**Lab Report: Thunderbird Tail Lights**

Rohan Seam and Ibrahima Diallo

TCES 230: Logic Design

12/2/2018

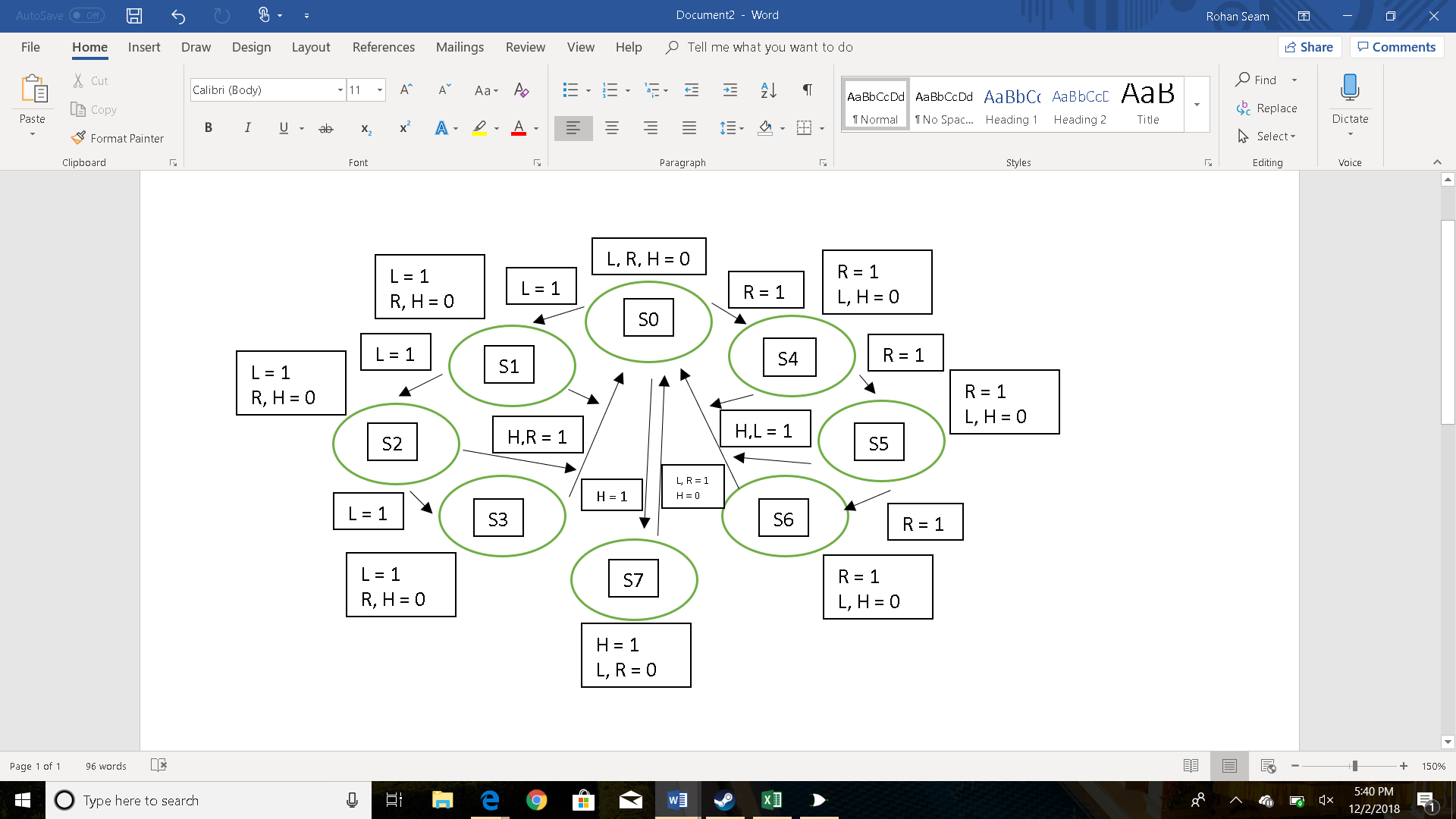
**Introduction**

In this lab, we were given a scenario where we needed to create a circuit that follows the tail light pattern of an older model Ford thunderbird. The pattern that we were required to follow was one that when the left turn signal was activated, the innermost left light would be on, followed by the inner two, and finally all 3. Then, it would return to the idle state where all the lights would be off, and the pattern would reset. We followed the same pattern for the right tail light as well. When hazards are activated, no matter what state the circuit is in, the circuit should reset to the idle state and then go to the hazard state which means all lights are on, then alternate between idle and all lights on. We were introduced to the concept of finite state machines and how they can benefit us with circuit design and implementation of this circuit. We also used online tools to help us simplify our logic to simplify our circuit. Finally, we used ModelSIm and Logisim to simulate our designed circuit.

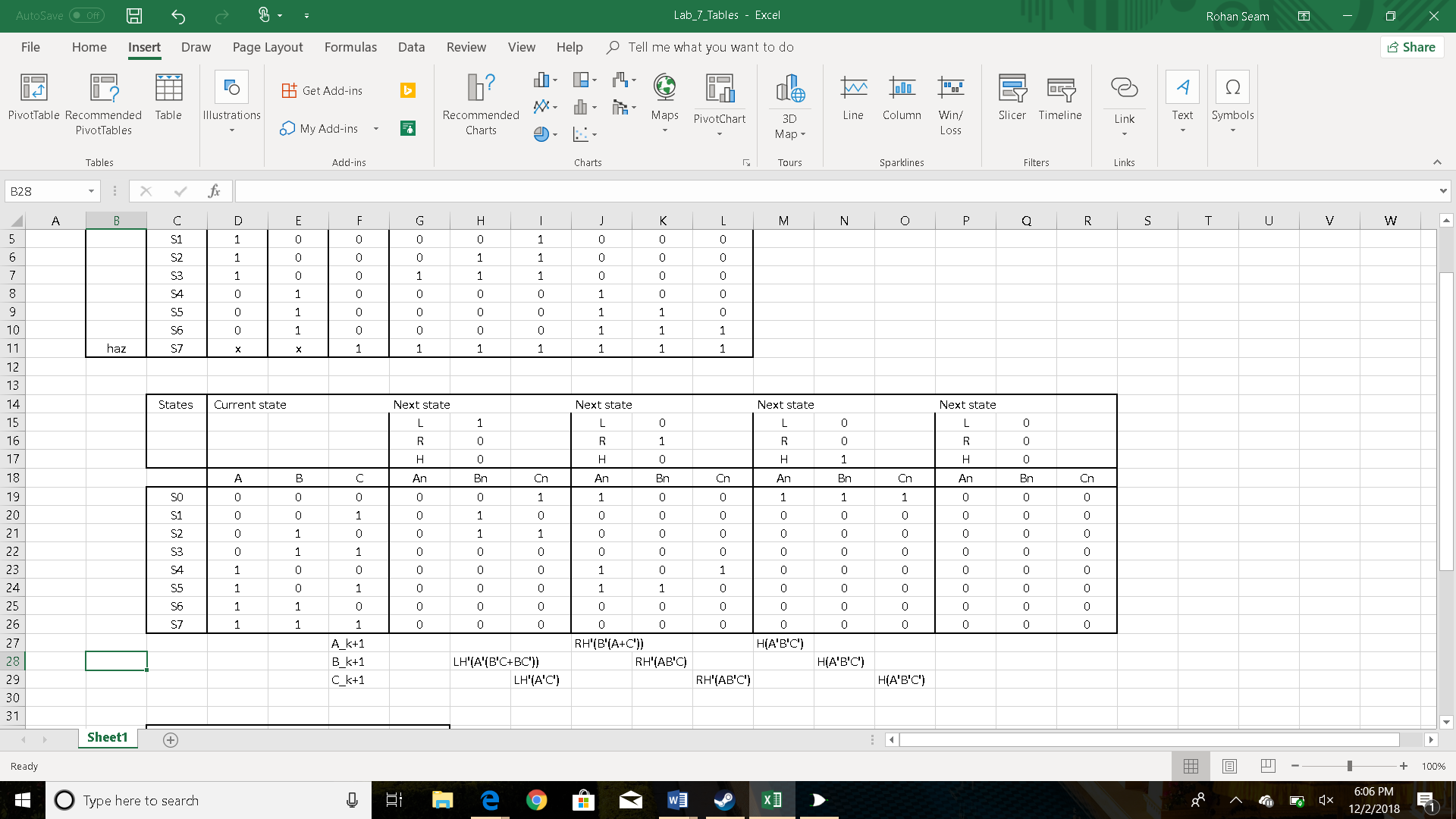
**Methods**

In order to even understand where to start with this circuit, we know that we needed to understand what our states were for the circuit. We must remember that when one of the blinkers is activated, the lights follow the pattern of the innermost on, two innermost ones on, all on, then back to idle state.

**Table 1:** State Diagram of the Circuit to be Designed

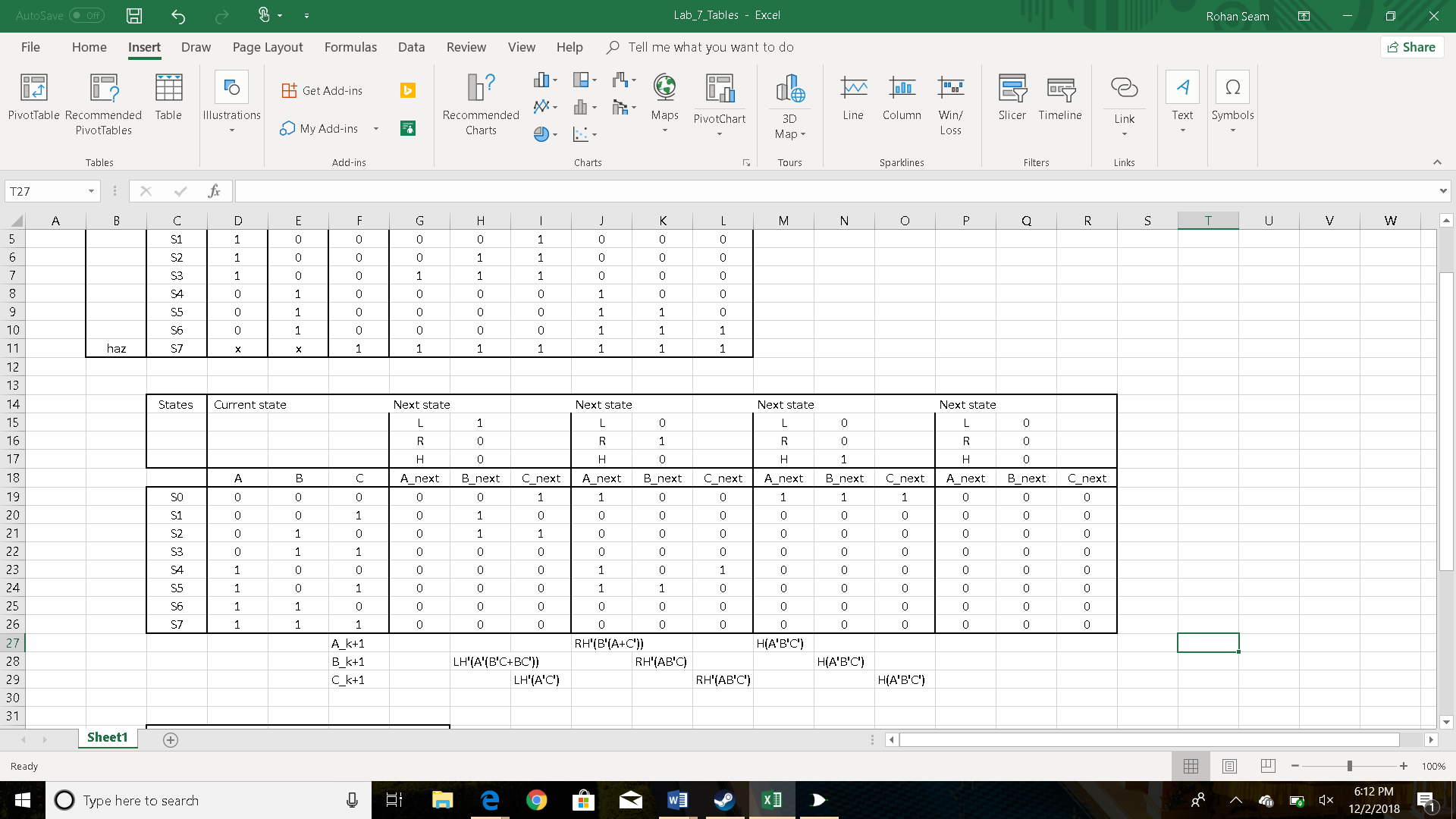
We can see that from the table that there are a total possible of 8 states that the circuit can currently be in. (L represents input from the left blinker, R represents input from the right blinker, and H represents the hazard lights). We know that S0 is the idle state and we have the possibility of heading to S1 when the left blinker is activated, S4 when the right blinker is activated, or S7 when the hazards are activated. If the left blinker is on and the hazards and right blinker is off, the left blinker will flash from innermost on, second and inner on, all on, then back to idle state. We see the same operation with the right blinker as well. One other interesting observation is that when we go from S0 to S7, we know that we must alternate between idle state and hazard state to notice the correct operation of the hazard lights. We then created binary labels for each of the states we have in table 1.

**Table 2:** Binary labels for states in table 1

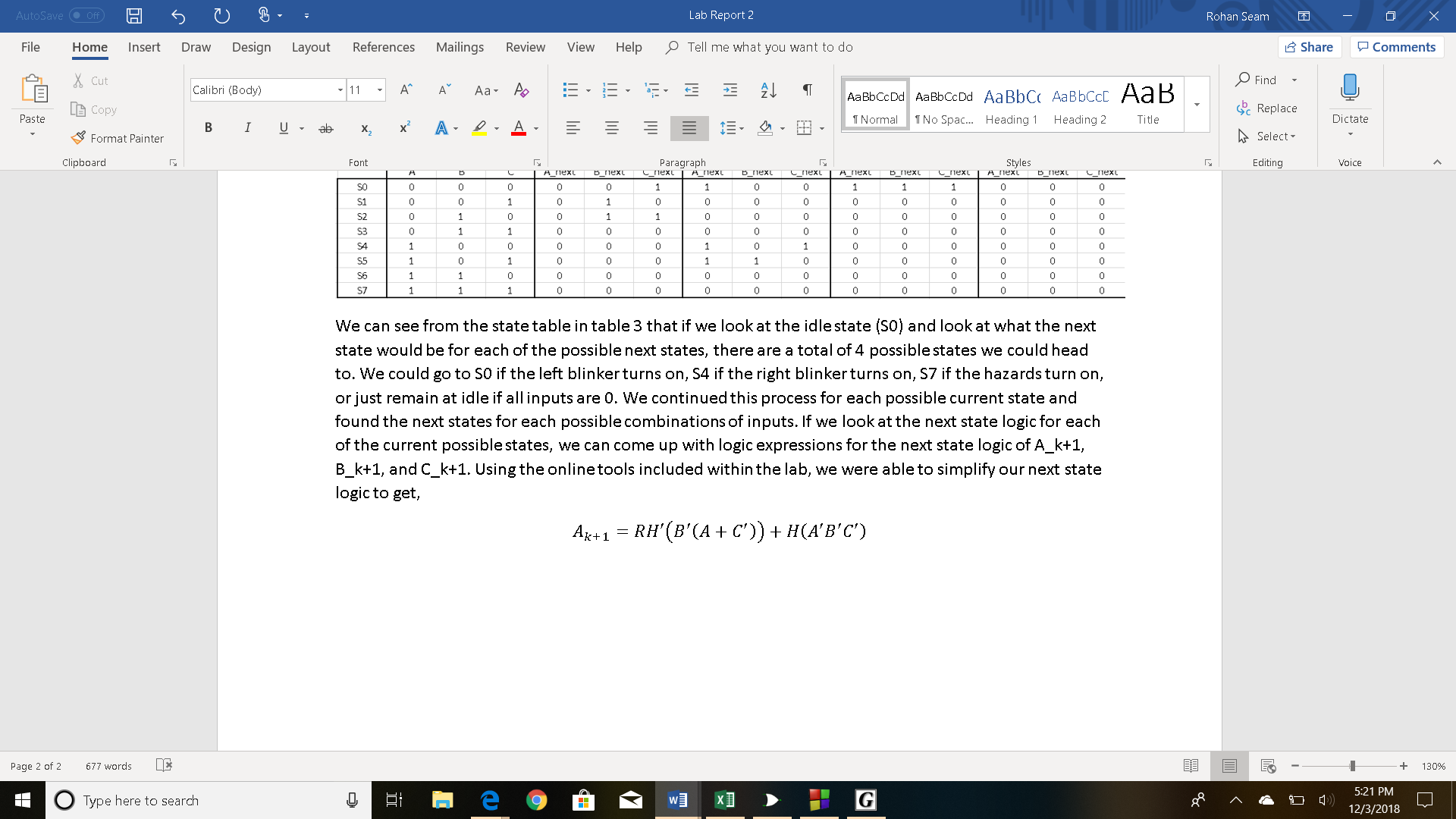
We assigned the idle state to state 000 and incremented up till we get to state 8 (S7), which is the “all lights on” state. We can see that from idle state, if we turn the right blinker on, we go to state (S4) which is 100. On the other hand, if we turn the left blinker on, we go to state (S1), 001.

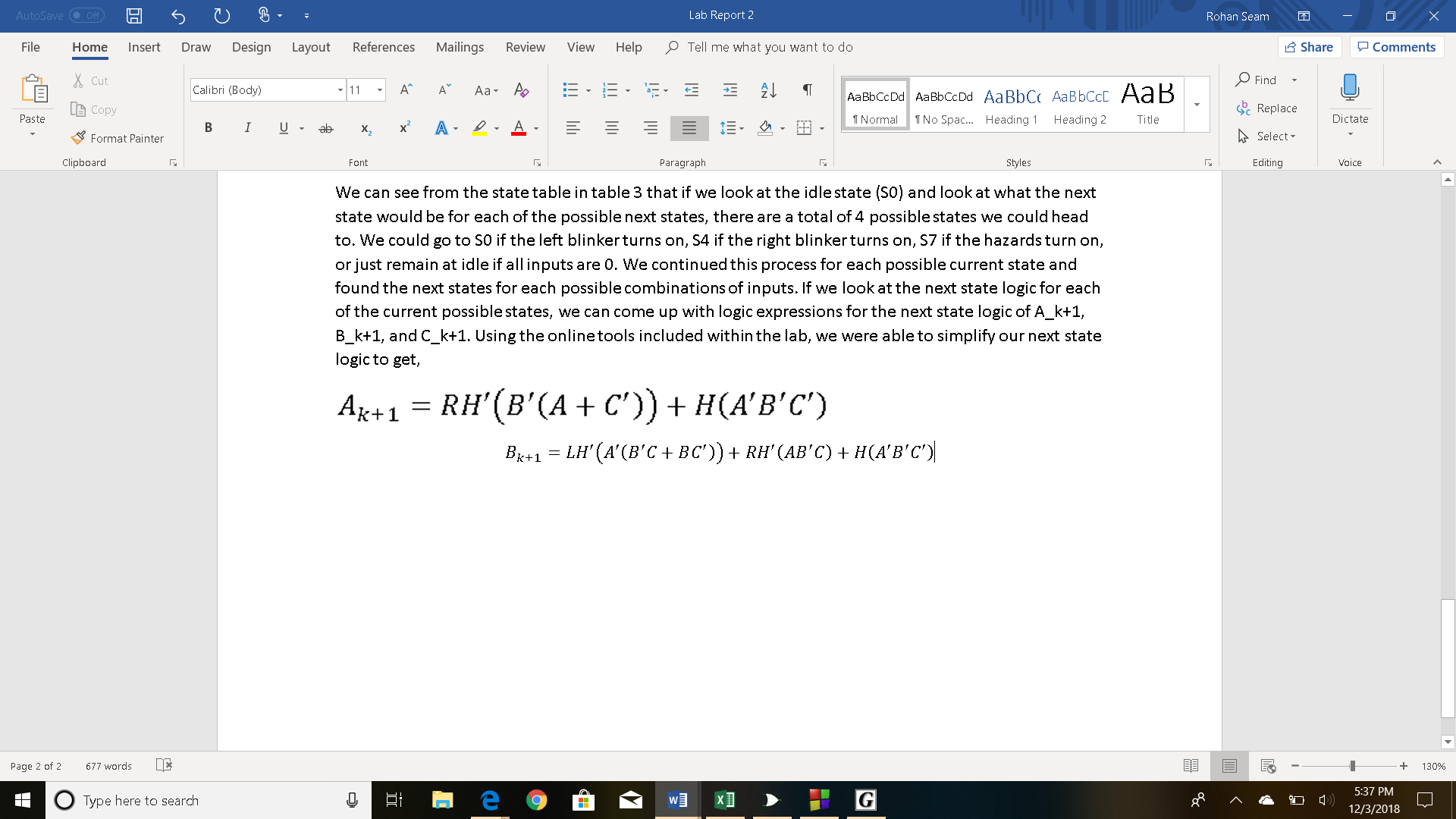
Next, we created a state table to understand what our current and next states would be for each instance of each state.

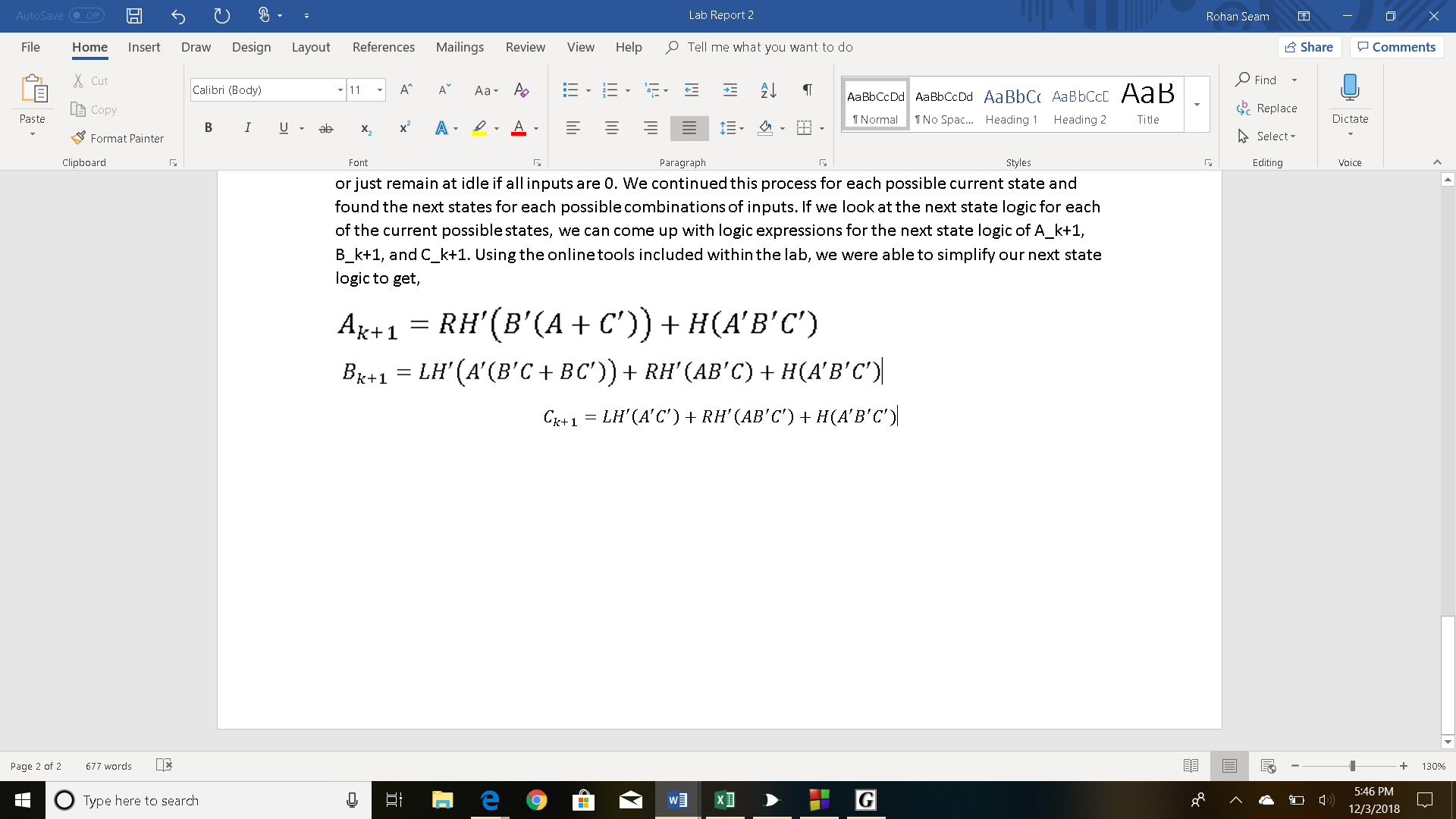
**Table 3:** State table of Circuit to be Designed



We can see from the state table in table 3 that if we look at the idle state (S0) and look at what the next state would be for each of the possible next states, there are a total of 4 possible states we could head to. We could go to S0 if the left blinker turns on, S4 if the right blinker turns on, S7 if the hazards turn on, or just remain at idle if all inputs are 0. We continued this process for each possible current state and found the next states for each possible combinations of inputs. One important thing to remember is that no matter what state we are in the circuit, if the hazards are activated, we return to idle state and then to the all light on state. If we look at the next state logic for each of the current possible states, we can come up with logic expressions for the next state logic of A\_k+1, B\_k+1, and C\_k+1. In order to formulate Kmaps to simplify our logic to find the logic expressions, we first looked at the first row of each of the next state sections and applied Kmap simplification techniques to formulate our logic expressions. Using the online tools included within the lab, we were able to simplify our next state logic to get,

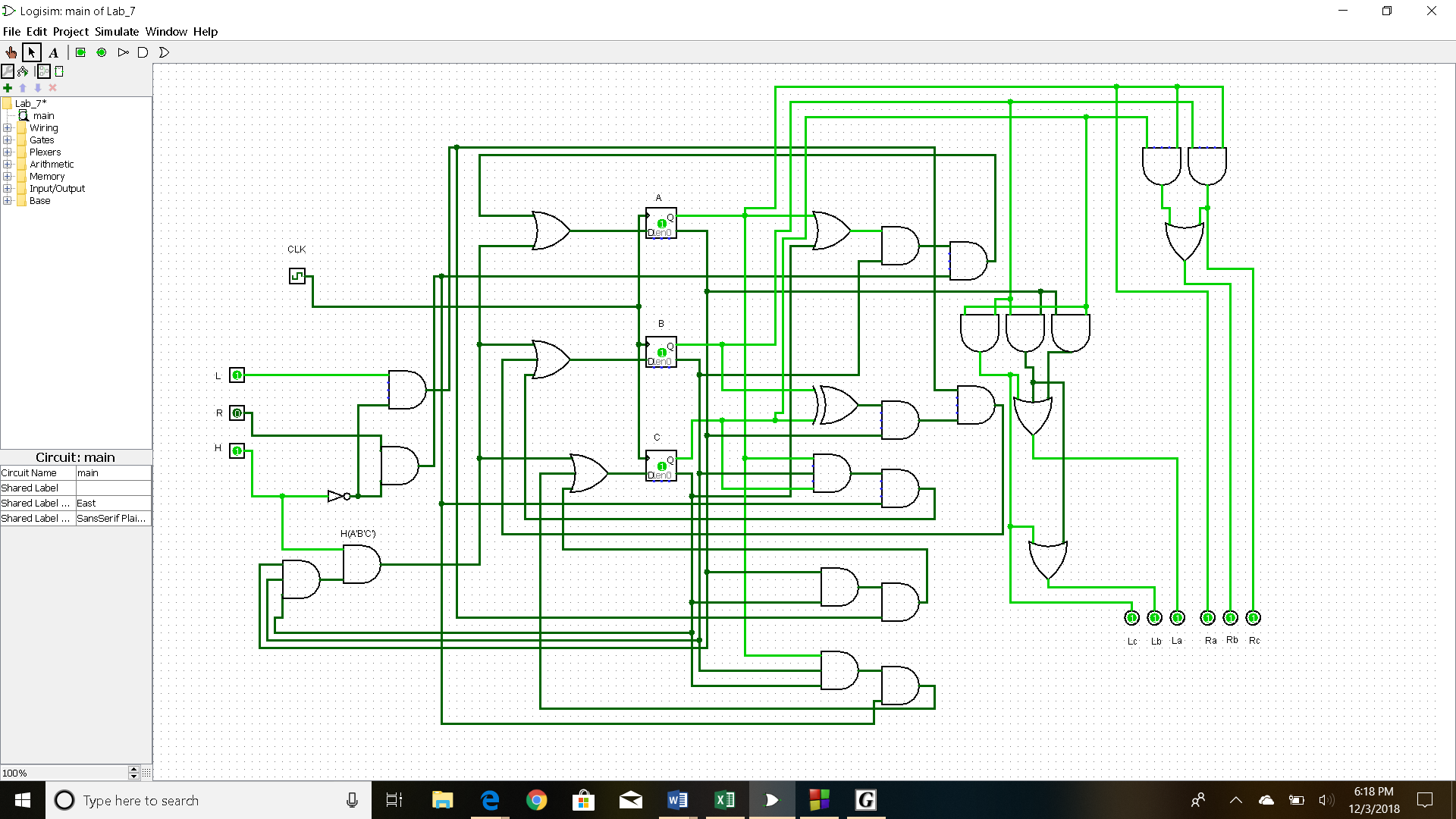




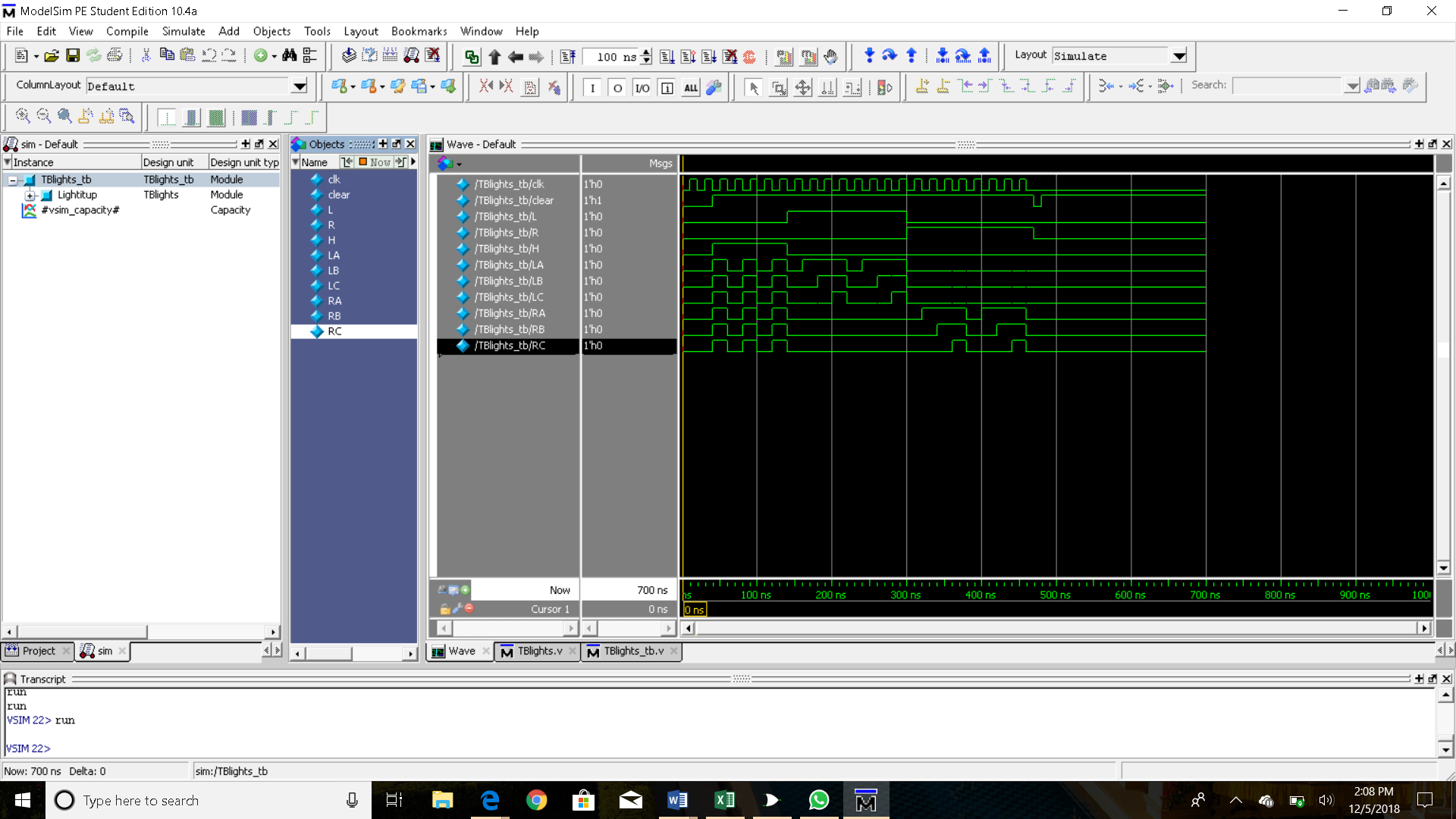


Now that we found the next state expressions, we were finally able to create the circuit and verify that is operates as intended by using the logic in the equations above.

**Figure 1:** Circuit Implementation of Tail Light Pattern

After building the circuit, we were able to simulate and verify the functionality of the circuit. We confirmed that it indeed functioned as per the instructions for the pattern we saw in the pre-lab instructions. On the left side of the circuit are our inputs (L, R, H). L is for the control of the left blinker, R is for the control of the left blinker, and H is for the control of the hazard lights. We tested every combination of inputs and compared to the output that we found in table 3. On the bottom right of the circuit, we see the outputs which represent the tail lights of the older model Ford Thunderbird. La represents the innermost light of the left tail light, Lb represents the middle, and Lc represents the outermost light of the left tail light. We see the same orientation for the right tail light as well. We then moved on to simulate the circuit using ModelSim and Verilog code. We were able to implement this circuit behavior with predefined modules for the D-flip flop (circuit that allows us to store state information).

**Table 4:** Timing diagram of circuit in figure 1

As we can see from the timing diagram generated based off the circuit in figure 1, we initially have all the output lights flashing in unison which means the hazards are active when we have the hazard input on high. Then, we see that the H input goes low and L is now high. This means that the left turn blinker should now be activated, and which is indeed true based on what we see now from the timing diagram. We know that the innermost left turn light should be on first (La), then the innermost two lights (La, Lb), then finally all 3 (La, Lb, Lc). Finally, we see that when we have L and H low, but R high, the right turn signal is activated which follows what we saw for the left turn light. Based on the specifications that we were presented earlier in the lab on the function of the tail lights, this timing diagram confirms the validity of the designed circuit.

**Conclusion**

Overall, the main idea of this lab was to implement new concepts we learned in class to further expand our knowledge of designing and creating digital circuits. We were given a scenario where we had to design and create a circuit that followed the light pattern of an older model Ford Thunderbird. We first determined all the possible states that the circuit could be in and created a state diagram that shows the states and routes that could be taken to reach each state. Then, we created a state table and used the concepts we learned earlier in the quarter to derive equations for the next state logic. We then used these expressions to design and simulate our circuit while also verifying our outputs with the expected ones. This lab taught me why state diagrams can help us with circuit design and how they are useful. Also, I also learned that it is very important to understand what the states of a circuit are and what the outputs of the circuit are. Furthermore, I now understand how to correctly derive next state expressions correctly using the online tools that were provided. If I were to redo this lab, I would make sure that I understand what is being asked of me in the lab. One major issue me and my partner had was mixing up states with output which set us back on design and simulation.