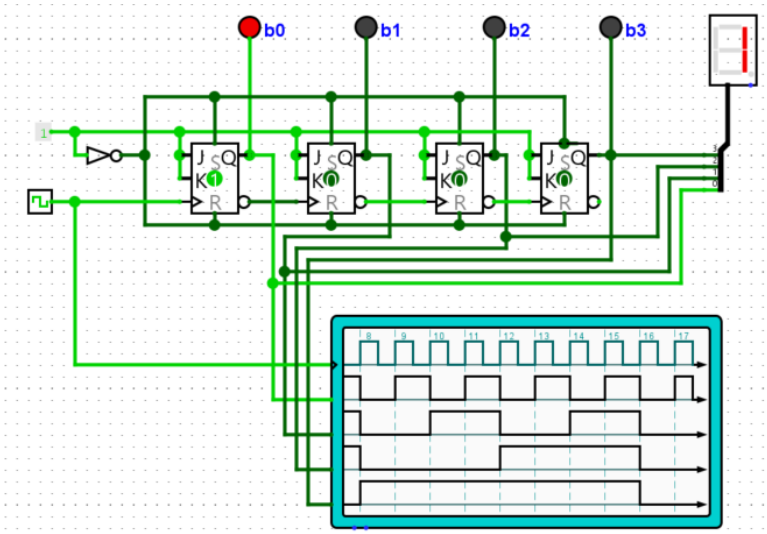
Digital IAssignment II – Model Solutions (*logisim* files included in a separate folder)

You can download *logisim-evolution* here (installation instructions are provided there)

Question 1

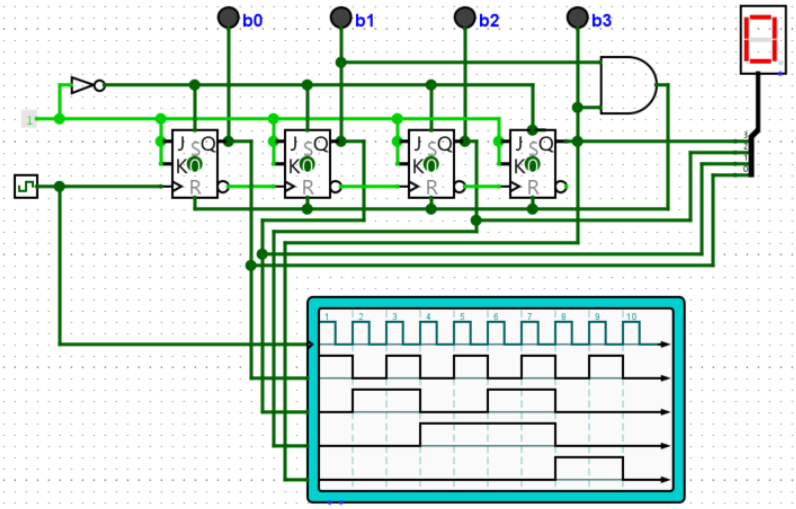
Simulate the circuit

- Counts up from 0000 to 1111 then back to 0000 (from the results that we will see)
- The given circuit uses negative edge triggered flip-flops.
- If you have positive edge triggered flip-flops (just like we do in this model solution), invert the Q output before applying it to the clock of the next flip flop.
- Alternatively, you can use $\bar{Q}$ output.



Simulating the circuit, with the switch replaced with a clock. $\bar{Q}$ is used to trigger succeeding flip-flops because the flip-flops are positive edge triggered. If we use Q, the counter would count downwards from 1111 to 0000 then back to 1111.

Now, suppose you want to modify the circuit to count from 0 to 9 and back to 0, do:

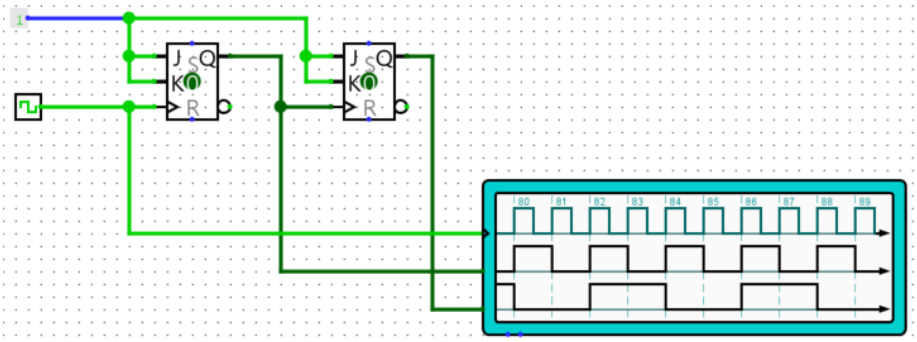


Decade counter using positive edge triggered flip-flops. For negative edge flip-flops, trigger the subsequent flip-flops using Q.

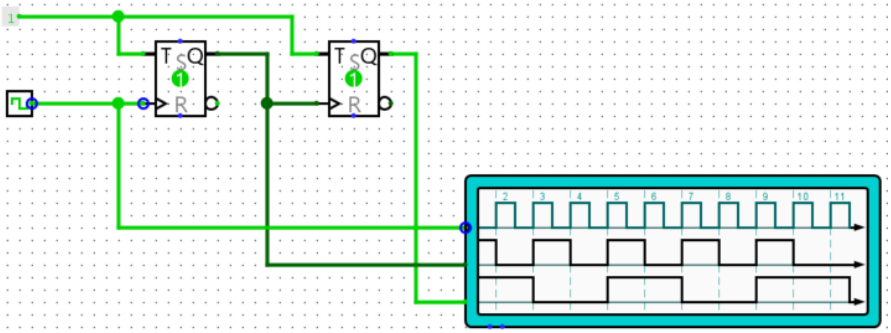
Question 2

Clock division

- 40 MHz to 10 MHz ($T = 100$ ns)
- Requires a ripple counter with $\lceil \log_2 4 \rceil = 2$ flip-flops
- It does not matter whether it is connected to count up or down



Frequency division by 4 (using JK flip-flops)



Frequency division by 4 (using T flip-flops)

Question 3

BCD to binary

Assumption: non-BCD codes (10 to 15) give a result of 0000_2

Inputs (BCD bits): **A, B, C** and **D**

Outputs (minimum of 4 bits): **W, X, Y** and **Z**

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

$W = A\bar{B}\bar{C}$

CD \ AB	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	0	0	0	0
10	0	0	0	0

$X = \bar{A}B$

CD \ AB	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	0	0

$Y = \bar{A}C$

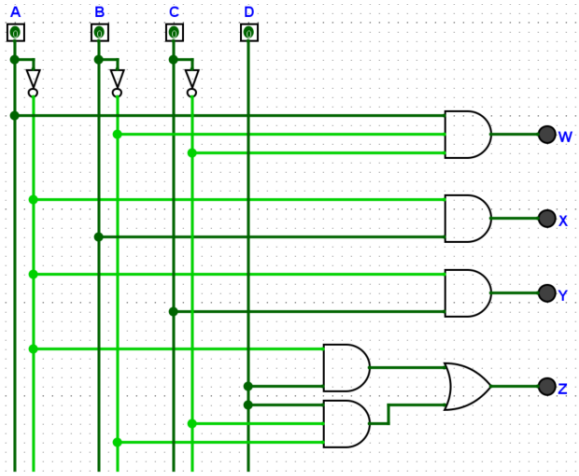
CD \ AB	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	0	0
10	0	1	0	0

$Z = \bar{A}D + \bar{B}\bar{C}D$

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	0	1
11	1	1	0	0
10	0	0	0	0

Truth table and minimized functions

a) AND-OR-NOT implementation
Implemented as is, using AND, OR, and NOT gates only.

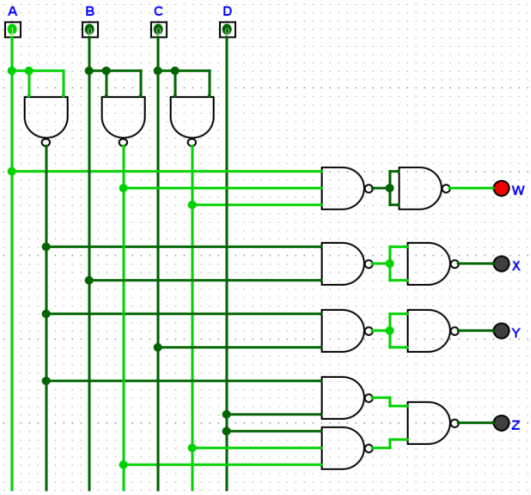


AND-OR-NOT implementation

b) NAND gates only implementation

$$W = \overline{\overline{A}\,\overline{B}\,\overline{C}}$$
$$X = \overline{\overline{\overline{A}}\,B}$$
$$Y = \overline{\overline{\overline{A}}\,C}$$
$$Z = \overline{\overline{\overline{A}}\,D} \cdot \overline{\overline{\overline{B}}\,\overline{C}\,D}$$

- NOT gates also replaced using 2 input NAND gates

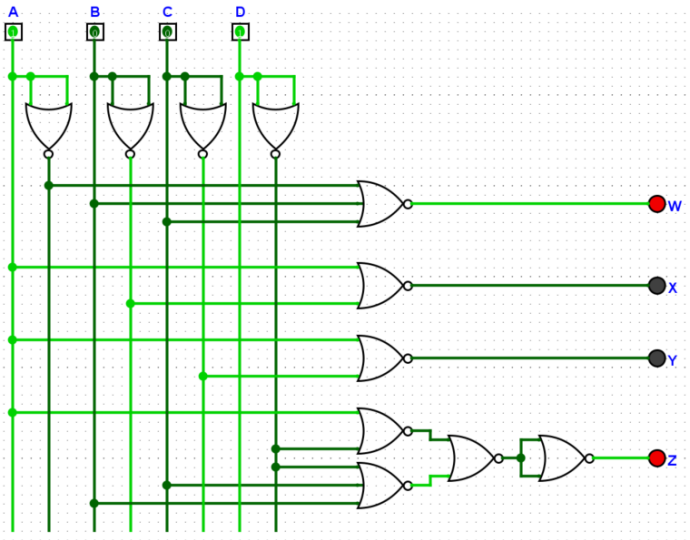


NAND gates only implementation

c) NOR gates only implementation

$$W = \overline{\overline{A}\,\overline{B}\,\overline{C}} = \overline{\overline{A} + B + C}$$
$$X = \overline{\overline{\overline{A}}\,B} = \overline{A + \overline{B}}$$
$$Y = \overline{\overline{\overline{A}}\,C} = \overline{A + \overline{C}}$$
$$Z = \overline{\overline{\overline{A}}\,D} \cdot \overline{\overline{\overline{B}}\,\overline{C}\,D} = \overline{(A + \overline{D})} + \overline{(B + C + \overline{D})}$$

- NOT gates also replaced using 2 input NOR gates



NOR gates-only implementation

d) Type 2 multiplexer implementation.

- **Choosing the multiplexer:**

$\#\ of\ select\ lines\ =\ Total\ number\ of\ input\ variables\ -Type\ \# = 4 - 2 = 2$

There are 4 input variables (4 BCD bits), and the type # is 2

Multiplexer to use: 4-to-1 (because of 2 select lines)

- We need to partition the truth table to remain with two variables (we will partition using D, then using C, for each output. For illustration, here we partition up to B.

i. Partitioning for $\boldsymbol{W}$

A	B	C	D		W	W(D)	W(C,D)	W(B,C,D)
0	0	0	0		**0**	0	0	0
0	0	0	1		**0**			
0	0	1	0		**0**	0		
0	0	1	1		**0**			
0	1	0	0		**0**	0	0	
0	1	0	1		**0**			
0	1	1	0		**0**	0		
0	1	1	1		**0**			
1	0	0	0		**1**	1	$\bar{C}$	$\bar{B}\bar{C}$
1	0	0	1		**1**			
1	0	1	0		**0**	0		
1	0	1	1		**0**			
1	1	0	0		**0**	0	0	
1	1	0	1		**0**			
1	1	1	0		**0**	0		
1	1	1	1		**0**			

ii. Partitioning for $\boldsymbol{X}$

A	B	C	D		X	X(D)	X(C,D)	X(B,C,D)
0	0	0	0		**0**	0	0	B
0	0	0	1		**0**			
0	0	1	0		**0**	0		
0	0	1	1		**0**			
0	1	0	0		**1**	1	1	
0	1	0	1		**1**			
0	1	1	0		**1**	1		
0	1	1	1		**1**			
1	0	0	0		**0**	0	0	0
1	0	0	1		**0**			
1	0	1	0		**0**	0		
1	0	1	1		**0**			
1	1	0	0		**0**	0	0	
1	1	0	1		**0**			
1	1	1	0		**0**	0		
1	1	1	1		**0**			

iii. Partitioning for $\boldsymbol{Y}$

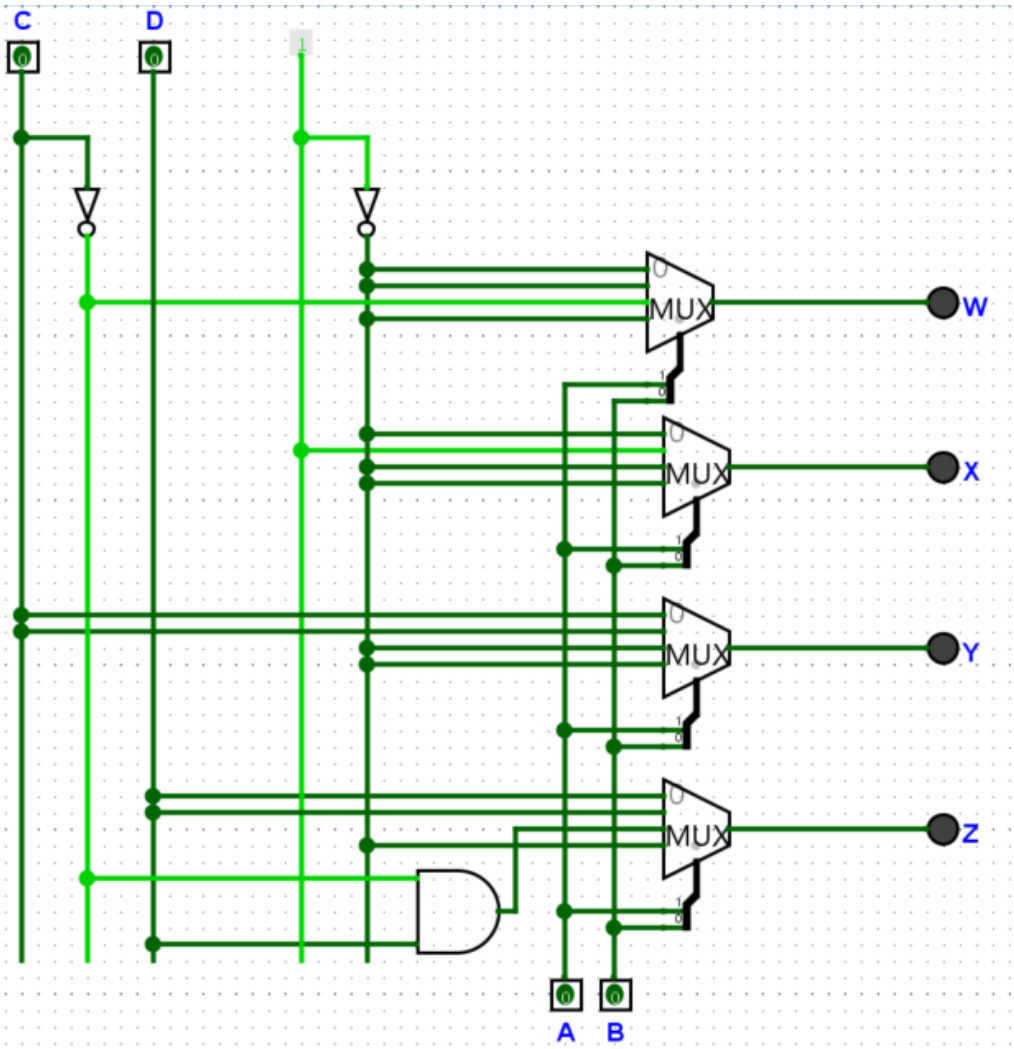
A	B	C	D		Y	Y(D)	Y(C,D)	Y(B,C,D)
0	0	0	0		**0**	0	C	C
0	0	0	1		**0**			
0	0	1	0		**1**	1		
0	0	1	1		**1**			
0	1	0	0		**0**	0	C	
0	1	0	1		**0**			
0	1	1	0		**1**	1		
0	1	1	1		**1**			
1	0	0	0		**0**	0	0	0
1	0	0	1		**0**			
1	0	1	0		**0**	0		
1	0	1	1		**0**			
1	1	0	0		**0**	0	0	
1	1	0	1		**0**			
1	1	1	0		**0**	0		
1	1	1	1		**0**			

iv. Partitioning for $\boldsymbol{Z}$

A	B	C	D		Z	Z(D)	Z(C,D)	Z(B,C,D)
0	0	0	0		**0**	D	D	D
0	0	0	1		**1**			
0	0	1	0		**0**	D		
0	0	1	1		**1**			
0	1	0	0		**0**	D	D	
0	1	0	1		**1**			
0	1	1	0		**0**	D		
0	1	1	1		**1**			
1	0	0	0		**0**	D	$\bar{C}D$	$\bar{B}\bar{C}D$
1	0	0	1		**1**			
1	0	1	0		**0**	0		
1	0	1	1		**0**			
1	1	0	0		**0**	0	0	
1	1	0	1		**0**			
1	1	1	0		**0**	0		
1	1	1	1		**0**			

Therefore, for type 2 implementation:

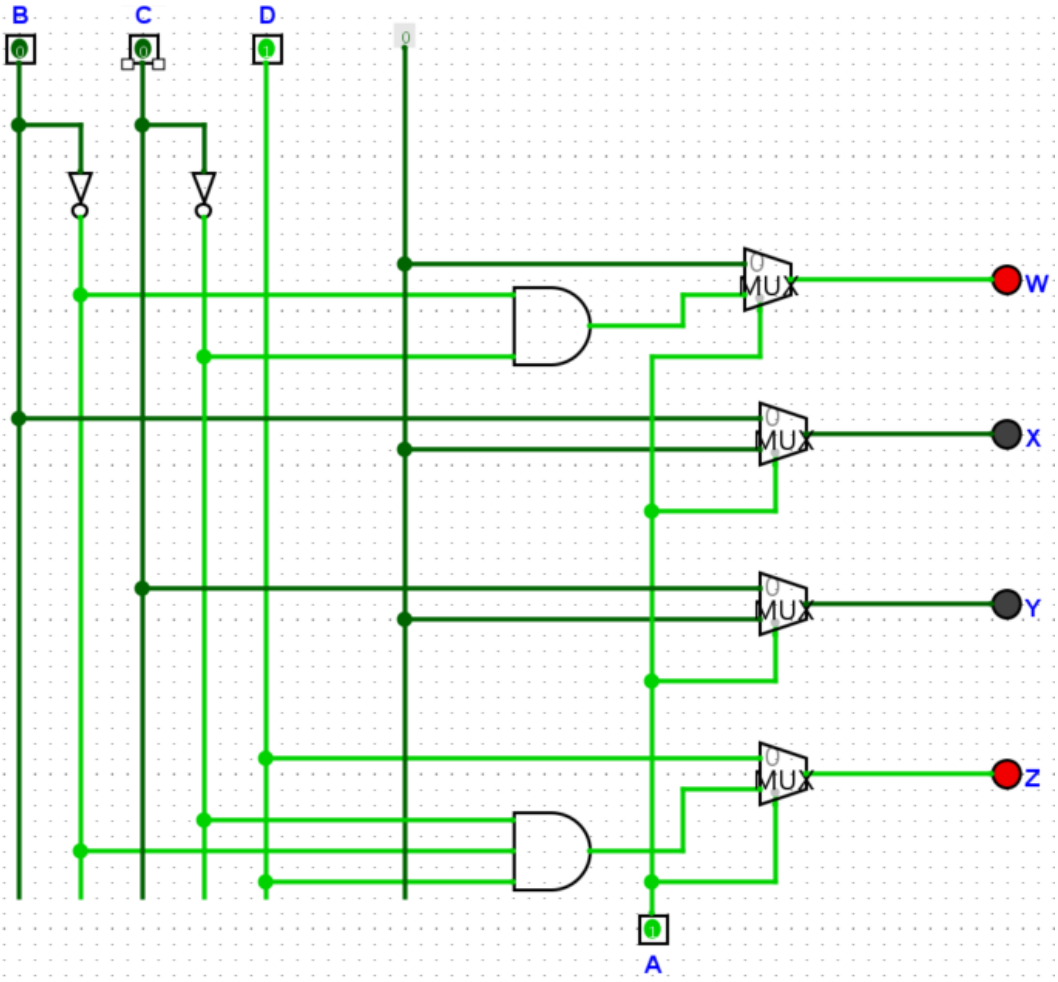
- select lines are ***A*** and ***B***, and
- ***C*** and ***D*** passed through combinational logic and fed into the 4-to-1 MUX lines for each output (***W(C,D), X(C,D), Y(C,D)***, and ***Z(C,D)***). The functions are obtained from the green columns.



Type 2 implementation

For illustration, for type 3 implementation:

- select lines are ***A***, and
- ***B, C*** and ***D*** passed through combinational logic and fed into the 2-to-1 MUX lines for each output (***W(B,C,D), X(B,C,D), Y(B,C,D)***, and ***Z(B,C,D)***). The functions are obtained from the green columns.



Type 3 implementation (using last column)

Question 4

Code to mark states from sequential circuit: S_1S_0

The code marks the following actual states. For simplicity, the states are represented by the symbols ***A, B, C*** and ***D***.

States from sequential circuit

S_1	S_0	State	Symbol
0	0	Main Green, Side Red	***A***
0	1	Main Yellow, Side Red	***B***
1	1	Main Red, Side Green	***C***
1	0	Main Red, Side Yellow	***D***

A truth table for decoding $\boldsymbol{S_1S_0}$ is shown below.

Decoding the states

Code			States			
S_1	S_0		***A***	***B***	***C***	***D***
0	0		1	0	0	0
0	1		0	1	0	0
1	1		0	0	1	0
1	0		0	0	0	1

$$A = \overline{S_1}\,\overline{S_0}$$

$$B = \overline{S_1}S_0$$

$$C = S_1S_0$$

$$D = S_1\overline{S_0}$$

Now we can use the decoded states to switch traffic lights on and off. For this, we will have a truth table for switching the street lights using the states (***A, B, C*** and ***D***). The symbols for the street lights are given in the table below.

Street light symbols

Symbol	Light
MR	main red
MY	main yellow
MG	main green
SR	side red
SY	side yellow
SG	side green

Truth table (states as input, street lights (on or off) as output)

states					**lights**					
A	***B***	***C***	***D***		***MR***	***MY***	***MG***	***SR***	***SG***	***SY***
1	0	0	0		0	0	1	1	0	0
0	1	0	0		0	1	0	1	0	0
0	0	1	0		1	0	0	0	0	1
0	0	0	1		1	0	0	0	1	0
others					*X*	*X*	*X*	*X*	*X*	*X*

Using K-Maps, the functions for switching the lights are found to be:

$$MR = \bar{A}\,\bar{B}$$

$$MY = B$$

$$MG = A$$

$$SR = \bar{C}\,\bar{D}$$

$$SY = D$$

$$SG = C$$

Generation of long (***LT***) and short trigger (***ST***) signals (both cannot occur at the same time, so we can generate the long trigger signal and its complement becomes the short trigger signal).

Truth table for ***LT***

Code			**Trigger**
S_1	S_0		***LT***
0	0		1
0	1		0
1	1		1
1	0		0

$$LT = \overline{S_1 \oplus S_0} = S_1 \odot S_0$$

$$XT = S_1 \oplus S_0$$

Designing the sequential circuit for the states

Here we need:

- State transition table (to obtain the circuit),
- Triggering circuit (to get the pulses),

Truth table for the sequential circuit for the states ($S_1S_0 = Q_1Q_0$) using D flip-flops.

PS		NS			inputs	
Q_1	Q_0	Q_1^+	Q_0^+		D_1	D_0
0	0	0	1		0	1
0	1	1	1		1	1
1	1	1	0		1	0
1	0	0	0		0	0

$$D_1 = Q_0$$

$$D_0 = \overline{Q_1}$$

Flip-flop triggering pulses

For this, we need to consider:

- Input from vehicle sensor
- Output from the long and short timers

The truth table may be made as follows (Long timer output (***L***) and vehicle sensor output (***V***) are included). ***L*** is HIGH when the count reaches the timer duration and low otherwise. It stays HIGH until reset by change of state. ***V*** is HIGH when there is a vehicle in the side street and LOW otherwise.

Triggering to change from states with Green (***A*** and ***C***)

Inputs					Output
vehicle	state		timer		trigger
V	***A***	***C***	***L***		T_1
0	1	0	0		0
0	1	0	1		0
0	0	1	0		1
0	0	1	1		1
0	0	0	0		0
0	0	0	1		0
1	1	0	0		0
1	1	0	1		1
1	0	1	0		0
1	0	1	1		1
1	0	0	0		0
1	0	0	1		0
others					x

Using a K-map, we get

$$T_1 = \bar{V}\ C + C\ L + VA\ L$$

Triggers to transition out of states ***B*** and ***D*** do not require input from the ***V*** and ***L***. We require the output for the short timer, ***S***, which gets *HIGH* when the short count is reached and stays *HIGH* until the sequential circuit changes state.

Triggering to transition from the 'orange' states (from ***B*** and ***D***)

state		**timer**		**trigger**
B	***D***	***S***		T_2
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		x
1	1	1		x

Again, using a K-map,

$$T_2 = DS + BS$$

A multiplexer can be used to select one of the two triggers, depending on the state. The ***LT*** signal is be used to select between the two. The output of the multiplexer is then ANDed with the external clock pulses to enable or disable triggering of the flipflops.

Timing circuits

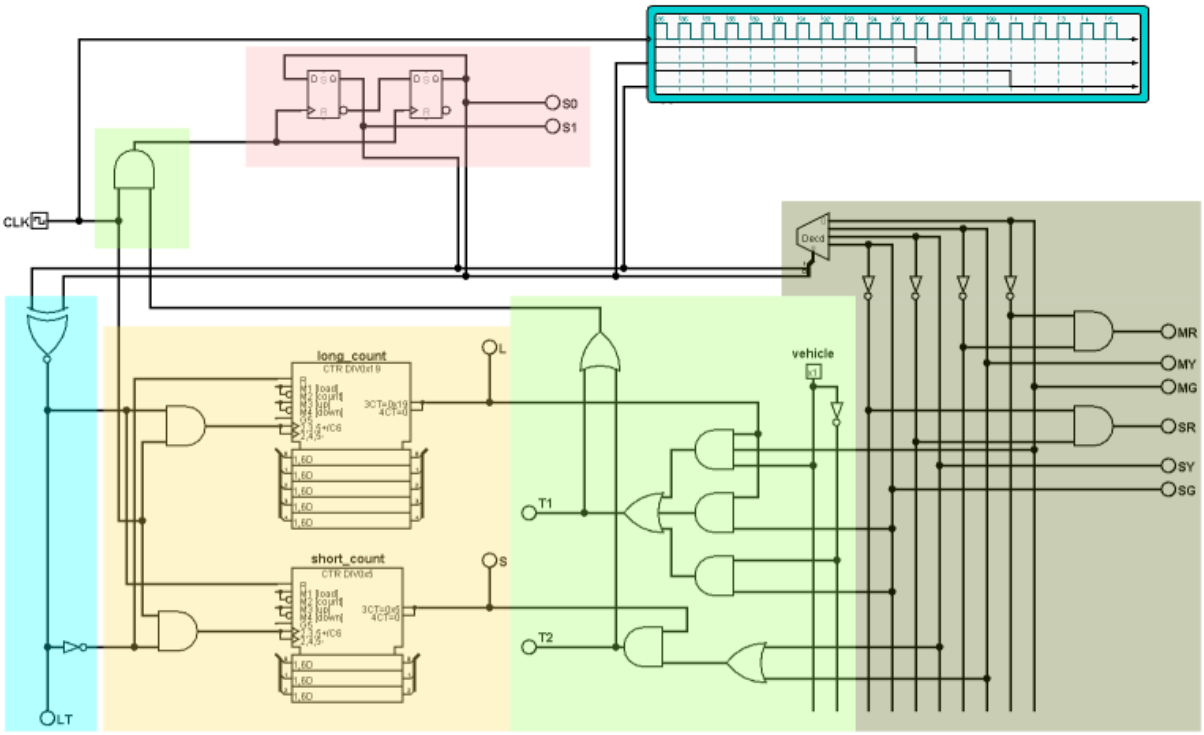
For this circuits, two timers are required:

- 25 s long timer (we use a count of 25 to implement this for illustration), and
- 4 s short timer (we use a count of 4 to implement this for illustration).

Two timers are used for the timing circuit. The long timer counts to 26 and stays at 26 until it is reset when the short timer takes over. When one timer is on, the other is disabled. The triggering pulse comes from $\boldsymbol{LT}$ and it is used to

- reset the timer that is not in use**,** and
- enable the external clock pulses to the timer that is in use.

The combined circuit for Question 4 is shown below. Various signals are marked using named LEDs. When the ‘vehicle’ button is ON, there are vehicles in the side street, and when it is OFF, there are no vehicles no vehicles in the side street. When there are no vehicles in the side street, the main street lights stay green until a vehicle is detected in the side street. Otherwise, it stays on for the duration of the long count.



State sequencing circuit

Combinational circuit for decoding the states and switching the lights

Long and short trigger circuit (to activate the timers)

Combinational logic to trigger state sequencing circuit

Timing circuit (with clocking and reset logic)

The overall circuit with various sections highlighted. This is one way of getting an implementation.