Prep Question 1: For your lab you are going to build a slightly different traffic light controller that is designed for the case where A Street is a busy street and B Street is a quiet street. Your controller will obey the following rules:

1. When the light is green on A Street and a car is waiting on B Street, give A Street a yellow light for one clock cycle and then give A Street a red light and B Street a green light for at least one cycle.
2. When the light is green on A Street and there is no car on B Street, leave the light green on A Street.
3. When the light is green on B Street (after the one cycle from step 1) and there is either no car on B Street or a car on A Street, give B Street a yellow light for one clock cycle and then give B Street a red light and A Street a green light for at least three cycles.
4. When the light is green on B Street and there is no car on A Street and there is a car on B street, leave the light green on B Street.
5. There is no need for a reset switch. Your FSM will be initialized by holding carA and carB both low for enough cycles for the FSM to ‘home’ to a known state.

Write a state diagram and a state table for your traffic light controller.

Here is the state diagram. We have three AG states to keep the light green for three cycles as specified by (3) above. The exit from AG3 is held until CarB is true, per (1) and (2) above. We have just a single BG state and the exit from this state occurs when either CarB is false or CarA is true. Note that the feedback arrow is the complement of the forward arrow to BY.
Here is the state table. Note that it takes us two lines to describe the OR arrow from BG to BY.

<table>
<thead>
<tr>
<th>State</th>
<th>CarA</th>
<th>CarB</th>
<th>nextState</th>
<th>A lights</th>
<th>B lights</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG1</td>
<td>*</td>
<td>*</td>
<td>AG2</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>AG2</td>
<td>*</td>
<td>*</td>
<td>AG3</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>AG3</td>
<td>*</td>
<td>0</td>
<td>AG3</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>AG3</td>
<td>*</td>
<td>1</td>
<td>AY</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>AY</td>
<td>*</td>
<td>*</td>
<td>BG</td>
<td>Yellow</td>
<td>Red</td>
</tr>
<tr>
<td>BG</td>
<td>0</td>
<td>1</td>
<td>BG</td>
<td>Red</td>
<td>Green</td>
</tr>
<tr>
<td>BG</td>
<td>1</td>
<td>*</td>
<td>BY</td>
<td>Red</td>
<td>Green</td>
</tr>
<tr>
<td>BY</td>
<td>*</td>
<td>*</td>
<td>AG1</td>
<td>Red</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

**Prep Question 2:** Ignoring reset, write the logic equations for the next-state variables of your traffic light controller from Prep Question 1 using a one-hot state assignment.

\[
\begin{align*}
\text{nextAG1} &= \text{BY} \\
\text{nextAG2} &= \text{AG1} \\
\text{nextAG3} &= \text{AG2} \lor (\text{AG3} \land \neg \text{CarB}) \\
\text{nextAY} &= \text{AG3} \land \text{CarB} \\
\text{nextBG} &= \text{AY} \lor (\text{BG} \land \neg \text{CarA} \land \text{CarB}) \\
\text{nextBY} &= \text{BG} \land (\text{CarA} \lor \neg \text{CarB})
\end{align*}
\]

**Prep Question 3:** Ensure that your FSM homes to a known state when carA and carB are both held low. This should already be the case.

*This is the case. Note that when CarB is low, the machine will advance through the states until it gets to AG3 where it will remain, so the machine homes to AG3 when carB is low regardless of the state of CarA.*

**Prep Question 4:** Write the output logic equations for your FSM from Prep Questions 1 through 3. You need not worry about the state of the outputs during the reset sequence.

\[
\begin{align*}
\text{GreenA} &= \text{AG1} \lor \text{AG2} \lor \text{AG2} \\
\text{YellowA} &= \text{AY} \\
\text{RedA} &= \text{BY} \lor \text{BG} \\
\text{GreenB} &= \text{BG} \\
\text{YellowB} &= \text{BY} \\
\text{RedB} &= \text{AY} \lor \text{GreenA}
\end{align*}
\]

Alternatively you can write RedB as RedB = \neg (\text{BG} \lor \text{BY})
**Prep Question 5:** In preparation you should draw a complete schematic of your traffic light controller. Your schematic should take two inputs, a clock input, and generate six outputs. Use the 74AC377 octal D-flip-flop to hold your state variables. You can use any of the gate parts 74AC00, AC02, AC08, AC32, AC10, AC20, etc… to implement your next state and output logic.

*I can’t stress enough the importance of a good schematic to make your wiring and debug go smoothly. Make sure to label all chips and pins and draw the schematic to reflect the function of the circuit – with information flowing from left to right and top to bottom.*
The Lab

Here is a photo of the lab wired up and ready for traffic. The five chips across the bottom are U4, U3, U2, U1, and U5 in that order. DIP switches 1 and 2 are carB and carA respectively. The lights on the top are for A Street and the ones on the bottom are for B street.