



Digital Controller for Pedestrian Crossing and Traffic Lights

Project Objective: -

To design and simulate, a digital controller for traffic and pedestrian lights at a pedestrian crossing using Microsim Pspice[®]. The controller must be based on next-state techniques using D-type flip-flops and multiplexers.

Specification: -

- 1) The lights will show red to pedestrians and green to traffic in the idle state.
- 2) When the pedestrian wishes to use the crossing he/she presses a button, the traffic lights turn amber and the pedestrian lights remain red.
- 3) After four seconds, the traffic lights turn from amber to red, and the pedestrian lights remain red.
- 4) After two seconds, the traffic lights remain red, while the pedestrian lights turn green and a buzzer is sounded.
- 5) After eight seconds, the pedestrian lights turn from green to amber and to buzzer changes pitch.
- 6) After four seconds the pedestrian lights change from amber to red, the buzzer becomes silent, and at this stage, the traffic lights still remain red.
- 7) After a further two seconds, the system is returned to the idle state once again, having the traffic lights green and the pedestrian lights red.

Parts used for project: -

Quite a number of parts had to be used for this project; each traffic light and pedestrian light needs a 16 to 1 multiplexer. The counter logic needs 4 multiplexers and 4 D-type flip-flops. The clear (or reset) function for this controller also needs a multiplexer. Due to the fact that the multiplexers used are 16 to 1 multiplexers, every output from each multiplexer needs to go through an inverter since 74150's invert their output. The switching circuit also needs a D-type flip-flop and a +5 D.C. power supply and due to the inflexibility of Pspice, two switches had to be introduced into the switching circuit – one to turn on at a certain time and one to turn off shortly after that interval. Below shows the response expected for the controller (P = switch) if the switch was pressed after 4.5 seconds, the buzzers are excluded from this and will be explained at the end of this report.

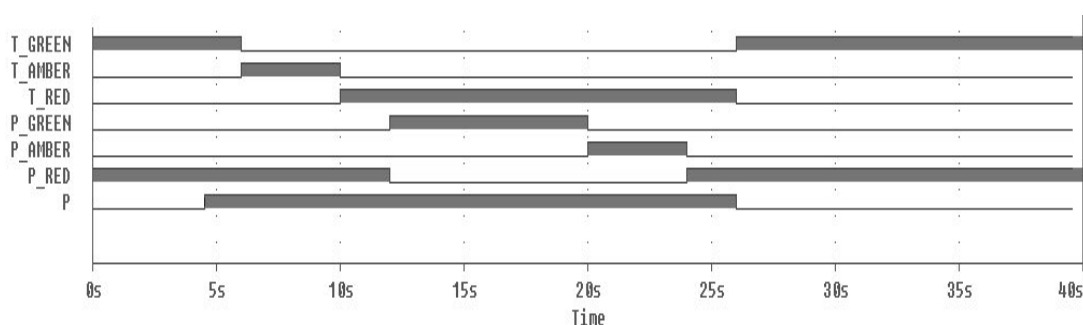
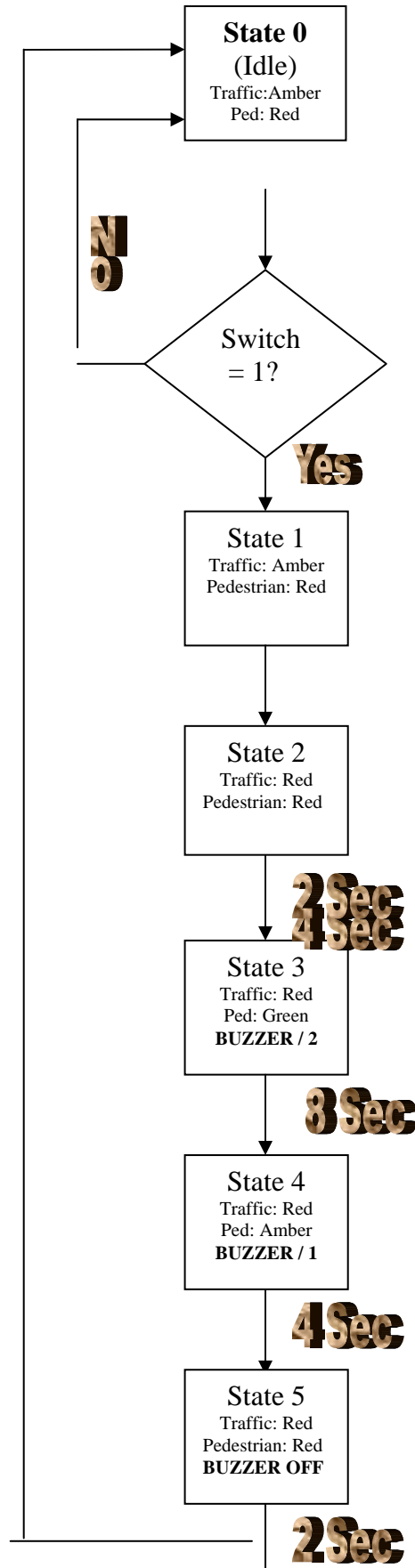


Figure 1:

Expected Responses for each Output (excluding Buzzers)

Initial Flowchart for Pedestrian / Traffic Lights Controller



Digital Controller for Pedestrian Crossing and Traffic Lights

This flowchart is the ideal flowchart for the sequence needed, but due to the limitations of being only allowed to use a limited amount of components a revised flowchart had to be introduced.

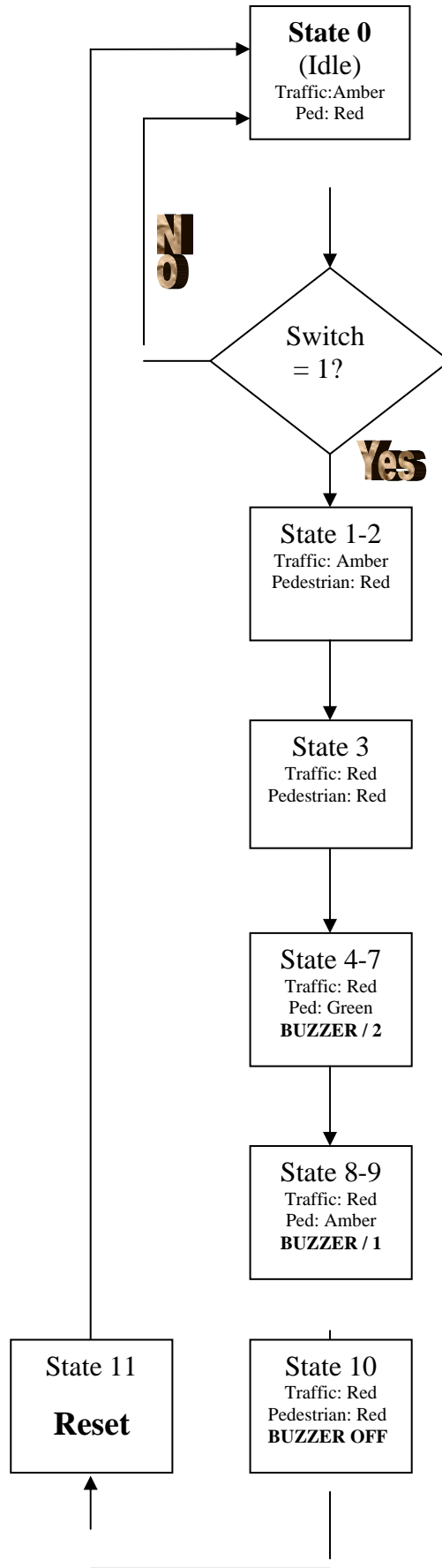
A number of parameters had to be considered:

- 1) Only a simple clock pulse can be used, therefore more states had to be introduced to accommodate time delays (by repeating functions for a set number of clock pulses).
- 2) Another additional state had to be introduced to reset the counter logic back to 0 (and also to reset the switch to 0) after all traffic light states have been executed.
- 3) The simple clock pulse is set to give a positive edge triggered after every 2 seconds so as to half the number of states needed for this operation.
- 4) When the switch is pressed, the state one will not be activated until the next positive edge is triggered from the clock pulse, this delay could vary from a microsecond up to nearly two seconds.
- 5) The buzzer cannot be simulated in Pspice so only the input for the buzzer can be simulated, this clearly shows when the buzzer is activated and at what level.
- 6) Once the switch is activated, the switch is permanently switched on until the switch's D-type flip-flop is reset; no matter how many times the switch is pressed and depressed.

With all these limitations and specifications in mind the revised flowchart for this controller was constructed.

Digital Controller for Pedestrian Crossing and Traffic Lights

Revised Flowchart for Pedestrian / Traffic Lights Controller



Now that the flowchart has been constructed, the truth table can be drawn up.

Digital Controller for Pedestrian Crossing and Traffic Lights

Clk	State	Present State				Next State				Traffic			Pedestrian			Swi (P)	Reset	Buzzer
		Q ₃	Q ₂	Q ₁	Q ₀	D ₃	D ₂	D ₁	D ₀	G	A	R	G	A	R			
0	0	0	0	0	0	0	0	0	P	1	0	0	0	0	1	1	1	0
2	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	X	1	0
4	2	0	0	1	0	0	0	1	1	0	1	0	0	0	1	X	1	0
6	3	0	0	1	1	0	1	0	0	0	0	1	0	0	1	X	1	0
8	4	0	1	0	0	0	1	0	1	0	0	1	1	0	0	X	1	½
10	5	0	1	0	1	0	1	1	0	0	0	1	1	0	0	X	1	½
12	6	0	1	1	0	0	1	1	1	0	0	1	1	0	0	X	1	½
14	7	0	1	1	1	1	0	0	0	0	0	1	1	0	0	X	1	½
6	8	1	0	0	0	1	0	0	1	0	0	1	0	1	0	X	1	1
18	9	1	0	0	1	1	0	1	0	0	0	1	0	1	0	X	1	1
20	10	1	0	1	0	1	0	1	1	0	0	1	0	0	1	X	1	1
22	11	1	0	1	1	0	0	0	0	0	0	1	0	0	1	X	0	0
24	0	0	0	0	0													
Unused States:																		
24	12	1	1	0	0	1	1	0	1	1	0	0	0	0	1	X	0	0
26	13	1	1	0	1	1	1	1	0	1	0	0	0	0	1	X	0	0
28	14	1	1	1	0	1	1	1	1	1	0	0	0	0	1	X	0	0
30	15	1	1	1	1	0	0	0	0	1	0	0	0	0	1	X	0	0

From this truth table, maps were obtained for each counter multiplexer, each traffic light, each pedestrian light, and for reset.

D3	$\bar{Q}_1\bar{Q}_0$ (00)	\bar{Q}_1Q_0 (01)	Q_1Q_0 (11)	$Q_1\bar{Q}_0$ (10)		D2	$\bar{Q}_1\bar{Q}_0$ (00)	\bar{Q}_1Q_0 (01)	Q_1Q_0 (11)	$Q_1\bar{Q}_0$ (10)
$\bar{Q}_3\bar{Q}_2$ (00)	0	0	0	0		$\bar{Q}_3\bar{Q}_2$ (00)	0	0	1	0
\bar{Q}_3Q_2 (01)	0	0	1	0		\bar{Q}_3Q_2 (01)	1	1	0	1
Q_3Q_2 (11)	1	1	0	1		Q_3Q_2 (11)	1	1	0	1
$Q_3\bar{Q}_2$ (10)	1	1	0	1		$Q_3\bar{Q}_2$ (10)	0	0	0	0

Digital Controller for Pedestrian Crossing and Traffic Lights

D1	\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
	(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2 (00)	0	1	0	1
\bar{Q}_3, Q_2 (01)	0	1	0	1
Q_3, Q_2 (11)	0	1	0	1
Q_3, \bar{Q}_2 (10)	0	1	0	1

D0	\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
	(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2 (00)	P	0	0	1
\bar{Q}_3, Q_2 (01)	1	0	0	1
Q_3, Q_2 (11)	1	0	0	1
Q_3, \bar{Q}_2 (10)	1	0	0	1

Karnaugh Maps For Controller

T Green	\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
	(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2 (00)	1	0	0	0
\bar{Q}_3, Q_2 (01)	0	0	0	0
Q_3, Q_2 (11)	0	0	0	0
Q_3, \bar{Q}_2 (10)	0	0	0	0

P Green	\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
	(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2 (00)	0	0	0	0
\bar{Q}_3, Q_2 (01)	1	1	1	1
Q_3, Q_2 (11)	0	0	0	0
Q_3, \bar{Q}_2 (10)	0	0	0	0

T Amber	\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
	(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2 (00)	0	1	0	1
\bar{Q}_3, Q_2 (01)	0	0	0	0
Q_3, Q_2 (11)	0	0	0	0
Q_3, \bar{Q}_2 (10)	0	0	0	0

P Amber	\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
	(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2 (00)	0	0	0	0
\bar{Q}_3, Q_2 (01)	0	0	0	0
Q_3, Q_2 (11)	0	0	0	0
Q_3, \bar{Q}_2 (10)	1	1	0	0

Digital Controller for Pedestrian Crossing and Traffic Lights

T Red	$\bar{Q}_1\bar{Q}_0$	\bar{Q}_1Q_0	Q_1Q_0	$Q_1\bar{Q}_0$	P Red	$\bar{Q}_1\bar{Q}_0$	\bar{Q}_1Q_0	Q_1Q_0	$Q_1\bar{Q}_0$
	(00)	(01)	(11)	(10)		(00)	(01)	(11)	(10)
$\bar{Q}_3\bar{Q}_2$ (00)	0	0	1	0	$\bar{Q}_3\bar{Q}_2$ (00)	1	1	1	1
\bar{Q}_3Q_2 (01)	1	1	1	1	\bar{Q}_3Q_2 (01)	0	0	0	0
Q_3Q_2 (11)	1	1	1	1	Q_3Q_2 (11)	1	1	1	1
$Q_3\bar{Q}_2$ (10)	1	1	1	1	$Q_3\bar{Q}_2$ (10)	0	0	1	1

Reset	$\bar{Q}_1\bar{Q}_0$	\bar{Q}_1Q_0	Q_1Q_0	$Q_1\bar{Q}_0$
	(00)	(01)	(11)	(10)
$\bar{Q}_3\bar{Q}_2$ (00)	1	1	1	1
\bar{Q}_3Q_2 (01)	0	0	0	0
Q_3Q_2 (11)	1	1	1	1
$Q_3\bar{Q}_2$ (10)	1	1	0	1

Digital Controller for Pedestrian Crossing and Traffic Lights

Method of Transforming Karnaugh Maps into Multiplexers

Transforming the truth table to actual logic is simplified using Karnaugh Maps and by setting each input of the multiplexer to the same value of the corresponding number on the Karnaugh map. Also the multiplexers are set the active high. The “Q” inputs of each multiplexer correspond to the outputs of the counter logic D-types. In this example, D3 from the counter logic is used.

D3		\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
		(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2	(00)	0	0	0	0
\bar{Q}_3, Q_2	(01)	0	0	1	0
Q_3, Q_2	(11)	1	1	0	1
Q_3, \bar{Q}_2	(10)	1	1	0	1

Minterm Key

		\bar{Q}_1, \bar{Q}_0	\bar{Q}_1, Q_0	Q_1, Q_0	Q_1, \bar{Q}_0
		(00)	(01)	(11)	(10)
\bar{Q}_3, \bar{Q}_2	(00)	0	1	3	2
\bar{Q}_3, Q_2	(01)	4	5	7	6
Q_3, Q_2	(11)	12	13	15	14
Q_3, \bar{Q}_2	(10)	8	9	11	10

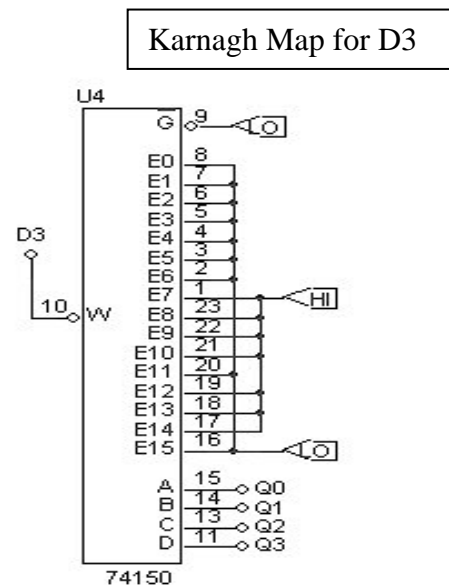


Figure 2: D3 multiplexer

This method was used for each multiplexer in the controller.

Digital Controller for Pedestrian Crossing and Traffic Lights

Now that all the Karnaugh Maps and the truth tables are written up all that has to be done is the simulation of the actual controller. First of all the counter logic had to be designed:

Counter Logic

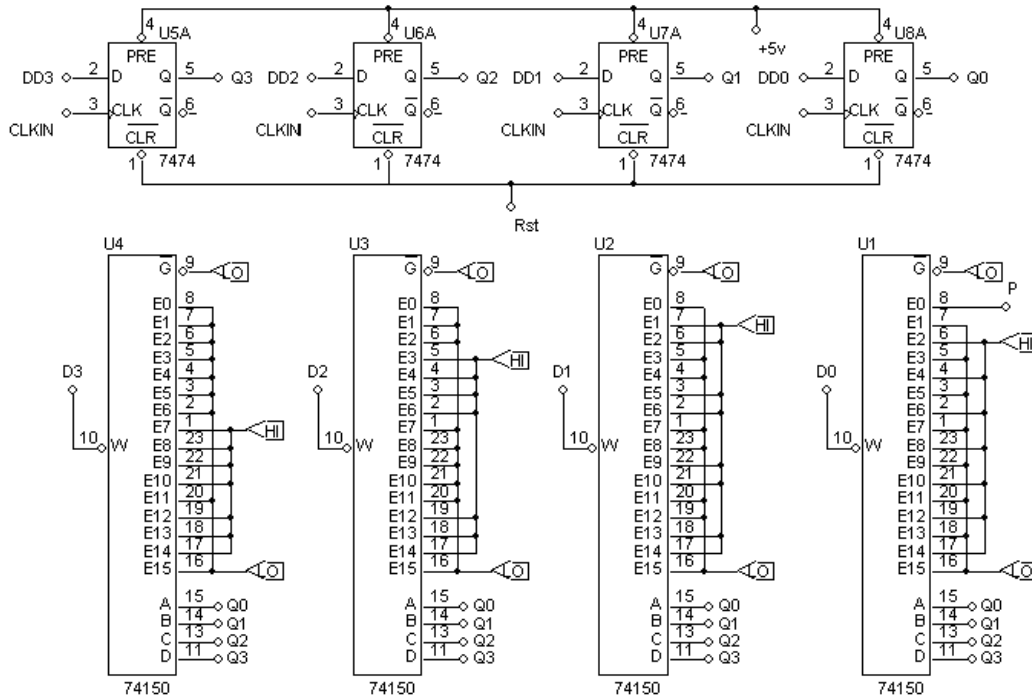
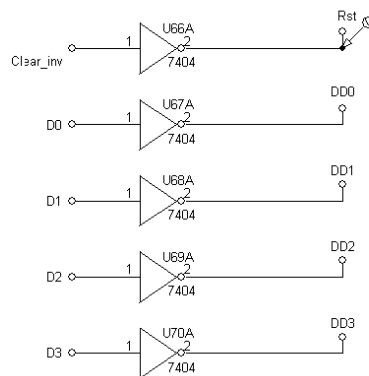


Figure 3: Counter Logic Schematic

Inverting Multiplexer Output



Reset Logic

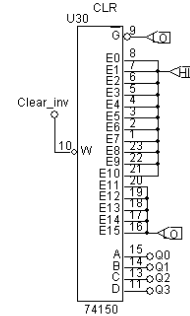
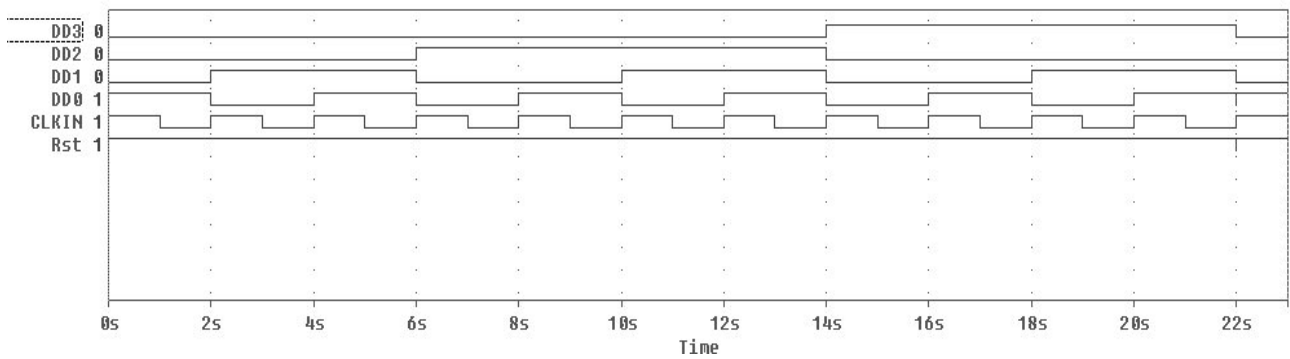


Figure 4: Inverting Multiplexer Outputs and Reset Logic



The counter logic above uses logic hi's and logic low's to represent 0's and 1's. The multiplexers are set to active high to give normal operation and the d-type flip-flops are pre-set to +5 volts. The multiplexer's outputs must be inverted since the device inverts the output. Below shows the counter output response. Note that the counter is reset at 22 seconds. For the moment assume that $P = 1$ (since we are only analysing counter logic) and that the clock is a simple pulse.

Figure 5: Counter Logic Response

Now that the counter logic is operational, the traffic light logic is constructed.

Traffic Light Logic

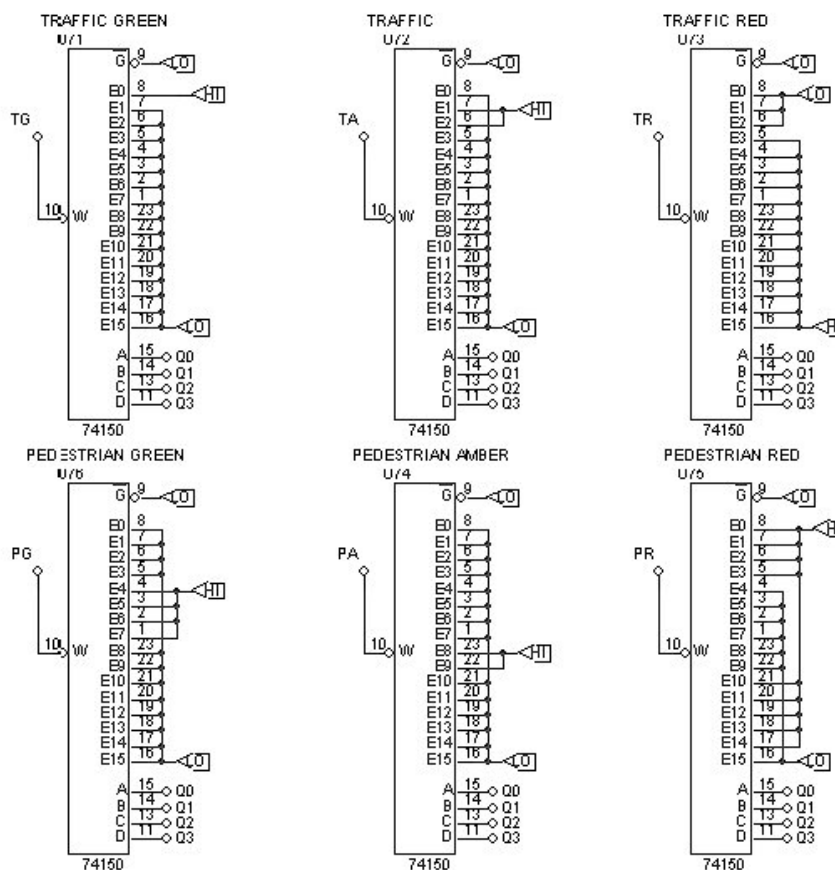


Figure 6: Traffic light logic schematic

As with before, the multiplexer output is inverted and the multiplexers are set to active high. These multiplexers are set in accordance with the maps for each output. The value for P is set equal to "hi" for this response since the switch is not considered yet.

Digital Controller for Pedestrian Crossing and Traffic Lights

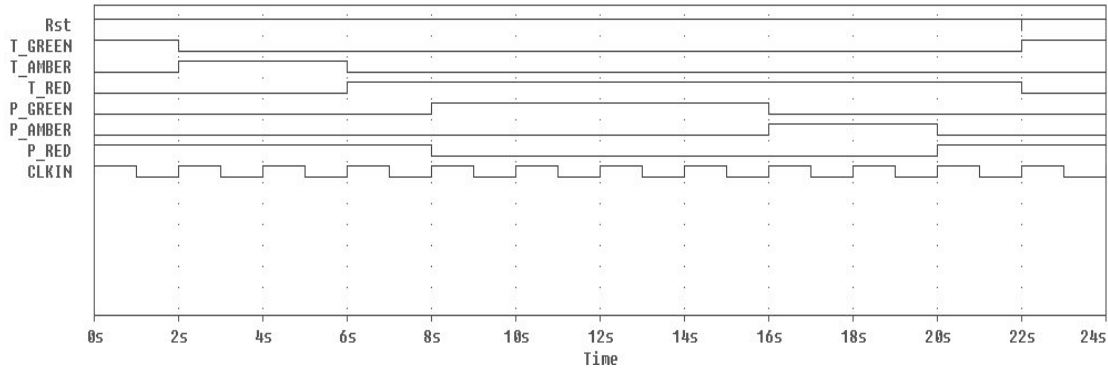


Figure 7: Traffic light logic response

The traffic lights start with the initial condition of having the traffic lights green and the pedestrian lights red, and on the first positive edge of the clock pulse state one is initiated. The logic is initiated step by step by the counter input to each of the multiplexers (Q_n).

Next the actual switch to enable to activation of the sequence is designed.

A de-bouncing switch is constructed using a d-type flip flop. This flip flop is made “high” as soon as the switch is pressed and stays high until it is reset, no matter if the switch is pressed repeatedly or not. The switch is simulated by 2 switches in Pspice, one switch is open until 4.5 seconds and the other switch is closed until 4.6 seconds, therefore simulating a switched being pressed at 4.5 seconds and being held for 0.1 second. Also included in the diagram below is the clock pulse logic, which is fed to the counter d-type, flip-flops.

Clock + Switch

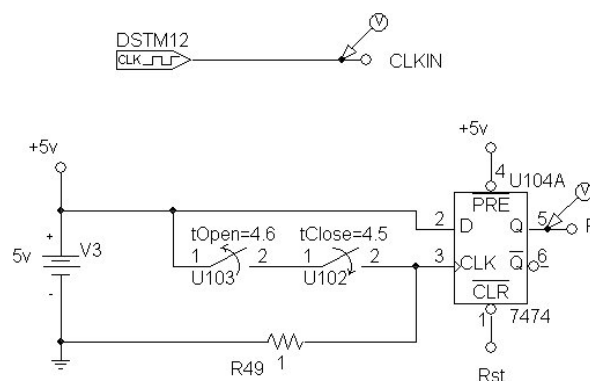


Figure 8: Switch and Clock Logic

The switch works well, but it should be noted that the counter logic will not change immediately when the d-type flip flop is set “high”, but until the next positive edge from the

Digital Controller for Pedestrian Crossing and Traffic Lights

clock pulse is generated, this is demonstrated below. Also shown below is that the counter logic is only activated when the switch is pressed, and is in state 0 at all other times.

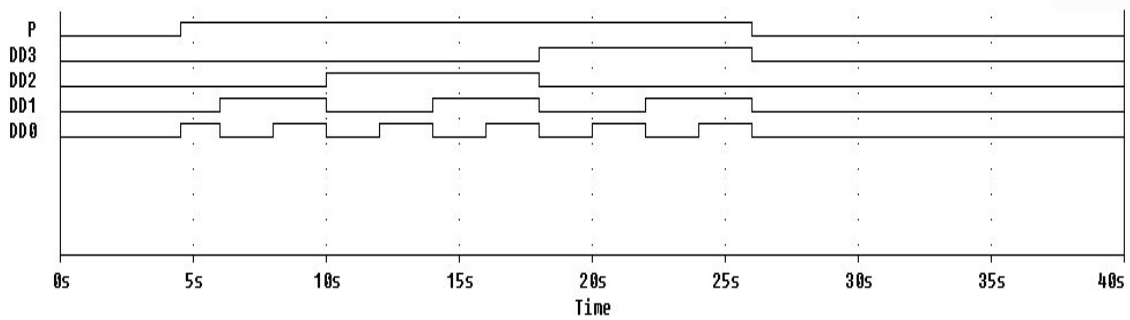


Figure 9: Counter Logic Response

Finally, the buzzers have to be accounted for as well.

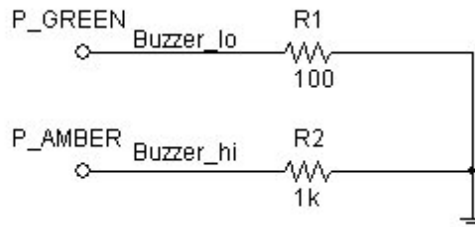


Figure 10: Buzzer Logic

The power for the buzzers was simply taken from the inputs of each of the pedestrian traffic lights that are active when the buzzers are on. For example, the buzzer is sounded at just under 1.5 volts when the pedestrian green light is on, when the buzzer is sounded at just over 3 volts the amber pedestrian light is on, this buzzer is pitched differently so as to alert the pedestrians to the change in colour of lights and to urge them to get off the street.

Digital Controller for Pedestrian Crossing and Traffic Lights

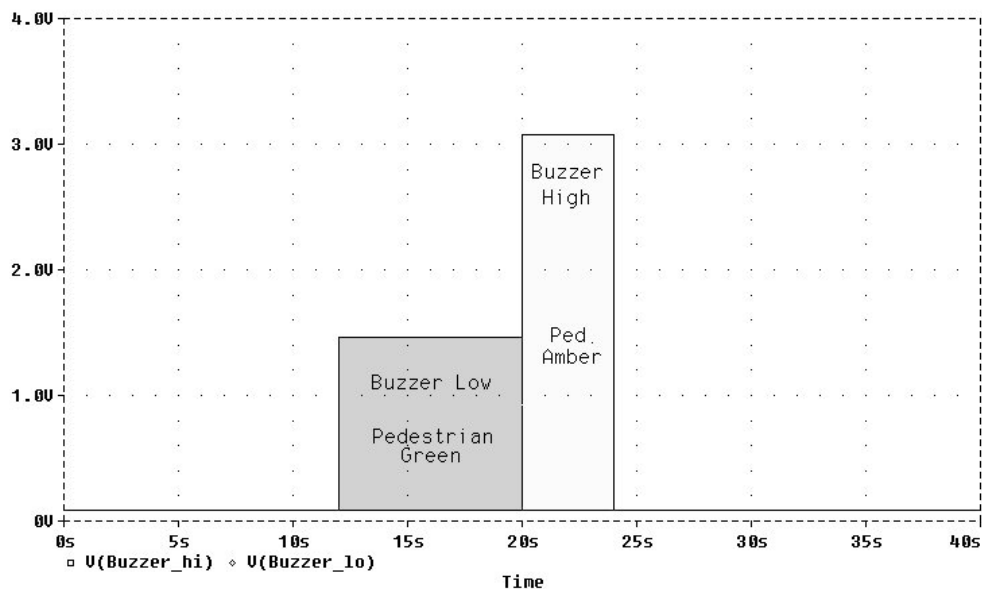


Figure 11: Buzzer Response

Complete Traffic / Pedestrian Light Control System

Conclusion:

This controller works very well and efficiently. The only problem from the design of the controller is that the controller doesn't immediately start the cycle when the switch is closed, but initiates it at the first positive edge pulse from the clock. Apart from this discrepancy, the rest of the circuit works efficiently and safely. The safety aspect of this controller is essential as it deals with people's safety crossing the road. There are eleven set states in this controller, but due to the extra possible states provided by the multiplexers, the extra states should be defined so as to make the pedestrians safe in the event of the controller going out of order and entering an unused state. So in this controller the unused states are set to make all lights (pedestrian & traffic) red, and the counter logic should return the unused state back to state 0.

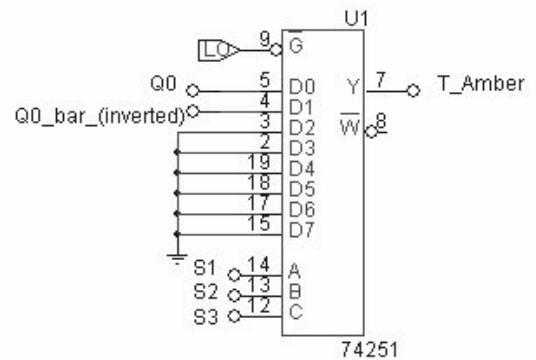
How this controller could be improved.

The specification for this controller makes no restrictions on the type of multiplexer used, but practically, for the construction of the actual controller the multiplexer used should have as little inputs as possible so as to avoid problems with the making of the controller. Below describes how to reduce multiplexer type from 16 to 1, to an 8 to 1 multiplexer.

Digital Controller for Pedestrian Crossing and Traffic Lights

Traffic Amber	Reduced Traffic Amber
0	Q_0
1	
1	$\overline{Q_0}$
0	
0	0
0	
0	0
0	
0	0
0	
0	0
0	
0	0
0	
Unused States:	
0	0
0	
0	0
0	

Reduced Multiplexer



The reduced multiplexer above is for the Traffic Amber Logic, this reduction can be done for each of the traffic and pedestrian multiplexers, but not for the counter logic which have to be left as 16 to 1 multiplexers. Q_0 and $\overline{Q_0}$ are used because Q_0 changes from 0 to 1 in for the two states covered by it's reduction and $\overline{Q_0}$ for the inverse of these values. Also, the least significant select input (Q_0) had to be left out due to the halving of the possible states for the multiplexer, but this does not affect the operation of the controller.