flop outputs, but they are difficult to apply and do not guarantee a minimum circuit. One rule is to maximise the number of 1s. The idea is that a large number of 1s may provide easier minimisation in the Karnaugh maps for the state sequencing logic. On this basis we will not use the states 000 and 001. The rest of the flip-flop outputs are assigned in order while constructing the transition table.

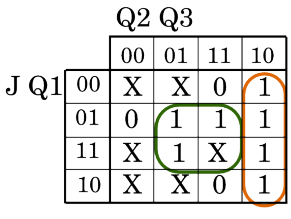
J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	X	X	X	X
0	8	0	0	1	X	X	X	X
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	X	X	X	X
1	8	0	0	1	X	X	X	X
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	X	X	X	X
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	X	X	X	X

Step 5: Fill in K-Maps and determine the minimised expressions

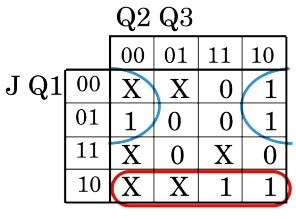
The next step is to determine the required logic expressions for the three flip-flop inputs D1, D2, and D3. Any minimising algorithm can be used, and we will use the normal graphical method (Karnaugh-maps). The Karnaugh maps are constructed for three separate circuits each determining a D value.



D1 = Q2'+ Q3Q1'
+ J'Q1Q3'



D2 = Q1Q3 + Q2Q3'

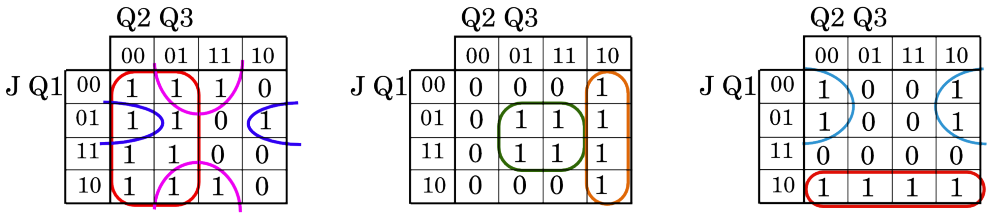


D3 = J'Q3' + J Q1'

Step 6: Construct the Diagram for all States (including don't cares).

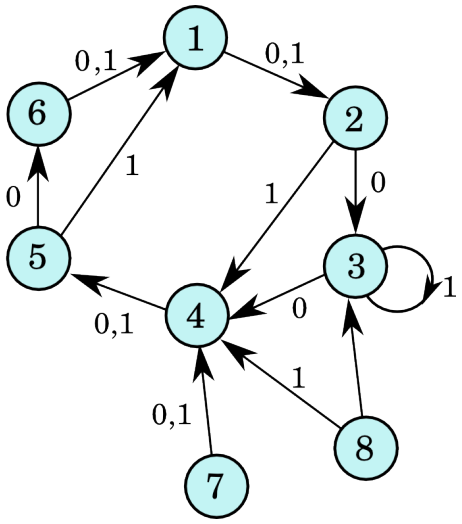
Once the minimisation has been done, we can replace the "don't care" outputs in the Karnaugh maps with the actual values we will get out of the circuit. Any don't care inside a circle is replaced with 1, and any outside all

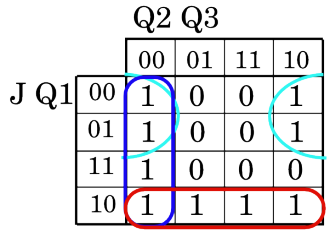
circles is replaced with 0. We have now a completely defined a sequential circuit and we should check whether the system behaves correctly even if it starts from one of the unused states. A convenient way of checking this is by constructing the complete transition diagram in which the unused states 7 and 8 are also included.



The Karnaugh maps now have a value for each of the don't care states we used in the design, so we can re-build the state transition table from them and draw the finite state machine that includes the unused states.

J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	4	1	0	1
0	8	0	0	1	3	1	0	0
0	1	0	1	0	2	0	1	1
0	2	0	1	1	3	1	0	0
0	3	1	0	0	4	1	0	1
0	4	1	0	1	5	1	1	0
0	5	1	1	0	6	1	1	1
0	6	1	1	1	1	0	1	0
1	7	0	0	0	4	1	0	1
1	8	0	0	1	4	1	0	1
1	1	0	1	0	2	0	1	1
1	2	0	1	1	4	1	0	1
1	3	1	0	0	3	1	0	0
1	4	1	0	1	5	1	1	0
1	5	1	1	0	1	0	1	0
1	6	1	1	1	1	0	1	0





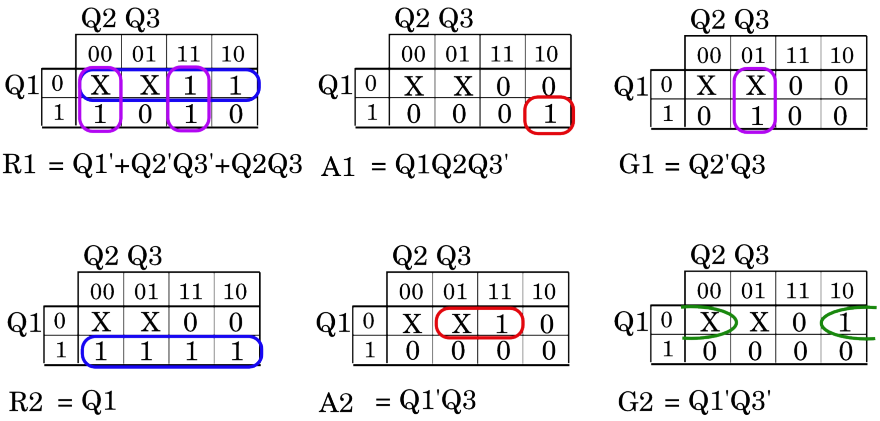
D3 = J'Q3' + JQ1' + Q2'Q3'

Disaster strikes! If the J input is logic 1 (quiet crossing) and the system finds itself in state 3, for example when switched on, or when it changes from busy mode to quiet mode, then it will be stuck in state 3. We have to go back and change some "don't care" bit(s) to fix the problem. This will mean revising our circles and the minimal equations. The problem occurs at Karnaugh map entry 1100. Looking at the three maps, we can see that by changing the 0 indicated for D3 to a 1 will cause the circuit to go to state 4 from state 3 when the safe input is 1. This fixes the problem and causes minimal damage (i.e. will add one extra term to the expression).

Step 7: Construct the Output Circuits in G

For each state we need to generate the signals that light the correct traffic light bulbs. There are six such circuits but fortunately they have three inputs only (Q1,Q2,Q3). Their K-Maps can be filled out by the requirements of lights to be either ON or OFF for each given state. Here again we will start by ignoring the two unused states which will provide "don't care" outputs to find the minimised circuits. Again, filling out the K-maps with the selected 1s and 0s will give us the actual operation of the lights for states 7 and 8. We will have to look at these whether they are safe. From the original specification we can fill in the values for the bulb outputs.

State	Q1	Q2	Q3	R1	A1	G1	R2	A2	G2
1	0	1	0	1	0	0	0	0	1
2	0	1	1	1	0	0	0	1	0
3	1	0	0	1	0	0	1	0	0
4	1	0	1	0	0	1	1	0	0
5	1	1	0	0	1	0	1	0	0
6	1	1	1	1	0	0	1	0	0
7	0	0	0	X	X	X	X	X	X
8	0	0	1	X	X	X	X	X	X



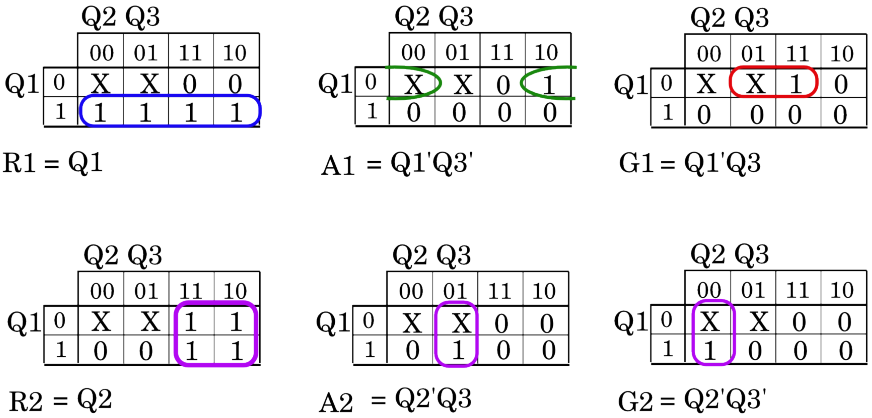
We can check what happens with the don't cares in the above Karnaugh maps by inspection. Consider first state 7 for which the Q values were 000. In the maps for R1 and G2 the don't care will be allocated a 1, and in all other maps it is allocated 0. This means that if by accident we start in state 7 the lights will be red for one direction and green for the other so this is OK. However if we look at state 8, Q values 001, then we will have R1, G1 and A2 set to 1 and the others zero. This means in one direction both red and green are lit and in the other direction amber is illuminated. This could be a problem, however it will only occur for one time state on switch on so we may be able to live with it. If the department of transport is unhappy we will have to fix up the problem in the same way that we did above. In summary we have the following circuits to build. The common terms are underlined. They need only be implemented once in hardware.

D1 = Q2' + J' · Q1 · Q3' + Q1' · Q3 D2 = Q1 · Q3 + Q2 · Q3' D3 = Q2' · Q3' + J' · Q3' + J · Q1'
R1 = Q1' + Q2' · Q3' + Q2 · Q3 A1 = Q2 · Q1 · Q3' G1 = Q2' · Q3
R2 = Q1 A2 = Q1' · Q3 G2 = Q1' · Q3'

A Different State Assignment

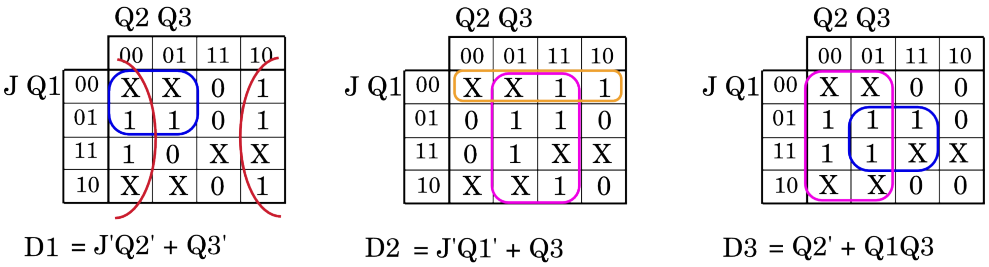
If we want to try to find a simpler overall circuit, we may try different flip-flop assignments for the states. One idea is to minimise the output circuitry. We could, for example, make R1=Q1 and R2=Q2, if these simple assignments will give us a correct complete state assignment. The third output, Q3 has to be assigned such that all used states are distinct. One possible set of assignments are shown below:

State	Q1	Q2	Q3	R1	A1	G1	R2	A2	G2
1	1	0	0	1	0	0	0	0	1
2	1	0	1	1	0	0	0	1	0
3	1	1	1	1	0	0	1	0	0
4	0	1	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1	0	0
6	1	1	0	1	0	0	1	0	0
7	0	0	0	X	X	X	X	X	X
8	0	0	1	X	X	X	X	X	X

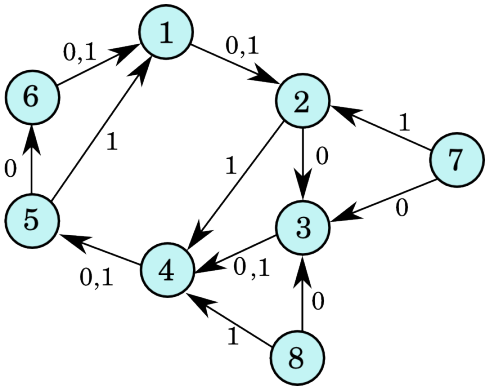


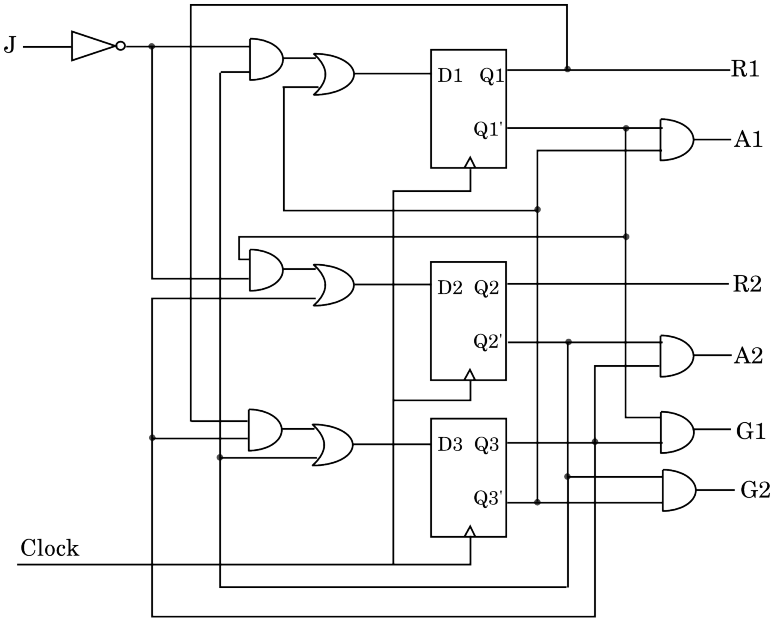
The output circuits are quite a lot simpler and smaller, but of course, we have to redesign the state sequencing logic circuitry with the new flip-flop state assignments.

J	State(t)	Q1	Q2	Q3	State(t+1)	D1	D2	D3
0	7	0	0	0	X	X	X	X
0	8	0	0	1	X	X	X	X
0	1	1	0	0	2	1	0	1
0	2	1	0	1	3	1	1	1
0	3	1	1	1	4	0	1	1
0	4	0	1	1	5	0	1	0
0	5	0	1	0	6	1	1	0
0	6	1	1	0	0	1	0	0
1	7	0	0	0	X	X	X	X
1	8	0	0	1	X	X	X	X
1	1	1	0	0	2	1	0	1
1	2	1	0	1	4	0	1	1
1	3	1	1	1	X	X	X	X
1	4	0	1	1	5	0	1	0
1	5	0	1	0	1	1	0	0
1	6	1	1	0	X	X	X	X



This circuit is simpler than the first one, and if we check the don't care states - which you can do as an exercise - it turns out to be safe. The final state diagram and circuit looks like this:


1
2
3
4
5
6
7
8
0,1
0,1
1
1
0
1
0
0,1
0,1
0
1


J
D1 Q1
Q1'
D2 Q2
Q2'
D3 Q3
Q3'
R1
A1
R2
A2
G1
G2
Clock