ELEN 21 HW #7

Tamir Enkhjargal
November 2018

Question #1:

Let's look at a simple state machine. Say that the input signal w represents a toggle switch for a flashlight that has two modes: blink once or stay on. The output z represents the light being on. To get the light to blink once (i.e. have z assert for once cycle), w needs to assert for just one cycle. To get the light to stay on (i.e. have z stay asserted), w needs to assert for two consecutive cycles. After that, it will stay on while w is de-asserted. When w asserts again, the light will be turned off.

Part A. Draw a state diagram to represent this behavior.

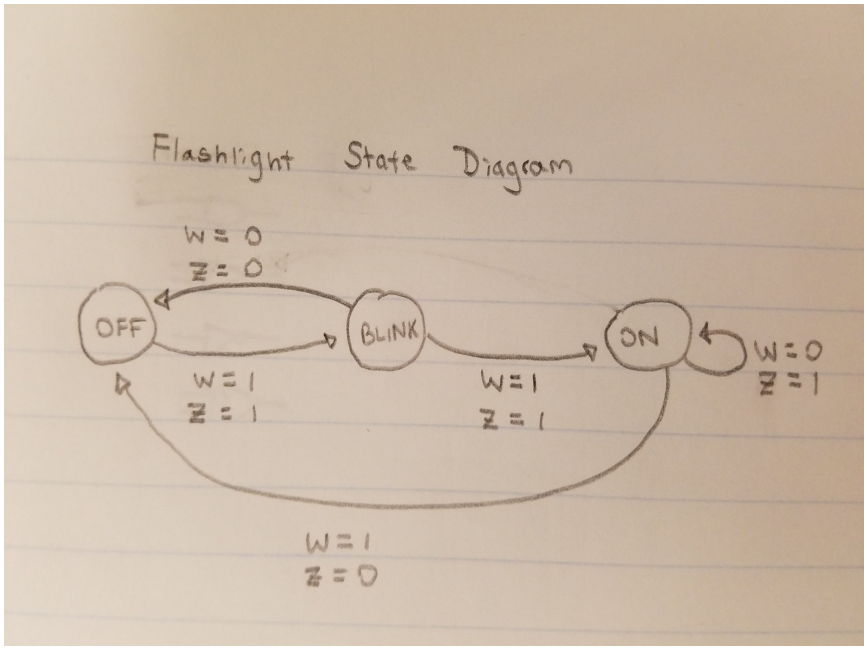


Figure 1: State Diagram for a Flashlight with Two Modes

Part B. Using the minimum amount possible, how many flops do we need to represent these states?

Since there are only 3 states, we would need a minimum of 2 flops to account for at least 4 states ($2^2 = 4$).

Part C. & Part D. Using the number of flops in your answer to **B**, create a state table that shows the next states as a function of the current states and the input signal, and that shows the value of z in each state. Determine logic equations for the next state signals, and the output, based on what you have defined in your state table.

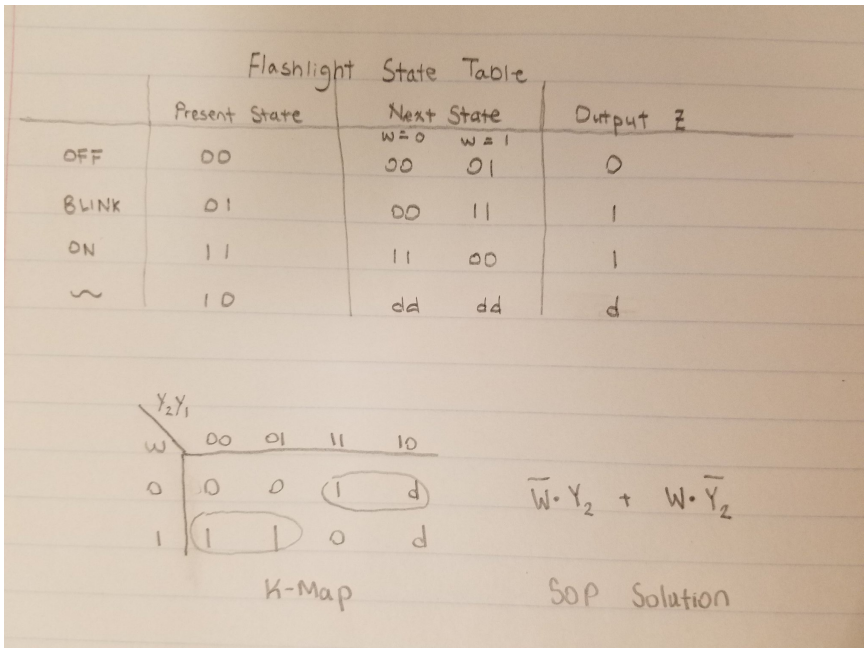


Figure 2: State Table and SOP Solution for a Flashlight with Two Modes

Question #2:

We want to define the state machine for the traffic lights at an intersection. We know that any given traffic light has three outputs, Red, Yellow, and Green. But we have two different directions of streets. We'll call them Main street and Side street. Therefore, there are three outputs (R, Y, G) that will cause the corresponding colored light to illuminate, but we need two versions of each Red on the Main street and Red on the Side street.

The normal (reset) state will be to show Green on the Main street and Red on the Side street. A signal called "car" will tell the machine that a car has arrived on the Side street, which will cause the machine to move from Green to Yellow and then to Red on the Main street, at which point the Side street can go to Green. When "car" is de-asserted, the Side street will cycle through Yellow back to Red and the Main street will go back to green.

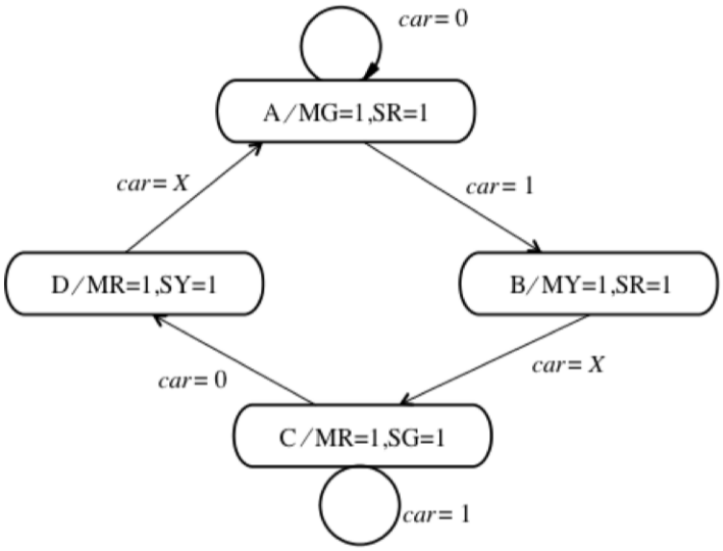
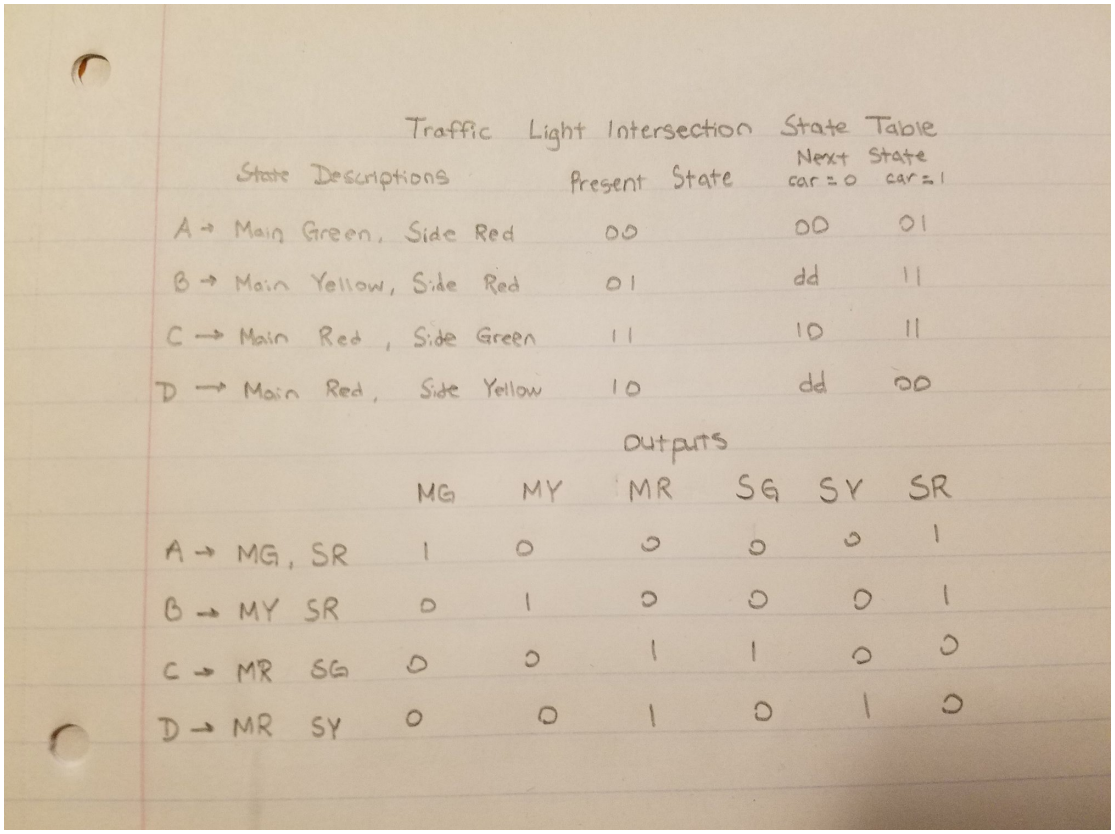


Figure 3: State diagram for traffic lights at an intersection,

Part A. Create a state table that shows the next states as a function of the current states and the input signal, and that shows the value of the six output control signals in each state.



Traffic Light Intersection State Table

State Descriptions	Present State	Next State car = 0	Next State car = 1
A → Main Green, Side Red	00	00	01
B → Main Yellow, Side Red	01	dd	11
C → Main Red, Side Green	11	10	11
D → Main Red, Side Yellow	10	dd	00

Outputs

	MG	MY	MR	SG	SY	SR
A → MG, SR	1	0	0	0	0	1
B → MY SR	0	1	0	0	0	1
C → MR SG	0	0	1	1	0	0
D → MR SY	0	0	1	0	1	0

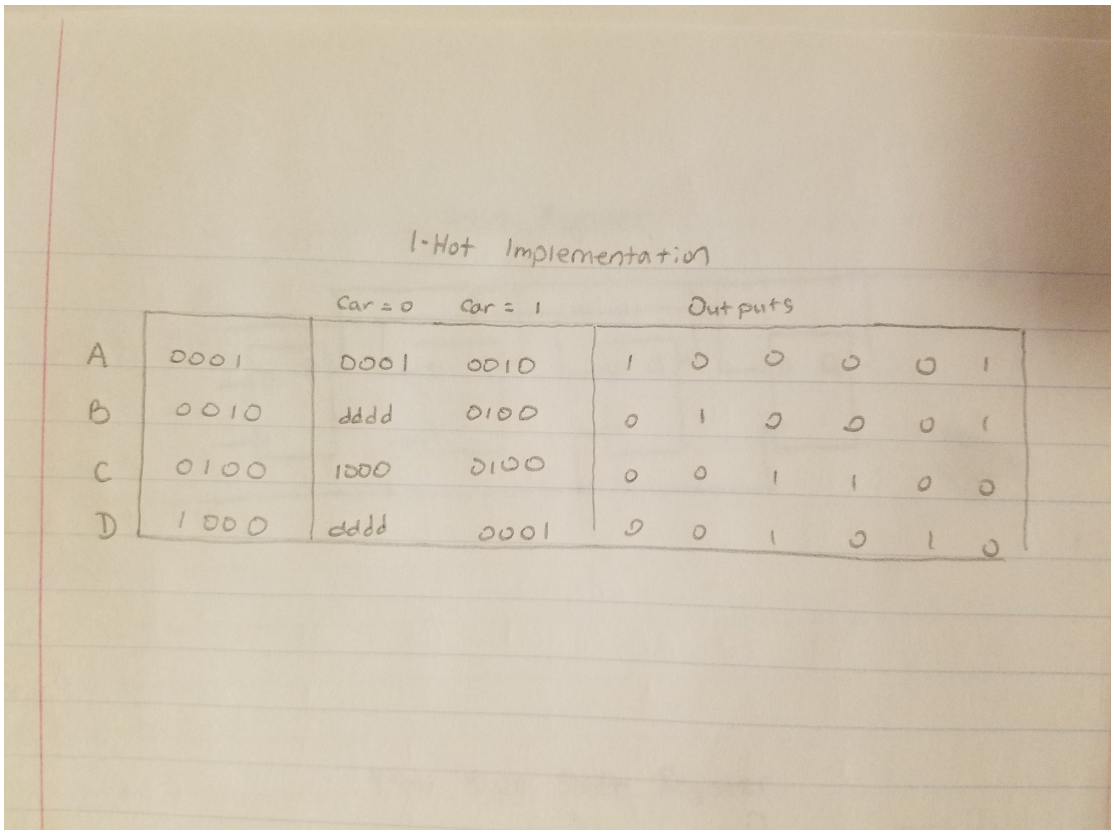
Figure 4: State table for traffic lights with six outputs

Part B. Determine logic equations for the next state signals, and the output control signals, based on what you have defined in your state table.

These values of $Main$ or $Side$ are 1 (green/on) or 0 (yellow/red/off). Car is 1 (Car in side) or 0 (No car in side)

The Main street green light depends on $\overline{Car} \cdot \overline{Side}$.
The Main street yellow light depends on $Car \cdot \overline{Main} \cdot \overline{Side}$.
The Main street red light depends on $Car \cdot Side$.
The Side street green light depends on $\overline{Main} \cdot Side$.
The Side street yellow light depends on $\overline{Car} \cdot Main$.
The side street red light depends on $\overline{Car} \cdot \overline{Side}$.

Part C. Now create the state table with 1-hot encoded state assignments.



1-Hot Implementation

		Car = 0	Car = 1	Outputs					
A	0001	0001	0010	1	0	0	0	0	1
B	0010	dddd	0100	0	1	0	0	0	1
C	0100	1000	0100	0	0	1	1	0	0
D	1000	dddd	0001	0	0	1	0	1	0

Figure 5: State table with one-hot implementation with six outputs

Part D. Determine the logic equations for the next state inputs to the flops, and for the six output control signals. Note that there are now four state variables so you'll need four next state equations.

The six output control signals should remain the same logically. With this one-hot implementation, we can show that the output of Main yellow inputs to Main red, which inputs to Side green. We see the same with the output from Side yellow to Side red to Main green.

$MGout = SR_{in}$,
$MY_{out} = MR_{in} = SGin$,
$SY_{out} = SR_{in} = MGin$,
$SGout = MRin$