

PSDE5006 Assignment 2

by Daniel Ince

FILE	14401_DANIEL_INCE_PSDE5006_ASSIGNMENT_2_152849_662547893.D OCX		
TIME SUBMITTED	24-APR-2018 12:11AM (UTC+0100)	WORD COUNT	3679
SUBMISSION ID	87274051	CHARACTER COUNT	20980



APRIL 24, 2018

ASSIGNMENT 2

PSDE5006

DANIEL INCE

Contents

1. Introduction	2
2. Preliminary Research	3
Finite state machines	3
Moore and Mealy machine comparison	3
Finite state reduction techniques	4
Flip-flop types	5
3. Design method, choices and calculations	6
Five second delay counter	7
Fifteen second counter	9
Controller	11
4. Testing and improving the design	15
5. Conclusions	20
6. Reference list and bibliography	21
Works Cited	21
7. Annexe	22

1. Introduction

This report is to document the design and construction of a traffic light controller using multisim software to upload the program to a Nexys Digilent 4 DDR FPGA board to test. This exercise will involve investigating finite state machines and applying the knowledge to the circuit design. Once designed and implemented, the system will be tested, including the individual testing of each part of the circuit to ensure they perform as intended individually. Following the testing, conclusions will be drawn up to evaluate the performance and recommendations made for improving the circuit.

2. Preliminary Research

Finite state machines

Finite state machines are designed to have a pre-determined set of states, the state which the machine is currently in depends on the previous state and the input. Therefore a FSM can be designed to have an individual state for every possible past state which the machine has been in. This is summarised in *I-Programmer*, "What this means that the entire history of the machine is summarized in its current state. All that matters is the state that it is in and not how it reached this state." (James, 2018). This explains how a machine can constantly be changing states based on its previous state and the current input, as the current state is only achieved through progressing past the previous state in response to an input to get to the current.

The website also states how a finite state machine is used to solve problems in programming by storing a series of state input tables. The program looks for the correct table cell in relation to the current input then moves to that state, which opens up the next input table specific to that state.

Moore and Mealy machine comparison

A Moore machine is a form of finite state machine which includes the output inside the state itself. A Mealy machine does not do this and includes the output within the transition to the next state.

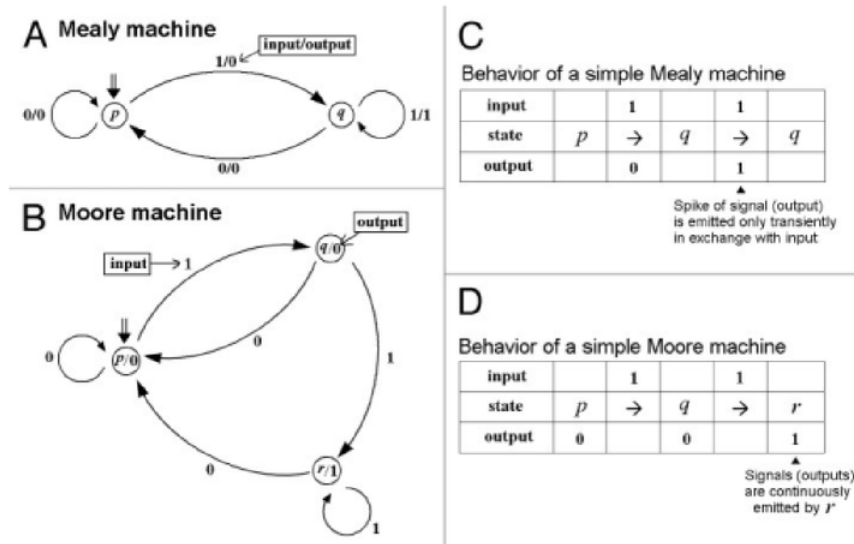


Figure 1: Moore and Mealy comparison.

Figure 1 shows an example of the differences between the two, taken from the website; *US National Library of Medicine*. The Mealy machine outputs during the transition as shown in the diagram and in the behaviour table. The Moore machine outputs during a current state. The state diagrams show that the Moore machine consequently has one more state than the Mealy, this is because the final output is not a state itself and instead occurs during the transition back to the starting state. This is the largest advantage of the Mealy machine as the same outcome can be achieved in fewer states, meaning less program language, circuitry and memory is needed to achieve the same computational power. (Kawano, et al., 2012)

Figure 1 also shows how the Moore machine will continually emit the output signal when in that state, whereas the Mealy will only briefly output during the state transition period. As this period is

triggered by an input, it can be said that the Mealy machine output depends on the current state and the current inputs. Whereas the Moore machine output only depends on the current state, as during that output state it will constantly output regardless of the input. For this reason, the counters developed during this assignment are Moore machines as the output only changes with the addition of the clock pulse to trigger the state change and increment of the counter, it does not instantly change when the input changes.

As stated in *Sequential Implementation*; "In Mealy machines, input change can cause output change as soon as logic is done – a big problem when two machines are interconnected – asynchronous feedback. Mealy Machines react faster to inputs. React in same cycle – don't need to wait for clock. In Moore machines, more logic may be necessary to decode state into outputs – more gate delays after." (University of California, 2005)

Another level above finite state machines are what is known as 'Pushdown Automata'. These are a form of state machine which read, write and remove symbols to the top of a memory stack. The states can be affected by what is currently at the top of the stack. A more complex version of this with more computing power is known as a Turing machine.

Figure 2 shows how the PDA takes an input from the stack and then pushes or reads and removes the top value from the memory stack. Once the PDA has read all of the input tape it reaches its final state. (Tutorials Point, n.d.)

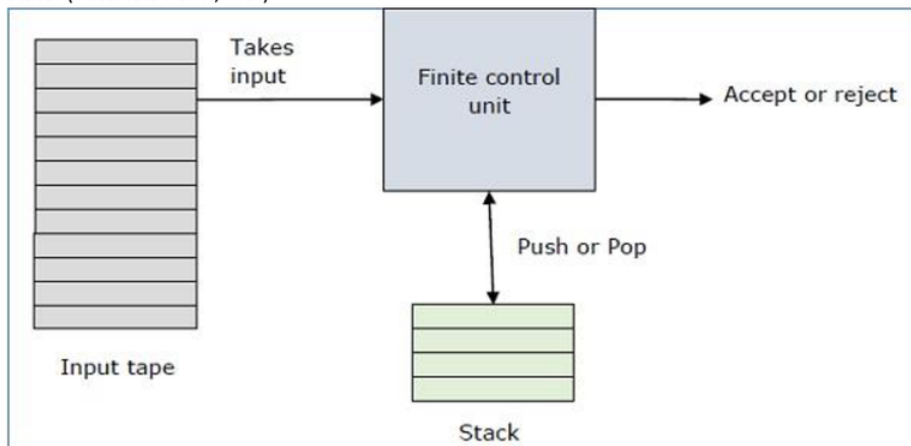


Figure 2: Pushdown Automata

Finite state reduction techniques

In order to optimise finite state machines, it is beneficial to reduce the amount of states and logic gates needed.

A row matching method can be performed when drawing up the state transition table, the table may highlight certain rows which although in different states, achieve the same output. All of these states can be combined into a new single state as they are equivalent. (Katz, 1993) Essentially this method cuts down on the states which 'have no effect' and the states which all have the same outcome by combining them each into one state.

Flip-flop types

A variety of flip-flop types can be used when implementing a state machine, which one depends on the application of the FSM. As stated in *State Machine Design*, D-type flip flops are more appropriate for smaller FSM designs as they include hold terms for logic HIGH. However they only flip from LOW to HIGH. Other types of flip-flop such as JK, SR and T types can flip both LOW to HIGH and HIGH to LOW, however these do not allow for holding the logic values. (AMD, 1993)

As explained, a D-type flip-flop can be used where the design is less complex as the HIGH to LOW switching will not be required as much, and the logic HIGH hold feature can be used. The other flip-flop types can be used in more complex designs in order to reduce the amount of flip-flops needed, because they perform both the stated functions, and the hold feature is less of a priority.

3. Design method, choices and calculations

The traffic light controller consists of three main blocks, a five second delay, fifteen second counter and a controller.

The states of the traffic light controller are as follows:

- Remain green.
- Wait button is pressed. Remain green for five seconds.
- Change to amber for a second.
- Change to red. Wait fifteen seconds.
- Change to red/amber for a second.
- Change back to green. Remain and wait for next input.

To achieve this, a finite state machine must be designed with the minimum amount of logic gates. Flip flops will be used as the logic gates and the Karnaugh mapping technique will be used to cut down the amount of logic required.

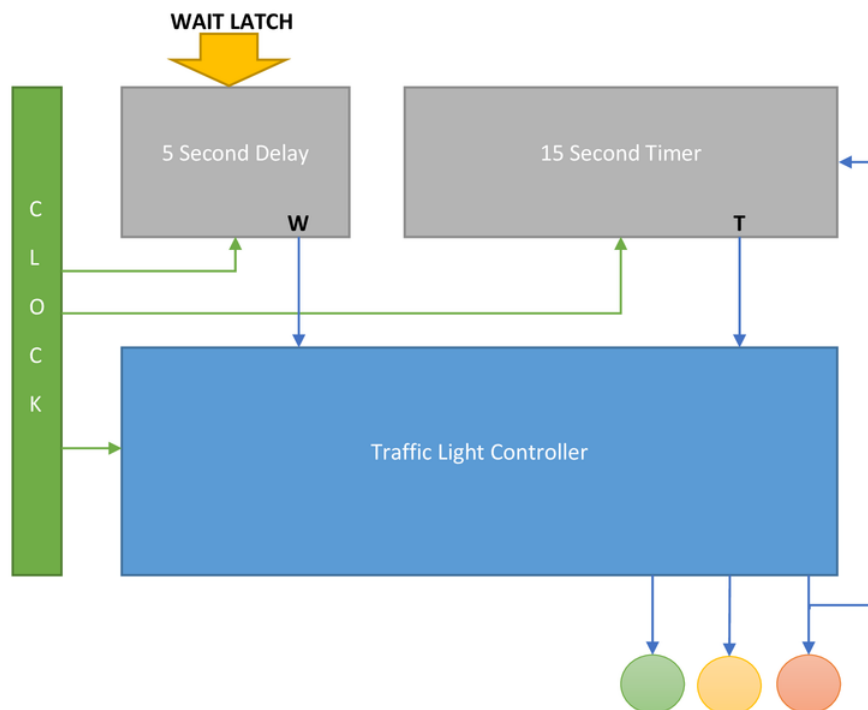


Figure 3: Pelican crossing block diagram

Figure 3 shows a block diagram of how the crossing is to function, each state machine is to work individually but together they perform as one state machine. First, the wait latch is activated, which triggers the 5 second delay counter, once this reaches five, it will output 'W' to the traffic light controller. The controller will then be prompted to cycle through its states until it reaches RED, where it will also output to the 15 second counter to start its sequence. Once fifteen seconds has

elapsed, the output T will be sent to the controller. The controller then cycles through until back at green, where it sits and waits for another input from the 5 second delay counter.

Five second delay counter

To design a five second counter, a state diagram and table must be first drawn, as there are 6 required states, three flip flops will be needed.

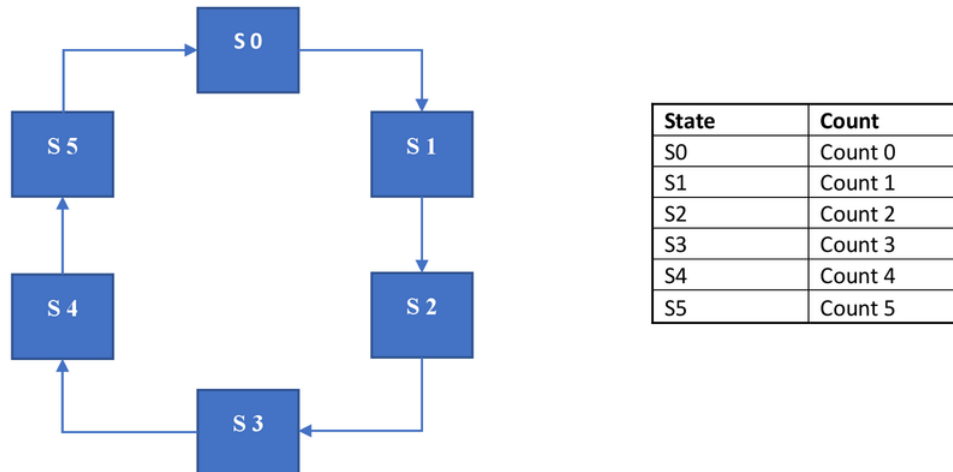


Figure 4: State diagram and table.

Present	Next	Present			Next		
		Qa	Qb	Qc	Qa+	Qb+	Qc+
S0	S1	0	0	0	0	0	1
S1	S2	0	0	1	0	1	0
S2	S3	0	1	0	0	1	1
S3	S4	0	1	1	1	0	0
S4	S5	1	0	0	1	0	1
S5	S0	1	0	1	0	0	0
S6	X	1	1	0	X	X	X
S7	X	1	1	1	X	X	X

The resulting table above displays the next states and the unused states as the counter counts to five before resetting back to zero. In order to cut down on the logic, Karnaugh mapping will be performed.

[Qa+]					
A\BC	00	01	11	10	
0	0	0	1	0	$Qa+ = BC + AC'$
1	0	0	X	X	
[Qb+]					
A\BC	00	01	11	10	
0	0	1	0	1	$Qb+ = A'B'C + BC'$
1	0	0	X	X	
[Qc+]					
A\BC	00	01	11	10	
0	1	0	0	1	$Qc+ = C'$
1	1	0	X	X	

Above show the Karnaugh maps and the resulting shortened logic expressions. From here, the program multisim was used to replicate the logic using D-type flip flops.

Figure 5 below displays the circuit having been built. The input 'BtnL' is used as the 'WAIT' button.

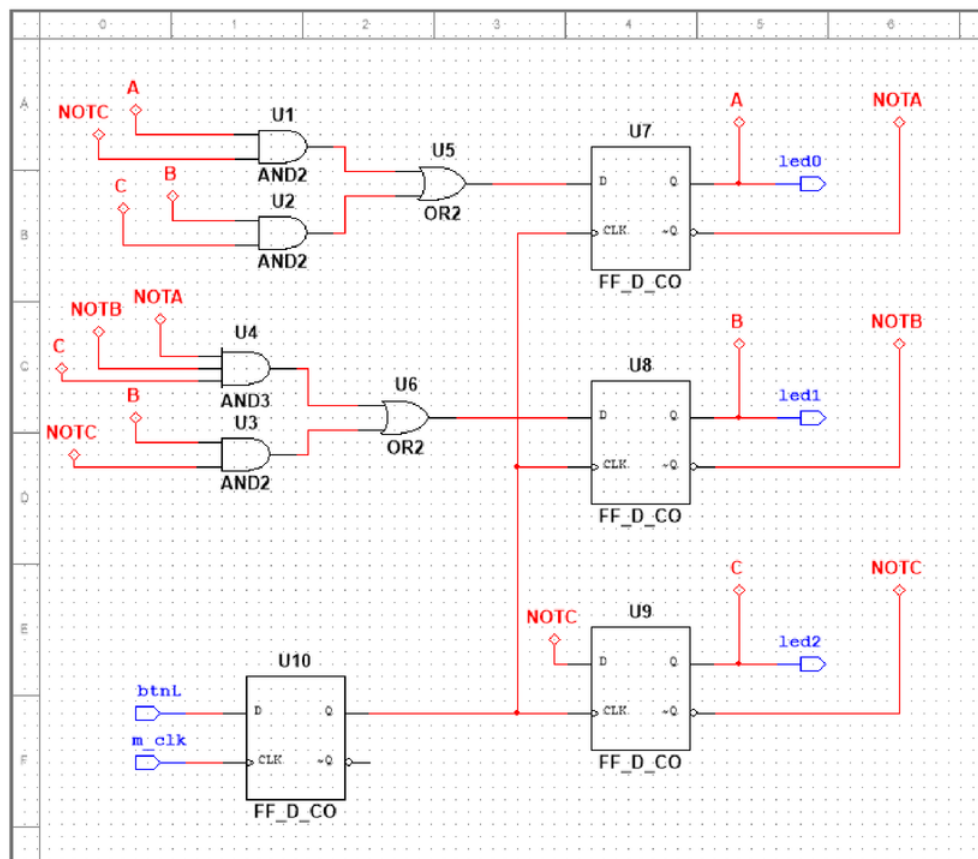


Figure 5: Five second delay counter

The output LEDs in this instance are used to test the circuit, on the final design they will be replaced by 'W'.

Fifteen second counter

The 15 second counter is used to allow pedestrians to cross the road while the light is switched to red, once this has elapsed, it will output the value 'T' to the traffic light controller. To design this counter a state diagram will be produced along with a table. In this instance, 16 states will be needed, so using the rule, a total of four flip flops will be required.

Present	Next	Present				Next			
		Qa	Qb	Qc	Qd	Qa+	Qb+	Qc+	Qd+
S0	S1	0	0	0	0	0	0	0	1
S1	S2	0	0	0	1	0	0	1	0
S2	S3	0	0	1	0	0	0	1	1
S3	S4	0	0	1	1	0	1	0	0
S4	S5	0	1	0	0	0	1	0	1
S5	S6	0	1	0	1	0	1	1	0
S6	S7	0	1	1	0	0	1	1	1
S7	S8	0	1	1	1	1	0	0	0
S8	S9	1	0	0	0	1	0	0	1
S9	S10	1	0	0	1	1	0	1	0
S10	S11	1	0	1	0	1	0	1	1
S11	S12	1	0	1	1	1	1	0	0
S12	S13	1	1	0	0	1	1	0	1
S13	S14	1	1	0	1	1	1	1	0
S14	S15	1	1	1	0	1	1	1	1
S15	S0	1	1	1	1	0	0	0	0

Above is the state table for the counter, the state diagram for this is the same as the five second counter, except there are 16 states from 0 to 15.

[Qa+] AB\CD	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	0	1
10	1	1	1	1
[Qb+] AB\CD	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0		0
[Qc+] AB\CD	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1
[Qd+] AB\CD	00	01	11	10

$$Qa+ = AC' + A'BCD + AB' + AD'$$

$$Qb+ = BC' + BD' + B'CD$$

$$Qc+ = C'D + CD'$$

00	1	0	0	1	$Qd+ = D'$
01	1	0	0	1	
11	1	0	0	1	
10	1	0	0	1	

When using flip flops in Multisim to create the logic, the circuit developed is as shown in figure 6.

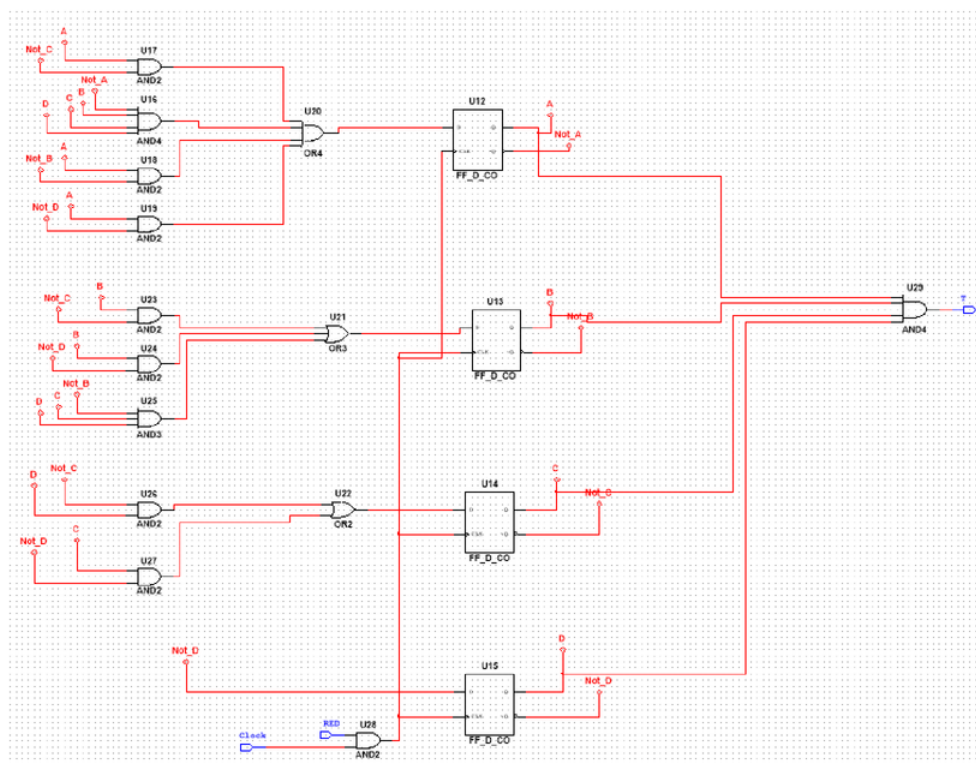


Figure 6: 15 second counter

The output of all four logic gates are sent into an 'AND' gate to output the value of 'T'. This will trigger the controller to continue its sequence once the pedestrian has finished crossing during the 15 second red light state.

Controller

The traffic light controller can only transition certain states through the use of inputs, else it will just remain in the same state regardless. The state diagram shown in figure 7 displays the states and which inputs are required for the dependant states to transition.

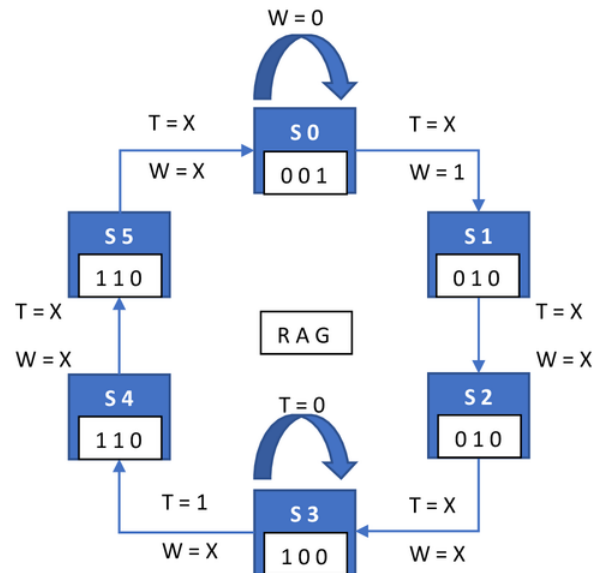


Figure 7: Controller state diagram

Figure 7 shows that in order for S0 to change to S1, the input W is required, also, for S3 to progress, an input from T is needed. A state table must be drawn and the logic expressions for the state machine and the Red, Amber and Green outputs worked out.

Present		W = 0		W = 1	
		T = 0	T = 1	T = 0	T = 1
S0		S0	S0	S1	S1
S1		S2	S2	S2	S2
S2		S3	S3	S3	S3
S3		S3	S4	S3	S4
S4		S5	S5	S5	S5
S5		S0	S0	S0	S0

Present		W = 0		W = 1		Output
		T = 0	T = 1	T = 0	T = 1	
		QA+, Qb+, Qc+	QA+, Qb+, Qc+	QA+, Qb+, Qc+	QA+, Qb+, Qc+	R, A, G
S0		000	000	001	001	001
S1		010	010	010	010	010
S2		011	011	011	011	010
S3		011	100	011	100	100
S4		101	101	101	101	110
S5		000	000	000	011	110

The table on the previous page shows the state transition table for the controller, including all instances with the additional inputs W and T, the Red, Amber, Green output table is also shown for each state.

[Qa+] TA\BC	00	01	11	10	[Qa+] TA\BC	00	01	11	10
00	0	0	0	0	00	0	0	0	0
01	1	0	X	X	01	1	0	X	X
11	1	0	X	X	11	1	0	X	X
10	0	0	1	0	10	0	0	1	0
[Qb+] TA\BC	00	01	11	10	[Qb+] TA\BC	00	01	11	10
00	0	1	1	1	00	0	1	1	1
01	0	0	X	X	01	0	0	X	X
11	0	0	X	X	11	0	0	X	X
10	0	1	0	1	10	0	1	0	1
[Qc+] TA\BC	00	01	11	10	[Qc+] TA\BC	00	01	11	10
00	0	0	1	1	00	1	0	1	1
01		0	X	X	01	1	0	X	X
11		0	X	X	11	1	0	X	X
10	0	0	0	1	10	1	0	0	1

[RED]A\BC	00	01	11	10	
0	0	0	1	0	Red = A + BC
1	1	1	X	X	
[AMBER] A	00	01	11	10	
0	0	1	0	1	Amber = B'C + BC' + A
1	1	1	X	X	
[Green] A\B	00	01	11	10	
0	1	0	0	0	Green = A'B'C'
1	0	0	X	X	

$$Qa+ = AC' + TBC$$

$$Qb+ = A'B'C + T'B + B'C$$

$$Qc+ = AB'C' + WC' + BC' + T'B$$

Above are the Karnaugh maps for the controller and the outputs, along with the resulting expressions.

Figure 8 shows the resultant circuit built in Multisim with flip flops.

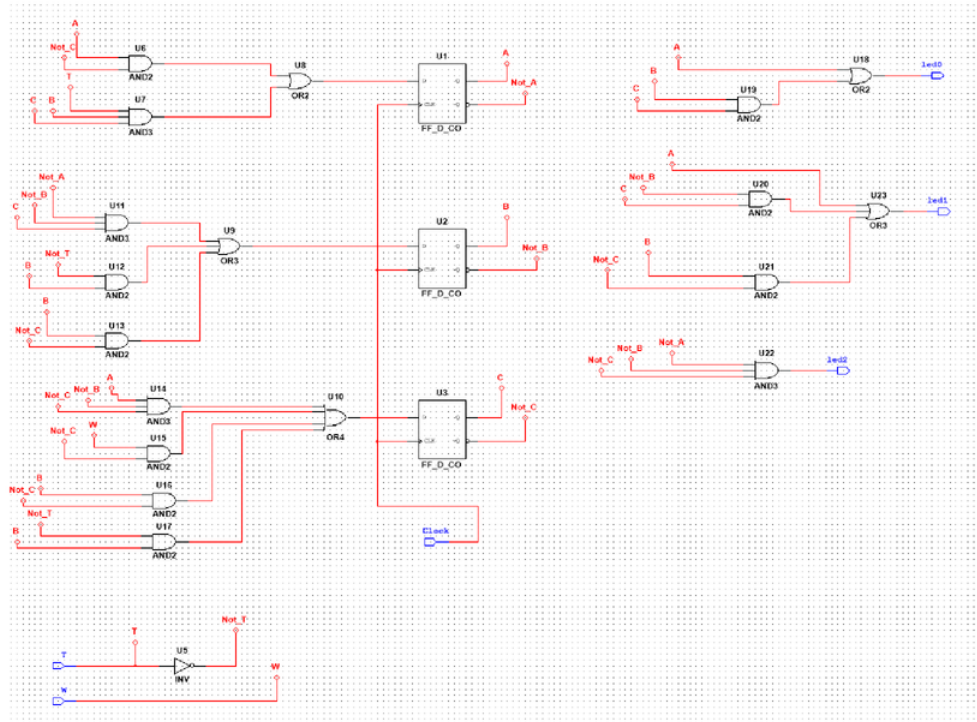


Figure 8: Controller circuit and outputs

All flip flops in every circuit are prompted by the clock pulse and so the clock connector is used to tie in all flip flops so that they all change state in unison. The red, amber and green outputs are connected to LEDs 0 to 2 on the Nexys board. Inverters are also present to flip the inputs for T as a 'NOT-T' logic input is required in this circuit.

As all three circuits have now been built, they can be individually compressed into smaller 'subcircuits' and connected together into a larger circuit which mimics the traffic light controller block diagram in figure 3.

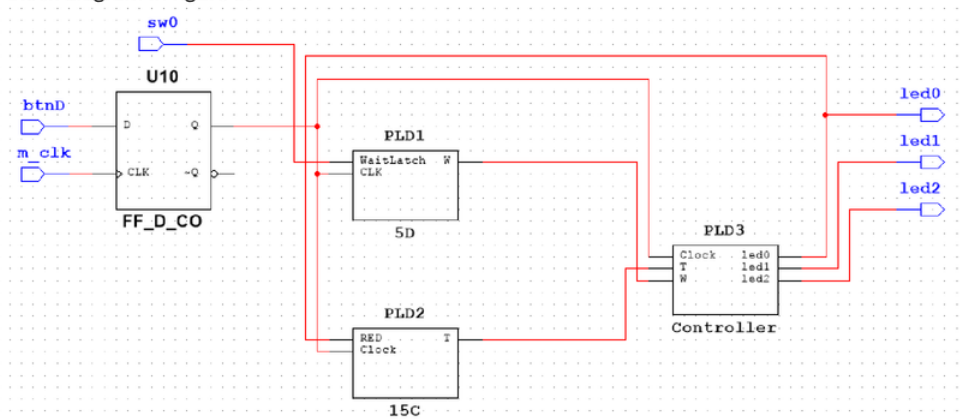


Figure 9: Complete traffic light controller

Figure 9 shows the entirety of the circuit combined with the sub circuits, the wait latch is operated by 'sw0' on the board and sends an input directly to 'PLD1', the 5 second delay circuit. The connector to 'btnD' is used to regulate the clock pulses to a user defined level. Operating this button will allow a pass of one clock pulse and allow the user to identify the state change apparent through observing the LEDs. This was done to observe every change when downloaded to the Nexys 4 DDR board at a manageable rate. The Red [led0] output is common to the input which prompts PLD2, the 15 second counter to start, before outputting 'T' to PLD3, the controller.

4. Testing and improving the design

In order to accurately test the design, each individual subcircuit must be tested along with the performance of the overall circuit. The Nexys 4 DDR board which had the program downloaded to and tested on can be seen in figure 10. This was used to test the finished program through use of the manual buttons and switches to control the clock pulse and observe the LED changes.

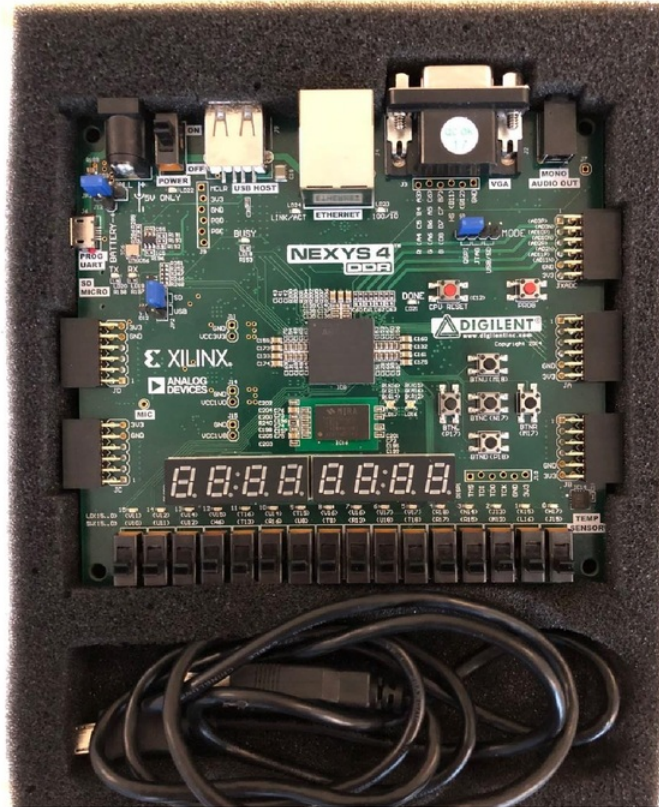


Figure 10: Nexys 4 DDR

The circuits also need to be tested within Multisim to observe the state changes. To do this, a logic analyser was connected to each individual subcircuit outputs and the master clock connector from the board was removed and replaced with a 1kHz signal. All other board connectors were removed and the program was run.

The first and simplest circuit to analyse was the five second counter, figure 11 shows the testing lay out for this particular circuit.

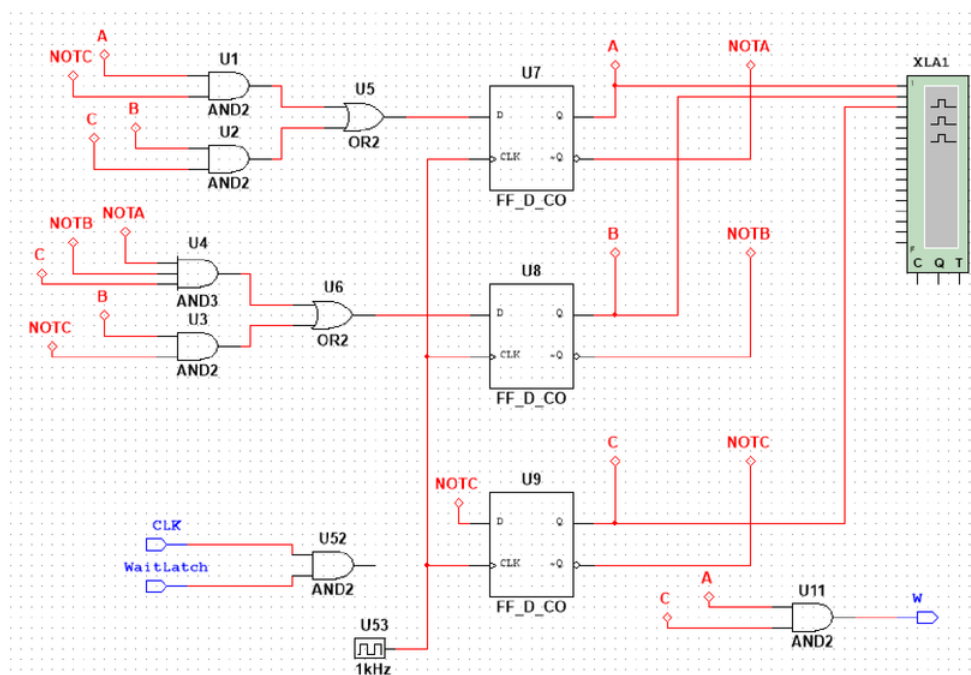


Figure 11: Testing of the 5 Second delay subcircuit

In figure 11, it can be seen that the 'WaitLatch' was removed and replaced with the 1kHz clock signal to allow the state machine to transition without the wait signal, for testing purposes.

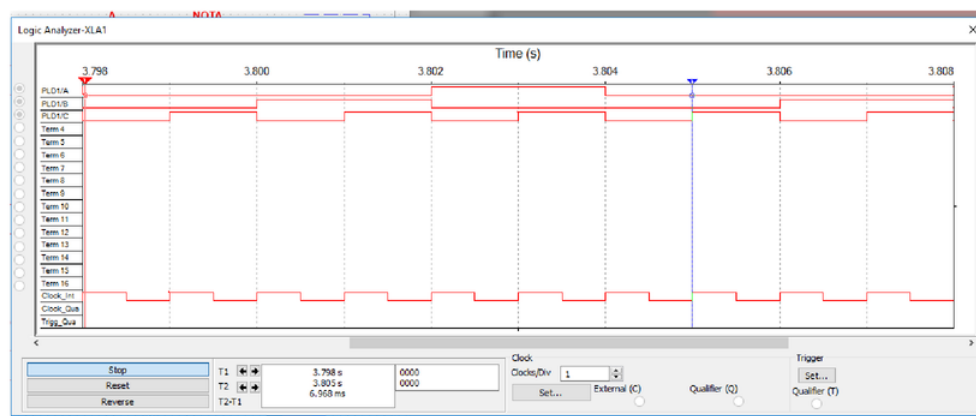


Figure 12: 5 second delay logic analyser results.

As shown in figure 12, the start of the count begins at the left of the display before reaching five and resetting back to zero where the blue marker is. The logic can be seen incrementing in binary as required, proving the operation of this circuit.

The next circuit to test was the fifteen second counter, the outputs were tied off into the logic analyser and the clock signal was replaced with the 1kHz signal as with the first circuit, shown in figure 13.

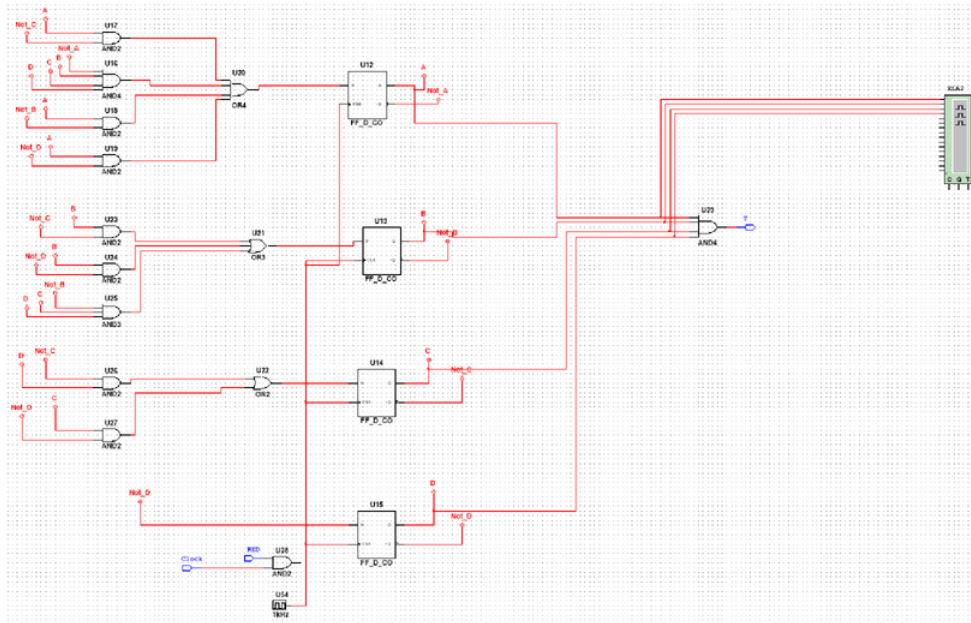


Figure 13:15 second counter testing

Figure 14 shows the results of the logic analyser.

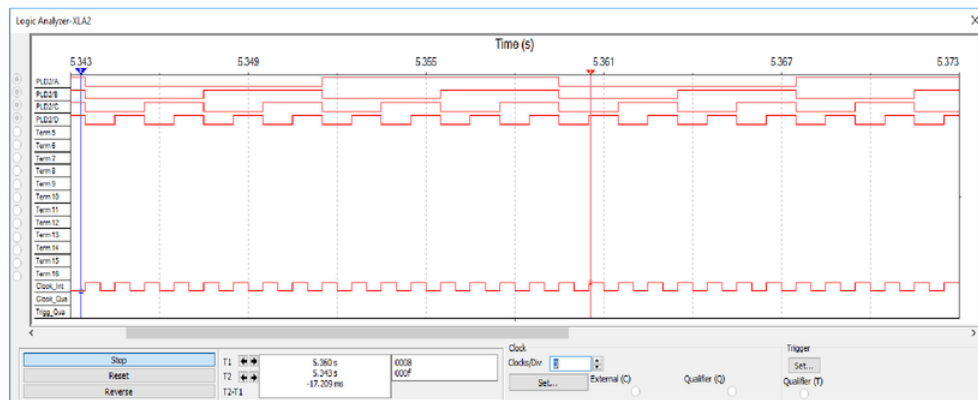


Figure 14: 15 second logic analyser results

The circuit can be seen counting from zero to 15 repeatedly in binary as required, proving the operation of this circuit.

The final circuit to test is the overall controller, as this has inputs from the other two circuits, the circuit must be run with forced inputs for T and W. As these inputs have no effect on the majority of the states, they can be continually forced without affecting the performance during the other states.

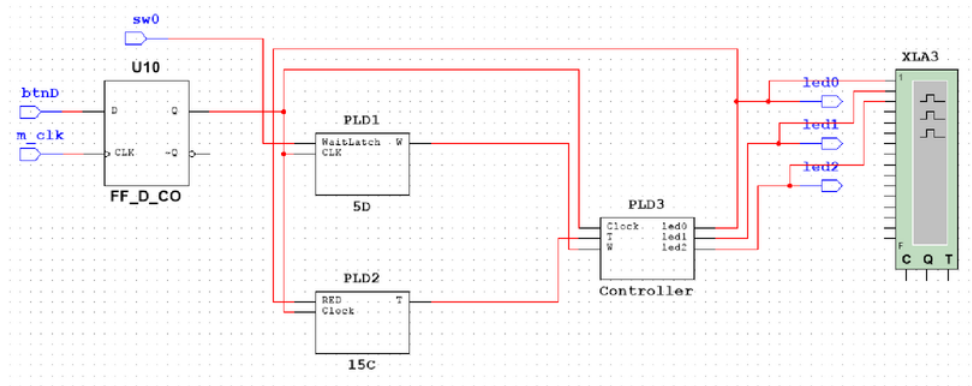


Figure 16: Controller testing.

Figure 15 shows how the controller was connected to the logic analyser for testing.

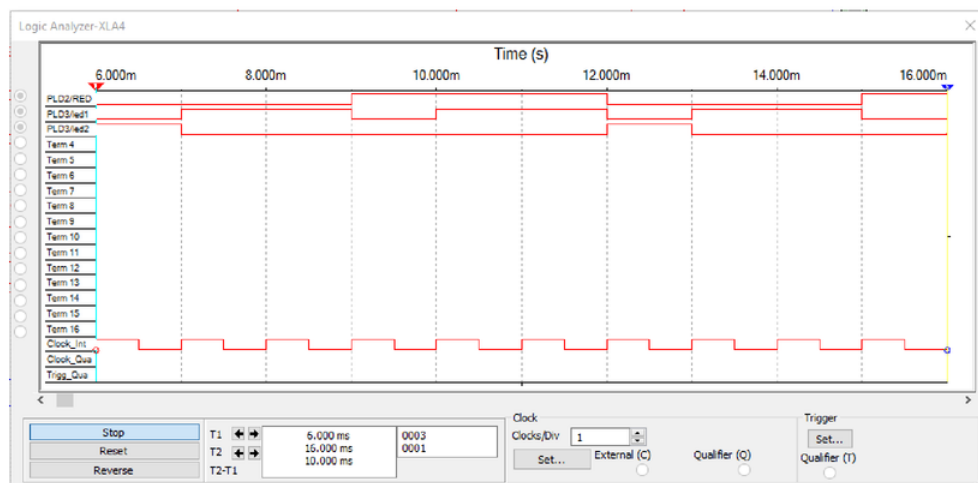


Figure 15: Controller logic analyser results.

Figure 16 shows the results of the controller testing.

When looking at these results, the state changes can be seen with each division. Referencing the state table, it can be seen that the controller begins outputting Green [led2], as the 'W' input will be continually forced, it prompts the state transition on the next clock pulse. Therefore the controller should instantly change to Amber [led1], it then remains with this same output during the next state also, which can be seen in the figure. The next state is then to output Red [led0] only, and as the T signal is forced, the controller transitions to the next state with the following clock pulse, which is

Red/Amber [led0 and led1]. Once again it remains with these outputs for two states before returning back to only outputting Green [led2]. This proves the controller as a whole.

To improve this design, it would involve redesigning the entire state machine for the controller, as additional inputs or states would shift the states along the transition table. This would mean a new set of Karnaugh maps would need to be produced to cut down on the amount of logic required as the current ones would no longer be applicable.

In order to improve and cut down on logic and states, the controller could be converted to a Mealy finite state machine, meaning the controller itself would react instantly to the inputs received from the counters, while also reducing the amount of logic needed. However, issues could arrive within transitioning between its' own states not dependant on the 'T' or 'W' signals, as another input would be required to switch the state machine at set intervals.

An improvement could be made to allow the system to also function as a junction traffic light with pedestrian crossing capability. Another two timers would need to be added, one for each switching of the two sets of lights, with a small offset between to allow traffic to finish crossing the junction before there is a green on the other side. The state machine would need to be modified to allow the 'W' signal to be effective at any time during the cycle, however the time until the system begins the transition towards a red light would vary depending on at which point during the cycle the wait latch was pressed.

5. Conclusions

Overall, this system is an effective traffic light controller for a pedestrian crossing. As this is a 'relatively' small system, the need to cut back on logic is not as much as what would be needed on a much larger system. Therefore it would be unnecessary to change the type of state machine used for this exercise.

The amount of states could be cut down by two if the length of time that the Amber and Red/Amber cycles could be increased. As this design requires two states to be used for states which have the same outputs. An additional timer could be added or existing timers would need to be modified and used repeatedly during the same cycle for different uses. To perform this improvement, the entire state machine would need to be redesigned, this is one disadvantage when using these in the design process, that a shift in the logic requires the designer to start from scratch. However, an advantage is that as they are 'finite' state machines, the system will not enter a state which is outside that of the state diagram, if the system has been designed correctly.

When investigating the other more advanced types of state machine, it becomes apparent that using higher level ones with memory capability would be required for the larger systems. Modifying the type of flip flop used to a JK flip flop would allow two inputs to be used to act against each other and change the output.

6. Reference list and bibliography

Works Cited

AMD, 1993. *State Machine Design*. [Online]

Available at: <http://noel.feld.cvut.cz/hw/amd/90005a.pdf>

[Accessed 20 March 2018].

James, M., 2018. *Finite State Machines*. [Online]

Available at: <http://www.i-programmer.info/babbages-bag/223-finite-state-machines.html>

[Accessed 20 March 2018].

Katz, R. H., 1993. *Contemporary Logic Design*. [Online]

Available at: <http://www-inst.eecs.berkeley.edu/~cs150/sp00/classnotes/katz-ch9-mod.pdf>

[Accessed 20 March 2018].

Kawano, T., Bouteau, F. & Mancuso, S., 2012. *US National Library of Medicine*. [Online]

Available at: https://openi.nlm.nih.gov/detailedresult.php?img=PMC3541313_cib-5-519-g1&req=4

[Accessed 02 March 2018].

Tutorials Point, n.d. *Pushdown Automata Introduction*. [Online]

Available at:

https://www.tutorialspoint.com/automata_theory/pushdown_automata_introduction.htm

[Accessed 20 March 2018].

University of California, 2005. *Sequential Implementation*. [Online]

Available at: <http://www-inst.eecs.berkeley.edu/~cs150/fa05/Lectures/07-SeqLogicIIx2.pdf>

[Accessed 20 March 2018].

7. Annexe

Figure 1: Moore and Mealy comparison.....	3
Figure 2: Pushdown Automata	4
Figure 3: Pelican crossing block diagram	6
Figure 4: State diagram and table.....	7
Figure 5: Five second delay counter	8
Figure 6: 15 second counter	10
Figure 7: Controller state diagram.....	11
Figure 8: Controller circuit and outputs.....	13
Figure 9: Complete traffic light controller	13
Figure 10: Nexys 4 DDR	15
Figure 11: Testing of the 5 Second delay subcircuit	16
Figure 12: 5 second delay logic analyser results.....	16
Figure 13: 15 second counter testing	17
Figure 14: 15 second logic analyser results	17
Figure 16: Controller logic analyser results.....	18
Figure 15: Controller testing	18

PSDE5006 Assignment 2

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

82/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

GRADING FORM: CSG-RUBIC

DANIEL INCE**82****RESEARCH-INFORMED LITERATURE**

There was evidence of extensive further reading and research from credible and reliable sources. Your interpretation and understanding of the additional research was very good and demonstrated a deep underlying appreciation of the subject. Your references were appropriate to the task and your reference list was of an appropriate length for this assignment.

8**KNOWLEDGE AND UNDERSTANDING**

The knowledge and understanding of the subject demonstrated by this assignment was generally excellent. There were very few gaps in your knowledge of the basic principles and the higher level concepts were very well explained. Your explanation of the design process and calculations was also excellent and it did extend your knowledge well beyond that contained in the course notes.

25**ANALYSIS**



Your analysis of the task was excellent, both in class and as shown in this assignment. You have demonstrated very good analytical skills and your approach to the task was very logical. You carried out a detailed evaluation of the results and included an excellent conclusion relating your practical results to accepted theory.

16

PRACTICAL APPLICATION AND DEP.



Your application of practical skills for this task was generally excellent. You used the resources available in MultiSim appropriately when designing the circuit and you used appropriate options for testing and evaluating the circuit performance. You managed to design a very efficient circuit that functioned well within the specification and achieved with very little assistance. Your results were very detailed, well presented and allowed for an excellent conclusion to be attempted.

25

SKILLS FOR PROFESSIONAL PRAC.



In this assignment you have demonstrated skills in working individually and also collaboratively. Your written report was very well structured and written in a coherent and logical way. The report was presented in a professional manner using appropriate language and formatting, good diagrams in appropriate places, good sentence/paragraph structure and clear calculations

8