

1809799

by Thomas Grennan

FILE	11301_THOMAS_GRENNAN_1809799_174902_766346302.DOCX (3.92M)		
TIME SUBMITTED	27-JAN-2020 06:43PM (UTC+0000)	WORD COUNT	5767
SUBMISSION ID	119197045	CHARACTER COUNT	28534

Thomas Grennan
Higher National Certificate: Electrical and
Electronic Engineering
SGEE5000 – Electronics II

Assignment 1: Finite State Machines

Submission Date: 28/01/2020

Word Count: 3854

1809799

Contents

Table of Figures	3
1. Executive Summary	5
2. Research into advanced digital design techniques	6
3. Design method, choices and calculations	13
Sequence Controller	14
5 Second Timer	21
15 Second Timer	23
4. Testing, results and analysis	29
4.1.0 Sequence Controller	29
4.1.1: 5 Second Timer	33
4.1.2: 15 Second Timer	36
4.2: Final Circuit Test	41
4.3: Modifying and Improving the Design	44
5. Conclusions and Recommendations	49
6. Reference list and Bibliography	51
7. Annexe for diagrams, data sheets and results tables	52

Table of Figures

Figure 1 – Combinational Logic Circuit	6
Figure 2 – Sequential Logic Circuit	6
Figure 3 – Sequential vs Combinational Logic Table	7
Figure 4 – Finite State Machine Diagram	8
Figure 5 – Mealy vs Moore	8
Figure 6 – Benefits of using a finite state machine	9
Figure 7 – Information on FPGAs and ASICs	10
Figure 8 – FPGA vs ASIC	10
Figure 9 – Comparison of the Merits of FPGAs and ASICs	11
Figure 10 – Traffic Light Circuit Overview	13
Figure 11 – RAG State Colours	14
Figure 12 – RAG State Diagram	14
Figure 13 – RAG Truth Table	15
Figure 14 – RAG Controller Karnaugh Map	16
Figure 15 – RAG Controller Circuit	17
Figure 16 – RAG Controller Outputs Truth table	18
Figure 17 – RAG Controller Outputs Karnaugh Map	18
Figure 18 – RAG Controller Output Circuit	19
Figure 19 – RAG Controller Full Circuit	20
Figure 20 – RAG Controller PLD	20
Figure 21 – 5 Second Timer State Diagram	21
Figure 22 – 5 Second Timer truth table	21
Figure 23 – 5 Second Timer Karnaugh Maps	22
Figure 24 – 5 Second Timer Circuit	23
Figure 25 – 5 Second Timer PLD	23
Figure 26 – 15 Second Timer State Diagram	24
Figure 27 – 15 Second Timer Truth Table	25
Figure 28 – 15 Second Timer Karnaugh Maps	26
Figure 29 – 15 Second Timer Circuit	27
Figure 30 – Traffic Light System Final Circuit	28
Figure 31 – Sequence Controller	29
Figure 32 – Sequence Controller Cycling Evidence	30
Figure 33 – Sequence Controller with Logic Analyzer	31
Figure 34 – Sequence Controller Logic Analyzer diagram	31
Figure 35 – RAG Controller PLD Circuit	32
Figure 36 – RAG Controller Test Table	32
Figure 37 – 5 Second Timer Circuit	33
Figure 38 – 5 Second Timer Lamp Test	33
Figure 39 – 5 Second Timer Logic Analyzer Circuit	34
Figure 40 – Logic Analyzer Timing Diagram	34
Figure 41 – 5 Second Timer PLD Circuit	35
Figure 42 – 5 Second Timer FPGA Test	35
Figure 43 – 15 Second Timer Circuit	36
Figure 44 – 15 Second Timer Lamp Tests	37
Figure 45 – 15 Second Timer Lamp Test (15s and output on)	38
Figure 46 – 15 Second Timer Logic Analyzer Circuit	38

Figure 47 – 15 Second Timer Logic Analyzer Timing Diagram	39
Figure 48 – 15 Second Timer PLD Circuit	39
Figure 49 – 15 Second Timer FPGA Test Table	40
Figure 50 – Final Traffic Light System	41
Figure 51 - Final Traffic Light System Lamp Test.....	41
Figure 52 - FPGA Board and Frequency Generator.....	42
Figure 53 - Final Traffic Light System PLD	42
Figure 54 - Traffic Light System Final Test	43
Figure 55 - Puffin Modification State Diagram	44
Figure 56 - Puffin Modification Truth Table.....	45
Figure 57 – Puffin Modification Karnaugh Maps	45
Figure 58 – Puffin Modification Circuit	46
Figure 59 – Puffin Modification Logic Analyzer	47
Figure 60 – Puffin Modification Lamp Test	47
Figure 61 – Puffin Modification PLD	48
Figure 62 – Final PLD Circuit with Puffin Modification	48
Figure 63 – Puffin Modification Test Table	48
Figure 64 - Benefits of using an ASIC	49

1. Executive Summary

The report was focused on the research, design, construction and testing of three circuits which will be connected together as a single circuit; a 5 Second Timer, 15 Second Timer and RAG controller. The circuit will operate by a pedestrian pressing a button to cross the road, this starts a 5 second timer, after 5 seconds the traffic lights will start the sequence to turn red. When the light turns red an output from the red light starts a timer of 15seconds to allow the pedestrian to cross, after 15 seconds a signal is sent to reset the traffic lights back to green for the traffic to move.

Firstly, the research was completed, this was centred on finding out information about Combinational and Sequential logic circuits, FPGAs and ASICs, Finite State Machines and the use of modern components to promote sustainable development.

The RAG controller was designed first, then the 5 second and finally the 15 second timer. Each stage had a state diagram, truth table, Karnaugh map and a circuit built from the expression derived from the map.

Each circuit was tested, each circuits sequence was tested by checking that the right lights were illuminating in the correct order to display the correct binary code, then a logic analyser was used to find out that the sequence was correct by looking at the timing diagrams. Each circuit was then modified into PLD mode and tested on the FPGA board to ensure correct operation.

After the testing all three circuits were connected together and tested through the same means.

A modification was then designed and implemented; this turned the pelican crossing into a puffin crossing. This was implemented by modifying the 5 second timer to allow an extra input in the form of a pressure pad whereby the pedestrian had to press the button and wait on the pad for 5 seconds to start the sequence. The modification was successfully designed and tested.

A conclusion and recommendation were then the final task. The valuable information about the task was discussed and what could then be improved. A recommendation was made by adding an audible beeper to the red light to allow visually impaired people to be more aware when crossing the road.

2. Research into advanced digital design techniques

A combinational logic circuit in digital electronics is a type of digital logic circuit, in this circuit its output is a function of its present inputs. Combinational circuits do not contain any feedback loops or memory, combinational circuits are also instantaneous. Combinational circuits have the ability to change their state dependent on any inputs that they receive and these circuits are designed to produce a specific output from a variety of inputs. (elprocus, 2016)

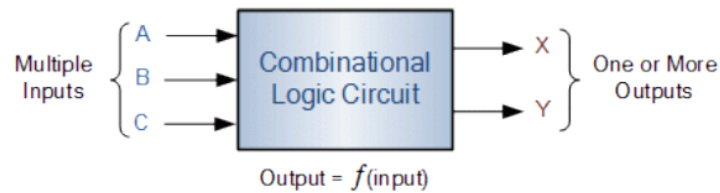


Figure 1 – Combinational Logic Circuit

Unlike combinational circuits which change states dependant on input signals during a certain time, sequential logic circuits contain a 'memory' built in. This memory enables sequential logic circuits take into account previous states of the circuit's inputs. The output depends on the combination of the logic input states along with the previous existing logic states. (ElectronicsTutorials, 2017)

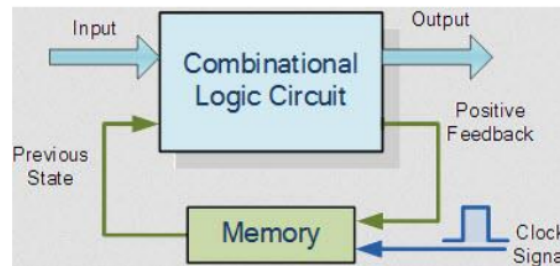


Figure 2 – Sequential Logic Circuit

There are many differences between combinational and sequential logic circuits which would influence the choice of the circuit type during the design stage. These differences are addressed in figure 3, the table was constructed with help from STechies (STechies, 2019)

Sequential vs Combinational Logic Circuits	
Sequential Logic Circuit	Combinational Logic Circuit
In sequential logic output depends upon present as well as past input.	In Combinational logic the output depends only upon present input.
Speed is slow.	Speed is fast.
Its design is more complex as compared to combinational circuits.	Easier design
Feedback path exists between inputs and outputs. These circuits need feedback from the output to save inputs in the memory space for the next set of operations.	No feedback between inputs and outputs. Combinational circuits generally output the input as per the logic designed.
Time dependent.	Time independent.
Elementary building blocks: Flip-flops	Elementary building blocks: Logic gates
Widely used for storing data.	Used for arithmetic as well as Boolean operations
Sequential circuits have the memory space needed for storing the current states sent as control inputs (enable) for the next set of operations. They contain memory elements that store data in digital circuits.	Combinational circuits don't have capability to store any state. They do not contain any memory element.
Sequential logic circuits are clocked, and so they are triggered for operations with the help of electronic pulses. The flip-flops, if triggered, are termed as synchronous sequential circuits. The circuits that are not triggered are termed as asynchronous sequential circuits.	Combinational circuits are independent of a clock; therefore, triggering is not essential for their operations.
These circuits have memory element.	These circuits do not have any memory element.
Can be difficult to use and handle	It is easy to use and handle.
Sequential circuits are used in the case of flip-flops and latches. In these circuits, latches are considered to be the simplest elements used for retaining the earlier state. The latches are also known as flip-flops.	These circuits are used in association with encoders, adders, decoders, subtractors, multiplexers, etc.

Figure 3 – Sequential vs Combinational Logic Table

These differences are very important when deciding the circuits application. For example, combinational circuits are widely used for encoders and decoders. Whereas, Sequential circuits are used primarily for latches and flip-flops.

A finite state machine (FSM) is a computation model, this model is executed with software and hardware to simulate sequential logic. These machines are often used for the purpose of problem solving in mathematics and engineering.

The purpose of an FSM is to store a sequence of different states which are all unique, the machine then transitions between different states depending on the input's values and the current machine state. Each machine has a fixed set of states that the machine will operate in and the machine is only able to operate in one state at a time and every state has a set of transitions which are linked to a number of inputs. (Wilson, 2015)

Figure 4 shows an FSM, in the diagram the circles are the states that the machine can be in, in this case it is either locked or unlocked, the arrows between the states are the transitions, such as the door being open or closed.

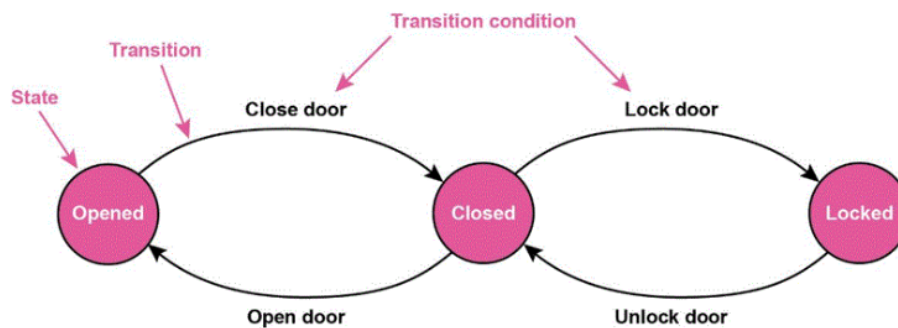


Figure 4 – Finite State Machine Diagram

There are two types of FSM: Moore and Mealy, a Mealy FSM is where the output is dependent on the current input value and the current state variable values. Whereas, a Moore FSM is where the output of the machine is entirely dependent on the state variable. A table comparing the characteristics is shown in figure 5. (tutorialspoint, 2018)

Mealy Machine	Moore Machine
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.
Mealy machines react faster to inputs. They generally react in the same clock cycle.	In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.

Figure 5 – Mealy vs Moore

FSMs are often utilised as a design tool especially for software projects. The use of FSMs as a design tool has become very common over the years in different software applications, this is also popular when the application uses the Unified Modelling Language. The benefits of using an FSM as a design tool are shown in figure 6.

- It's applicable to a large variety of software projects.
- It can be valuable at many stages of development, from top-level design to module coding.
- It provides a fairly rigorous strategy that yields a simple and consistent code structure.
- It creates a useful documentation and maintenance product that reflects the organization of the code.
- It can incorporate an effective debugging tool with minimal effort.
- It lends itself to design-by-committee and peer reviews.

Figure 6 – Benefits of using a finite state machine

The idea behind FSM design is that a great amount of software problems shares the perspective of state machines, due to this the problems that arise can often be solved using methods borrowed from other disciplines such digital electronics.

Software programmes that are best suited to FSMs model are categorized as being very control intensive with “distinct modes”, this means that they have quite a complicated logical structure, therefore it contains an extensive amount of control blocks such as conditional statements and case structures. (Staff, 2009)

FSMs are also used by ASIC Engineers as they have the ability to be coded directly into “Verilog” and they can be designed in a graphical user interface which would then generate a code.

Field Programmable Gate Arrays (FPGAs) are semiconductor devices based around matrixes of configurable logic blocks. Unlike Application Specific Integrated Circuits (ASICs) FPGAs have the ability to be reprogrammed for a specific desired application, they can also be edited to add any functionality requirements after production. (Xilinx, 2018)

FPGAs are often utilised when designing ASICs and processors. FPGAs can be continuously reprogrammed until the circuit is in its final stages and is free of software bugs, the ASIC design can then start, multinational processor companies such as Intel utilise this process (Figure 7). (Touger, 2018)

"Engineers can create a design using an FPGA, then rather than spending time optimizing the circuit layout, they bake the fixed layout into a single design mask for manufacturing. By being a fixed design like an ASIC, it is faster than a variable design, but without the die area benefits of ASIC-like power savings. However, it was designed in FPGA time, rather than ASIC time (up to six months saved), and saves power through its fixed design."

Figure 7 – Information on FPGAs and ASICs

ASICs are integrated circuits that are not standard, they have been designed and manufactured a specific purpose and application. ASICs are often used for products that have a big production run. Due to the ASIC containing a large section of electronics on a single integrated circuit the prototyping and design stage is expensive, due to this costing ASICs are often only used for mass production of high volume products. (electronics-notes, 2018)

Even though ASICs are initially expensive they can actually be cost effective in the long run for high volume applications and products. ASICs give the opportunity for designers to meet an exact specification and by doing this a full design can be contained in one integrated circuit, by doing this it reduces the number of external components.

ASICs and FPGAs are visually compared in Figure 8. (Singh, 2018)

	FPGA	ASIC
NRE	✓	
Performance		✓
Time to market	✓	
Design Flow	✓	
Cost per Unit (high volume)		✓
Barrier to Entry	✓	
Energy Efficiency		✓
Analog Blocks		✓

Figure 8 – FPGA vs ASIC

A table was created comparing the merits of both the FPGA and ASIC, this was done with the help of Rohit Singh from Numato.com (Figure 9) (Singh, 2018)

Comparison of the Merits of FPGAs vs ASICs	
FPGA	ASIC
Reconfigurable circuit. FPGAs can be reconfigured with a different design. They even have capability to reconfigure a part of chip while remaining areas of chip are still working! This feature is widely used in accelerated computing in data centres.	Permanent circuitry. Once the application specific circuit is taped-out into silicon, it cannot be changed. The circuit will work same for its complete operating life.
Unsuitable for very high-volume mass production.	Suitable for very high-volume mass production.
Design is specified generally using hardware description languages (HDL) such as VHDL or Verilog.	Same as for FPGA. Design is specified using HDL such as Verilog, VHDL etc.
Less energy efficient, requires more power for same function which ASIC can achieve at lower power.	Much more power efficient than FPGAs. Power consumption of ASICs can be very minutely controlled and optimized.
Suitable for applications such as Radars, Cell Phone Base Stations. This beneficial as the current design could be upgraded often to use a more efficient algorithm or to a better design. In these applications, the high-cost of FPGAs is not the deciding factor. Instead, programmability is the deciding factor.	ASICs are unsuitable for application areas where the design might need to be upgraded frequently or once-in-a-while.
Preferred for prototyping and validating a design or concept. Many ASICs are prototyped using FPGAs themselves! Major processor manufacturers themselves use FPGAs to validate their System-on-Chips (SoCs). It is easier to make sure design is working correctly as intended using FPGA prototyping.	It is not recommended to prototype a design using ASICs unless it has been absolutely validated. Once the silicon has been taped out, almost nothing can be done to fix a design bug

Figure 9 – Comparison of the Merits of FPGAs and ASICs

With the merits found in the table above it makes the choice between FPGAs and ASICs easier when designing circuits. Both FPGAs and ASICs have benefits and disadvantages, they both have extensive merits which are vital to be taken into account before designing a circuit.

The use of modern components and designs in electronics engineering is vital for sustainable development. It is essential when designing new products that it complies with ecological sustainability.

Modern designs such as ASICs have low power consumption which can be optimized for certain applications to improve power consumption again, using integrated circuits like ASICs also reduces the needs for external components, with less components there is less room for breakdowns improving the products sustainability.

Once an ASIC is fully developed, it helps the systems that it is running to become more sustainable, if an ASIC goes into fault it can be replaced easily as most of its circuitry is contained in the chip.

The use of FPGAs is good for sustainable development for electronics, due to their ability to reconfigure and change current circuits by modifying and trial and error they help with the waste reduction that is usually associated with trials and prototypes, When the circuit is completed to specification it can then be modified onto an ASIC without the problem of software bugs, this reduces the cost of wasted components and reduces the amount of components sent to landfill where 80% of electronic waste ends up. By doing this it decreases the carbon footprint, utilises reuse of components and reduces over consumption which are effective solutions for bettering the environment and society.

3. Design method, choices and calculations

The task for this assignment is to design a traffic light control system using the principles of finite state machines.

This circuit will be compiled of pedestrian lights and traffic lights. When the button (B) is pressed, A 5 second timer will start, this allows cars to have some time to pass through, after 5 seconds the output (W) will go high and start the RAG controller. The traffic lights will change from green to amber then red , at red a 15 second timer starts, this allows the pedestrian to cross the road, after 15 seconds a trigger signal is sent to the controller to reset the lights.

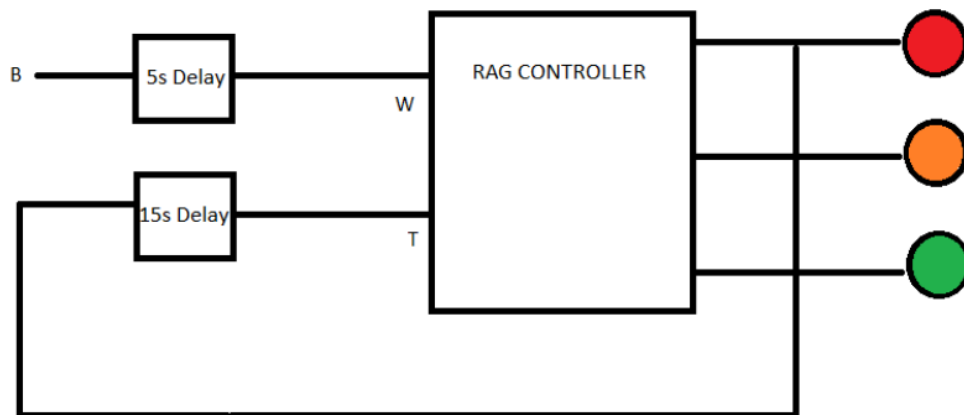


Figure 10 – Traffic Light Circuit Overview

Sequence Controller

The first stage of the design was to create a sequence controller, this controls the order of the traffic lights. During this design state diagrams, truth tables, Karnaugh maps and Boolean expressions were derived to create a circuit to control these lights.

Firstly, the order of the states and colours were decided, these states have individual colours which illuminate during that state (Figure 11)

State 0 – Green

State 1 – Amber

State 2 – Amber

State 3 – Red

State 4 – Red/Amber

State 5 – Red/Amber

Figure 11 – RAG State Colours

Figure 12 shows the state diagram for the sequence controller, this is a good visual display of how the circuit will change from state to state. This shows what state the traffic light colours are in at each step, it also displays the timers and input triggers. This diagram also helps with the truth table and Karnaugh maps.

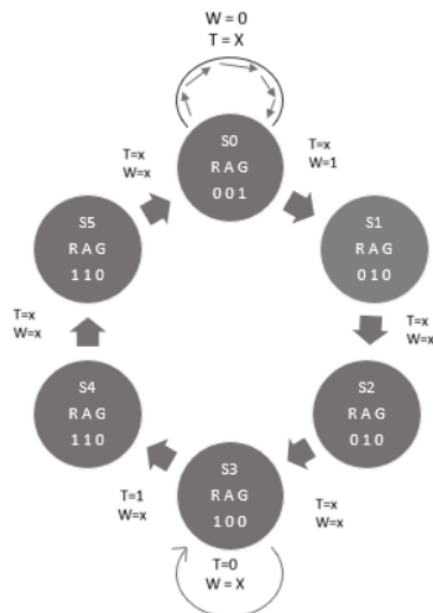


Figure 12 – RAG State Diagram

Figure 13 is the truth table for the RAG controller, the truth table displays binary numbers of 1 and 0, this also displays what state the inputs have to be in order to change the output state. This state table is needed in order to fill in the Karnaugh map, the Boolean expressions will then be derived from this Karnaugh map.

		NEXT				Current			W=0						W=1						OUTPUTS		
		W=0		W=1					T=0		T=1		T=0			T=1							
State	T=0	T=1	T=0	T=1	Qa	Qb	Qc	Qa+	Qb+	Qc+	Qa+	Qb+	Qc+	Qa+	Qb+	Qc+	Qa+	Qb+	Qc+	R	A	G	
S0	S0	S0	S1	S1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
S1	S2	S2	S2	S2	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	
S2	S3	S3	S3	S3	0	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0	1	0	
S3	S3	S4	S3	S4	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	0	
S4	S5	S5	S5	S5	1	0	0	0	1	0	1	1	0	1	1	0	1	1	1	1	1	0	
S5	S0	S0	S0	S0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
S6	X	X	X	X	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S7	X	X	X	X	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

Figure 13 – RAG Truth Table

Figure 14 shows the Karnaugh maps that have been derived from the truth table. A Karnaugh map is one of the best methods of simplifying Boolean algebra, each cell of the map depicts a combination of the input signals. When grouping a Karnaugh map it is important to have the least number of groups, and the larger the group the better. However, group sizes can only be of the size "2ⁿ", therefore they can be in groups of 1, 2, 4, 8 and so on. Every "1" in the map must be grouped, however, "Xs" can be used as 1s or 0s, so they can make up groups if need be, this makes it very useful when trying to make as larger group as possible.

When inputting data into the Karnaugh map It is important to swap the third and fourth section, this is due to the map only being able to switch one state at a time. For each map there are two maps for the states of the two inputs T and W. (T=0 or T= 1 and W=0 or W=1)

QA+									
W=0					W=1				
T Qa/Qb Qc	00	01	11	10	T Qa/Qb Qc	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
$QA+ = A'C + TBC$									
QB+									
W=0					W=1				
T Qa/Qb Qc	00	01	11	10	T Qa/Qb Qc	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
$QB+ = B'C + TB + A'BC$									
QC+									
W=0					W=1				
T Qa/Qb Qc	00	01	11	10	T Qa/Qb Qc	00	01	11	10
00	0	0	1	1	00	1	0	1	1
01	1	0	x	x	01	1	0	x	x
11	1	0	x	x	11	1	0	x	x
10	0	0	0	1	10	1	0	0	1
$QC+ = A'C + TB + B'C + W'C$									

Figure 14 – RAG Controller Karnaugh Map

After the grouping, as shown below each map (Figure 14), the groups are simplified into a Boolean expression, the expression is derived by looking at the borders “Co-Ordinates” of the map, the algebra is extracted from the co-ordinates that don’t change states. Only groups of 1 are counted for the expression, the bigger the group of 1 the smaller the expression.

After deriving the algebraic expression, it is then turned into a circuit (Figure 15).

The expression for QA+ was: "A.NOTC + TBC" for this QA+ is the output, the "+" means "or", therefore, either "A.NOTC" or "TBC" can turn the output on if they are high. The outputs are connected to flip flops and a 1Khz clock which will change between "A" and "NOT A" at every clock pulse, this is to change the output state to match the input state. The same steps were copied for the other two expressions. "T" was also put through an inverter gate to get "NOT T".

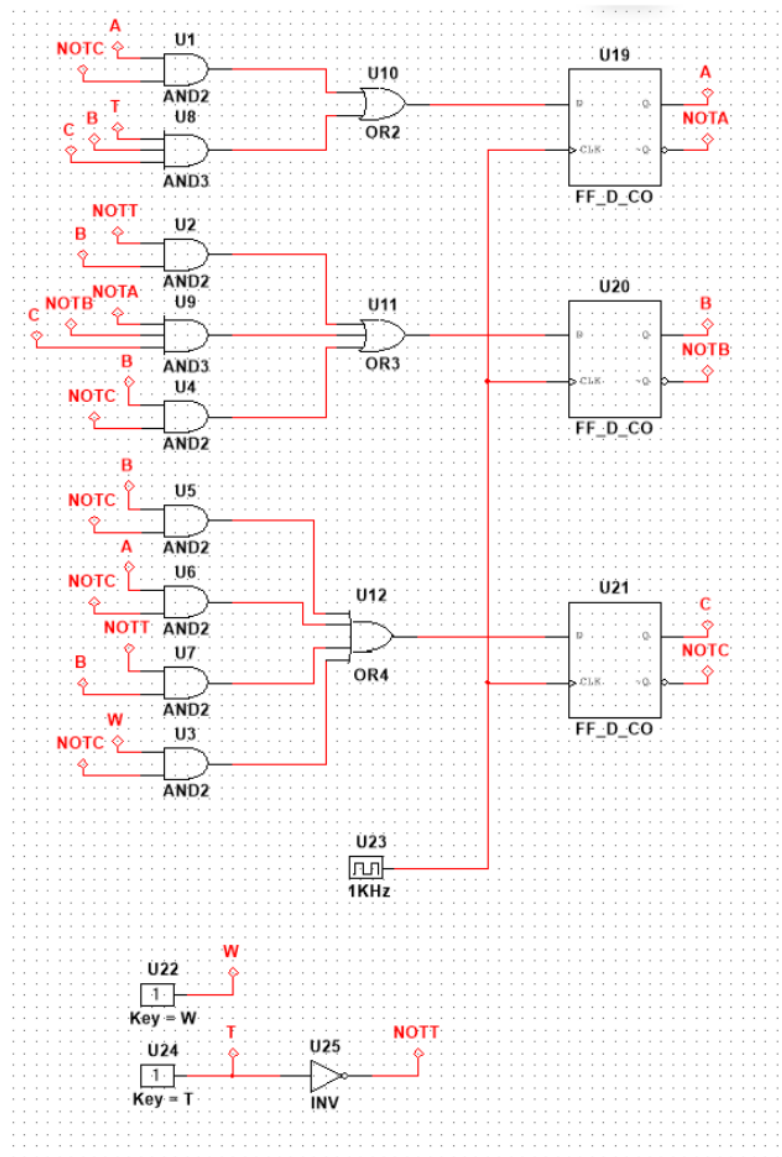


Figure 15 – RAG Controller Circuit

The truth table was then created for the RAG outputs, this connects the controller circuit to the outputs. (Figure 16)

Sequence Controller						
State	Qa	Qb	Qc	R	A	G
S0	0	0	0	0	0	1
S1	0	0	1	0	1	0
S2	0	1	0	0	1	0
S3	0	1	1	1	0	0
S4	1	0	0	1	1	0
S5	1	0	1	1	1	0
XX	XX	XX	XX	XX	XX	XX
XX	XX	XX	XX	XX	XX	XX

Figure 16 – RAG Controller Outputs Truth table

Figures 17 show the Karnaugh maps which were derived from the truth table. These maps were grouped in the biggest possible group and then the expression was derived as shown below each map.

Red				
Qa/Qb Qc	00	01	11	10
0	0	0	1	0
1	1	1	x	x
Red:	$QA + QbQc$			

Amber				
Qa/Qb Qc	00	01	11	10
0	0	1	0	1
1	1	1	x	x
Amber:	$QA + 'QbQc + Qb'Qc$			

Green				
Qa/Qb Qc	00	01	11	10
0	1	0	0	0
1	0	0	x	x
Green:	$QA'Qb'Qc$			

Figure 17 – RAG Controller Outputs Karnaugh Map

Figure 18 is the circuit that was created from the expression, this circuit will then be connected with the controller to create the first sub circuit.

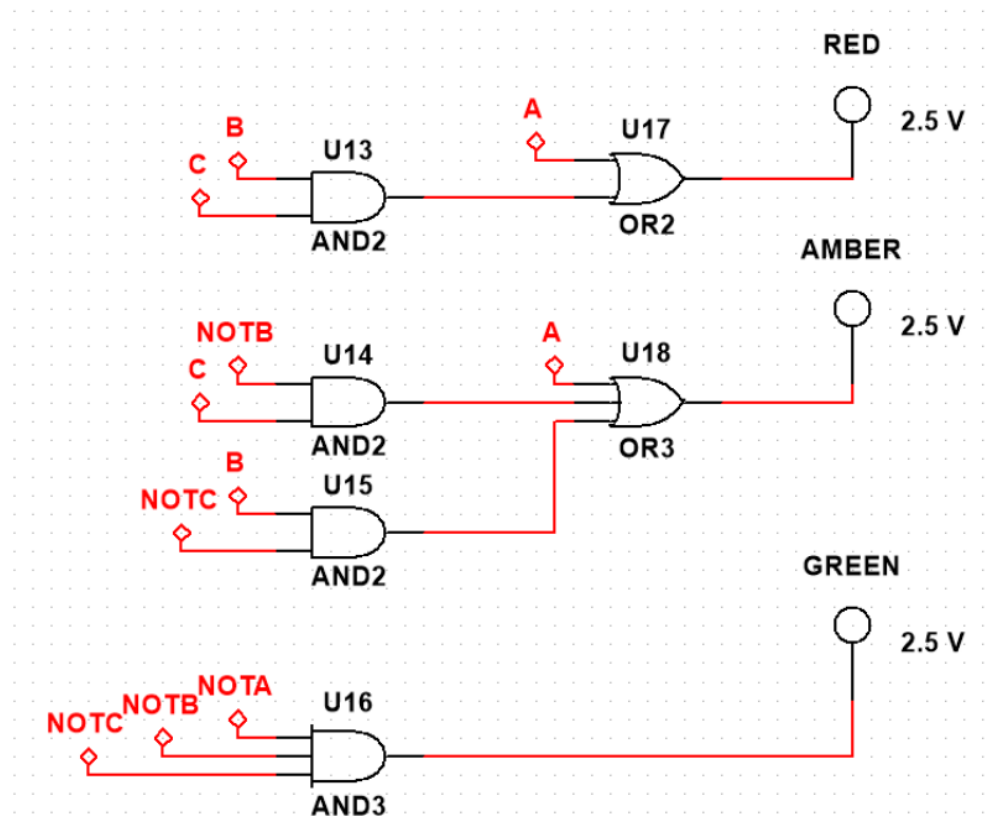


Figure 18 – RAG Controller Output Circuit

Figure 19 shows the two circuits together.

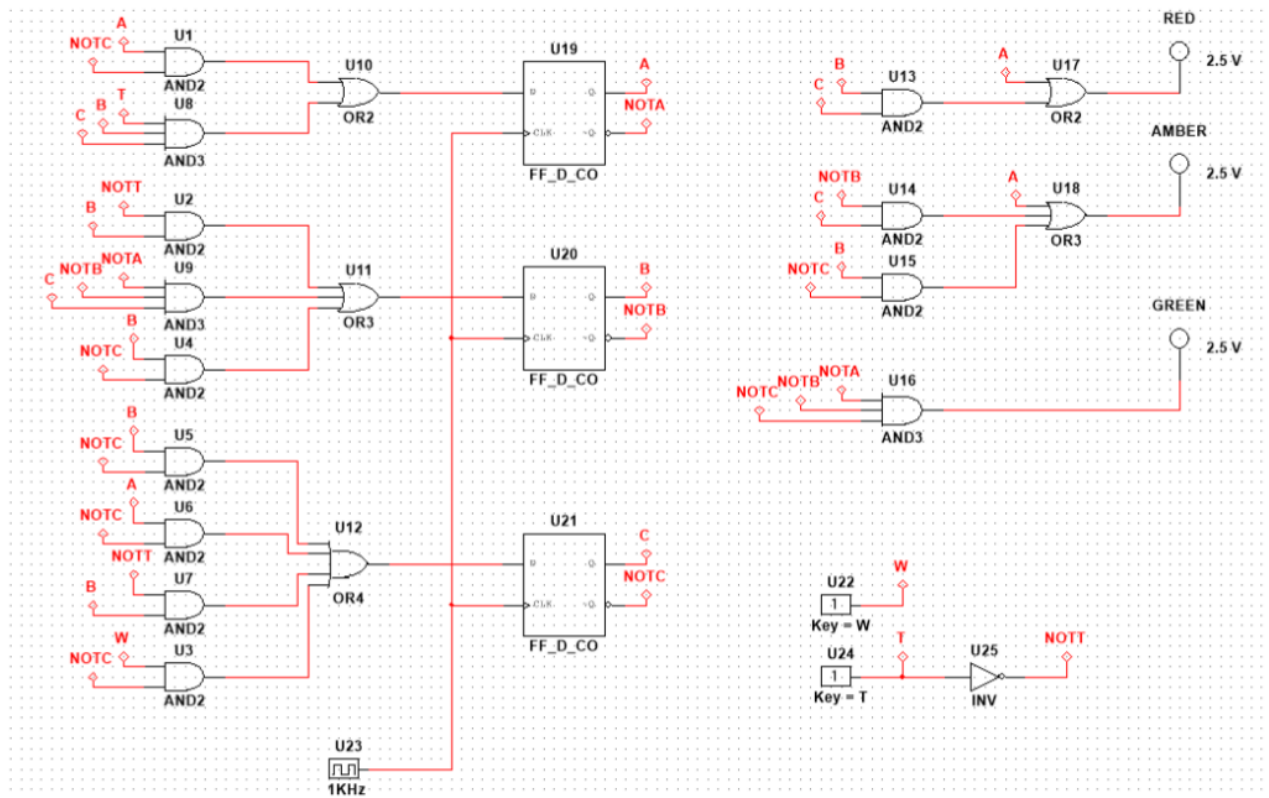


Figure 19 – RAG Controller Full Circuit

The circuit was then turned into a PLD sub circuit for the FPGA board, As shown in figure 20 the sub circuit has existing connectors to allow all of the sub circuits to be seamlessly connected together, this includes the trigger output from the 15second timer and the wait signal from the 5 second timer. The final sub-circuits will be connected together and will run on the same clock frequency so it can work as one main circuit.

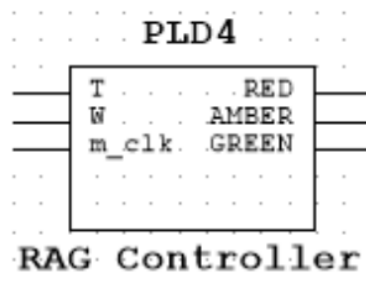


Figure 20 – RAG Controller PLD

5 Second Timer

The second part of the circuit to be designed is the 5 second timer. A state diagram was created for this circuit (Figure 21). This circuit is designed to count from S0 to S5, when the current step is S5 and the binary code is 101 the output will be high, it will then return to S0. This circuit will start a 5 second timer when the button is pressed, after the 5 seconds elapses a wait signal will be sent to the RAG controller.

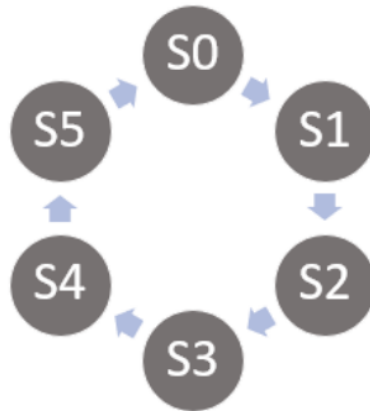


Figure 21 – 5 Second Timer State Diagram

The next stage was to create a truth table from the state diagram, the table was made for when the button is in either the high or low state. (Figures 22)

	Next		Present			B = 0			B = 1			Output (W)
	B = 0	B = 1	QA	QB	QC	QA+	QB+	QC+	QA+	QB+	QC+	
S0	S0	S1	0	0	0	0	0	0	0	0	1	0
S1	S2	S2	0	0	1	0	0	1	0	1	0	0
S2	S3	S3	0	1	0	0	1	0	0	1	1	0
S3	S4	S4	0	1	1	0	1	1	1	0	0	0
S4	S5	S5	1	0	0	1	0	0	1	0	1	0
S5	S0	S0	1	0	1	1	0	1	0	0	0	1
S6	XX	XX	1	1	0	X	X	X	X	X	X	X
S7	XX	XX	1	1	1	X	X	X	X	X	X	X

Figure 22 – 5 Second Timer truth table

From this truth table Karnaugh maps will be made for QA+, QB+, QC+ and the wait output signal, the Boolean expressions have been derived under each map(Figure 23)

QA+				
B QA /QB QC	00	01	11	10
00	0	0	1	0
01	1	0	X	X
11	1	0	X	X
10	0	0	1	0
$QA+ = QBQC + QA'QC$				

QB+				
B QA /QB QC	00	01	11	10
00	0	1	0	1
01	0	0	X	X
11	0	0	X	X
10	0	1	0	1
$QB+ = QB'QC + QA'QBQC$				

QC+				
B QA /QB QC	00	01	11	10
00	0	0	0	1
01	1	0	X	X
11	1	0	X	X
10	1	0	0	1
$QC+ = QB'QC + B'QC + QA'QC$				

W				
QA / QB QC	00	01	11	10
0	0	0	0	0
1	0	1	X	X
$W = QAQC$				

Figure 23 – 5 Second Timer Karnaugh Maps

After deriving the Boolean expressions, the circuit was then created on multisim. QA+, QB+ AND QC+ are connected to flip-flops, these flip flops are clock driven to a specific frequency, for every clock pulse the state of the output changes to match the input signal. (Figure 24)

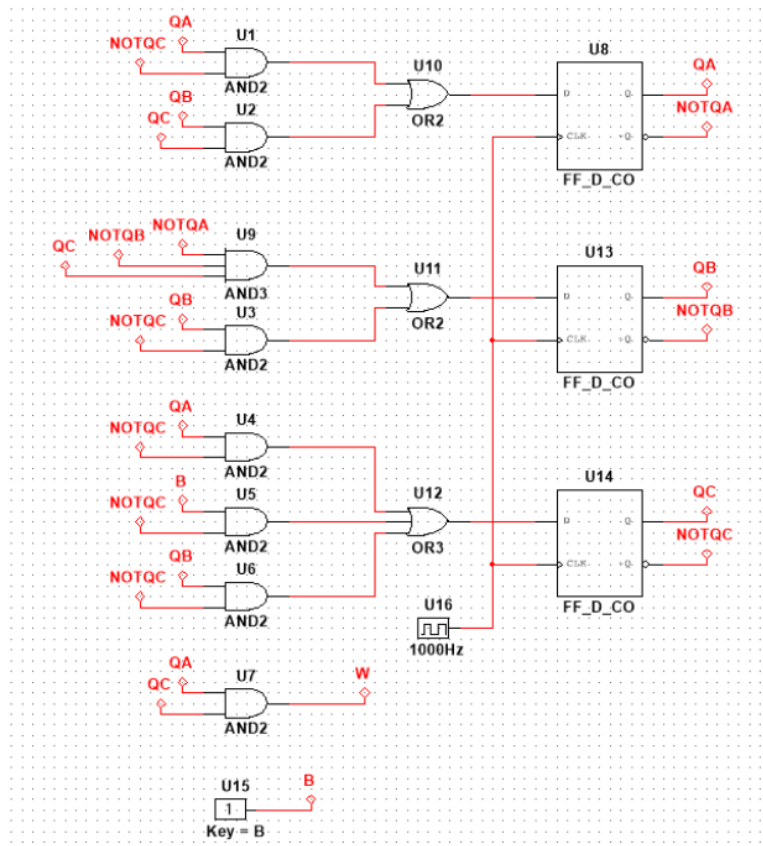


Figure 24 – 5 Second Timer Circuit

This circuit was then turned into a PLD sub-circuit, this sub-circuit will be connected to the 15 second timer and the RAG controller to complete the full circuit. This sub circuit has connectors for a clock signal, an input from the pedestrian button and a wait output after the 5 seconds elapses. (Figure 25)

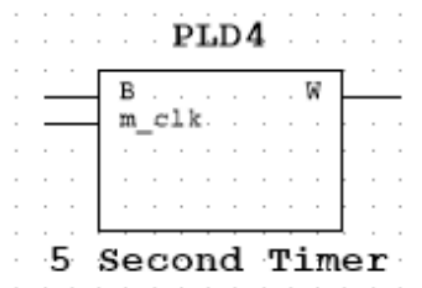


Figure 25 – 5 Second Timer PLD

15 Second Timer

The final circuit to be created for the main circuit is the 15 second timer, this timer counts from S0 to S15, at S15 the output will go high, the circuit will then reset back to S0. (Figure 26)

The circuit will work by taking a signal from the RAG controllers red light, when the red light is high it sends an input to the 15 second timer to start the timer, when the 15 seconds elapses it sends a trigger signal to the RAG controller to reset.

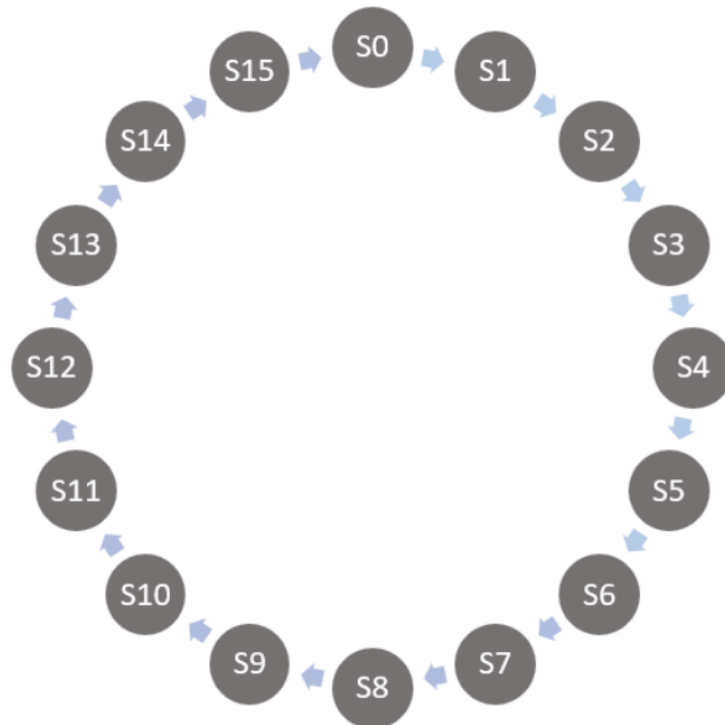


Figure 26– 15 Second Timer State Diagram

Using the state diagram, the truth table for the 15 second timer was created, this truth table shows that at S15 when the binary code is “1111” the output is high and then the circuit resets back to S0. From this truth table the Karnaugh maps for QA+,QB+,QC+ and QD+ can be created. (Figure 27)

Current State	Next State	Current				Next				Output
		QA	QB	QC	QD	QA+	QB+	QC+	QD+	
S0	S1	0	0	0	0	0	0	0	1	0
S1	S2	0	0	0	1	0	0	1	0	0
S2	S3	0	0	1	0	0	0	1	1	0
S3	S4	0	0	1	1	0	1	0	0	0
S4	S5	0	1	0	0	0	1	0	1	0
S5	S6	0	1	0	1	0	1	1	0	0
S6	S7	0	1	1	0	0	1	1	1	0
S7	S8	0	1	1	1	1	0	0	0	0
S8	S9	1	0	0	0	1	0	0	1	0
S9	S10	1	0	0	1	1	0	1	0	0
S10	S11	1	0	1	0	1	0	1	1	0
S11	S12	1	0	1	1	1	1	0	0	0
S12	S13	1	1	0	0	1	1	0	1	0
S13	S14	1	1	0	1	1	1	1	0	0
S14	S15	1	1	1	0	1	1	1	1	0
S15	S0	1	1	1	1	0	0	0	0	1

Figure 27 – 15 Second Timer Truth Table

Figures 28 show the Karnaugh maps which have been derived from the truth table, the biggest groups of 1s have been grouped in order to achieve the simplest Boolean expression as shown under the figures.

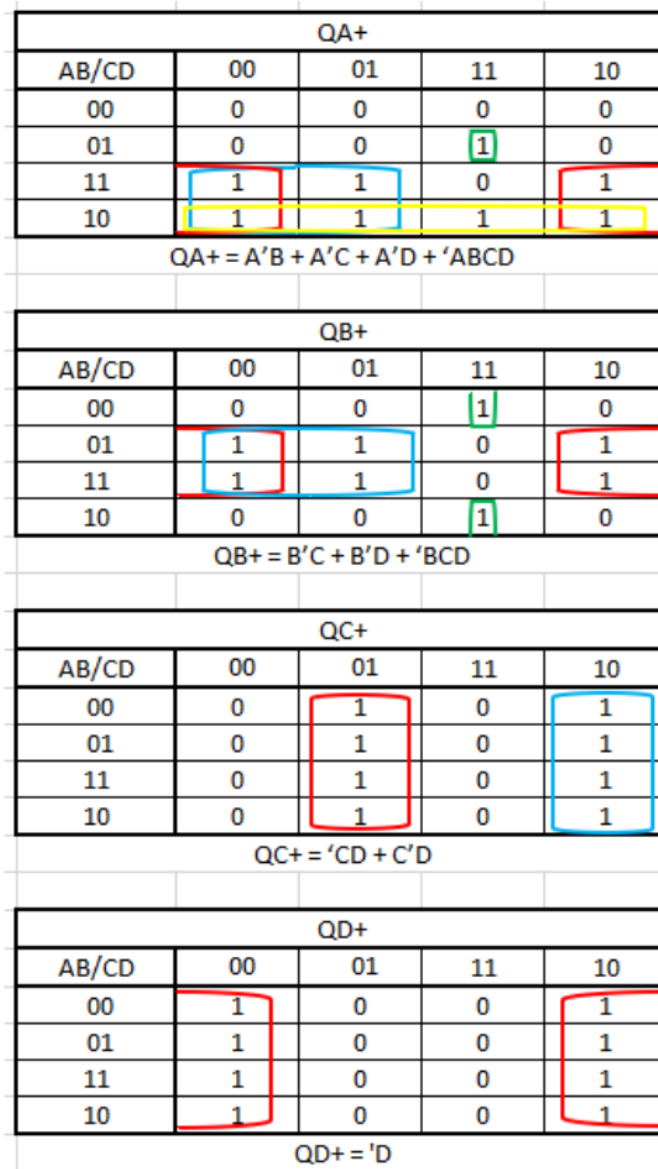


Figure 28 – 15 Second Timer Karnaugh Maps

From these Boolean expressions the 15 second timer circuit was created on multisim. Each individual expression from QA+ to QD+ from the Karnaugh maps were connected to AND gates. These AND gates were then connected to OR gates, a flip flop is then connected to the OR gate output, this flip flop then gives an output of either A or 'A, B or 'B, C or 'C and D or 'D for each part of the respective circuit. (Figure 29)

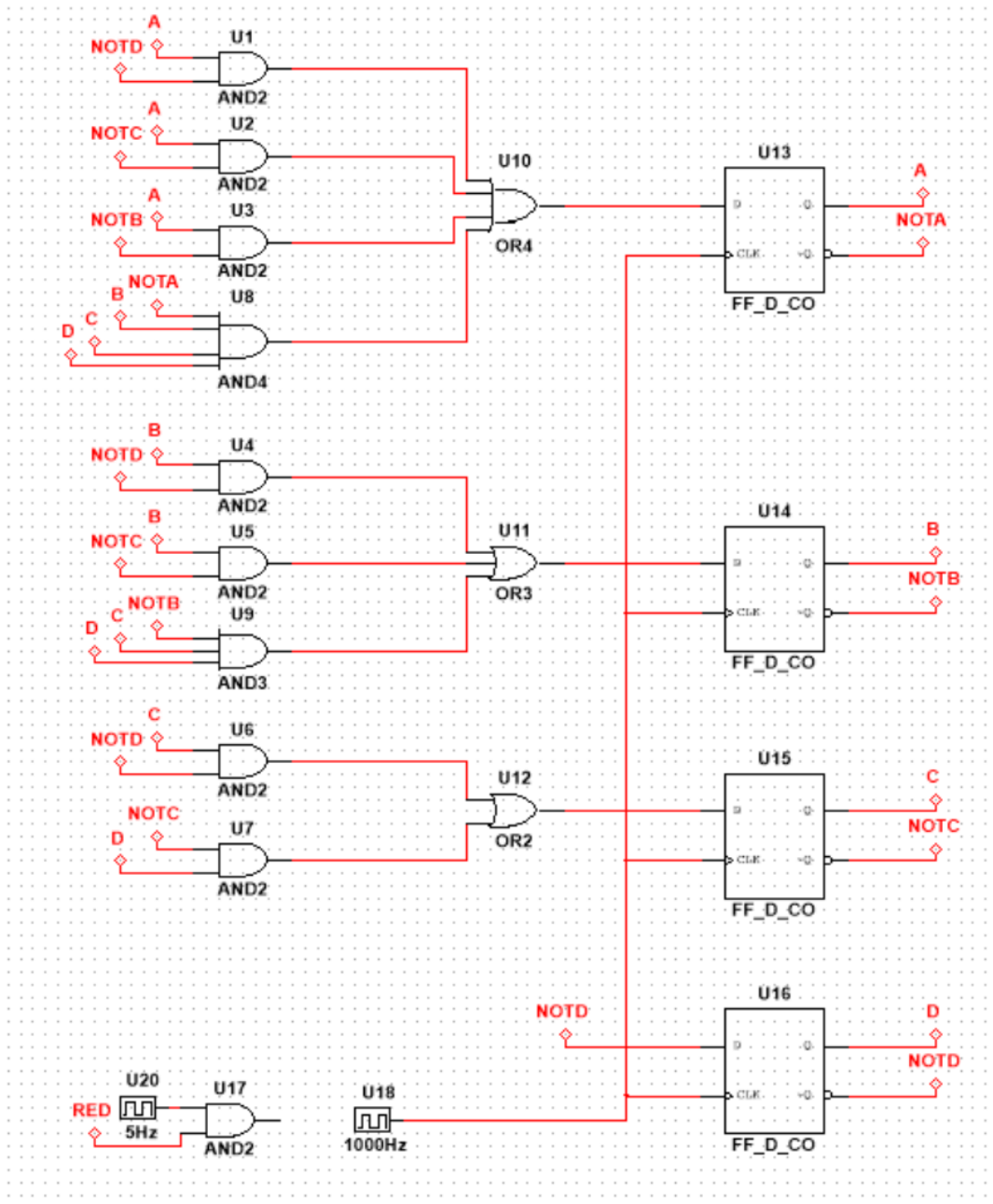


Figure 29 – 15 Second Timer Circuit

After creating all of the sub-circuits, the final circuit is created. This contains all three sub-circuits, Button B sends an input to the 5 second timer to start the circuit, after 5 seconds it sends a wait signal to the RAG controller, after the controller goes through its sequence to red it then sends a signal to the 15 second timer, this then starts a 15 second delay for the pedestrian to cross. After the 15 second time elapses the traffic lights will return to green via a trigger. (Figure 30)

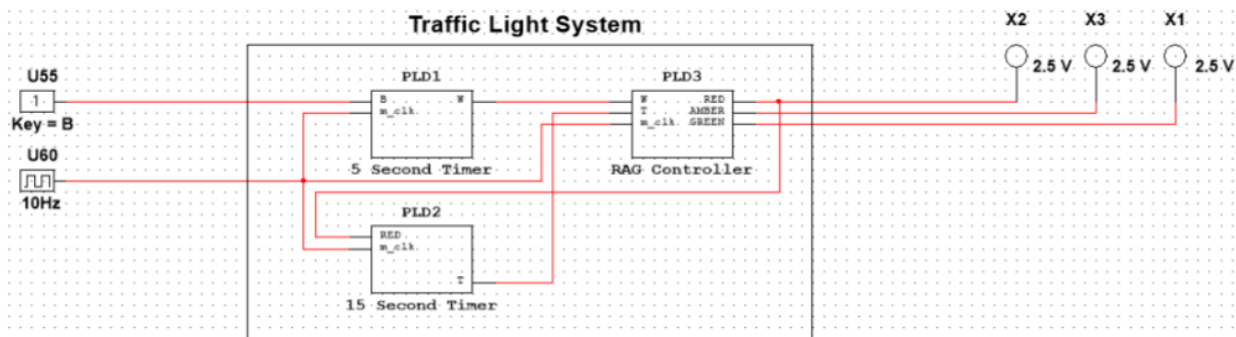


Figure 30 – Traffic Light System Final Circuit

Figures 32 show the circuit LEDs counting through the correct sequence.

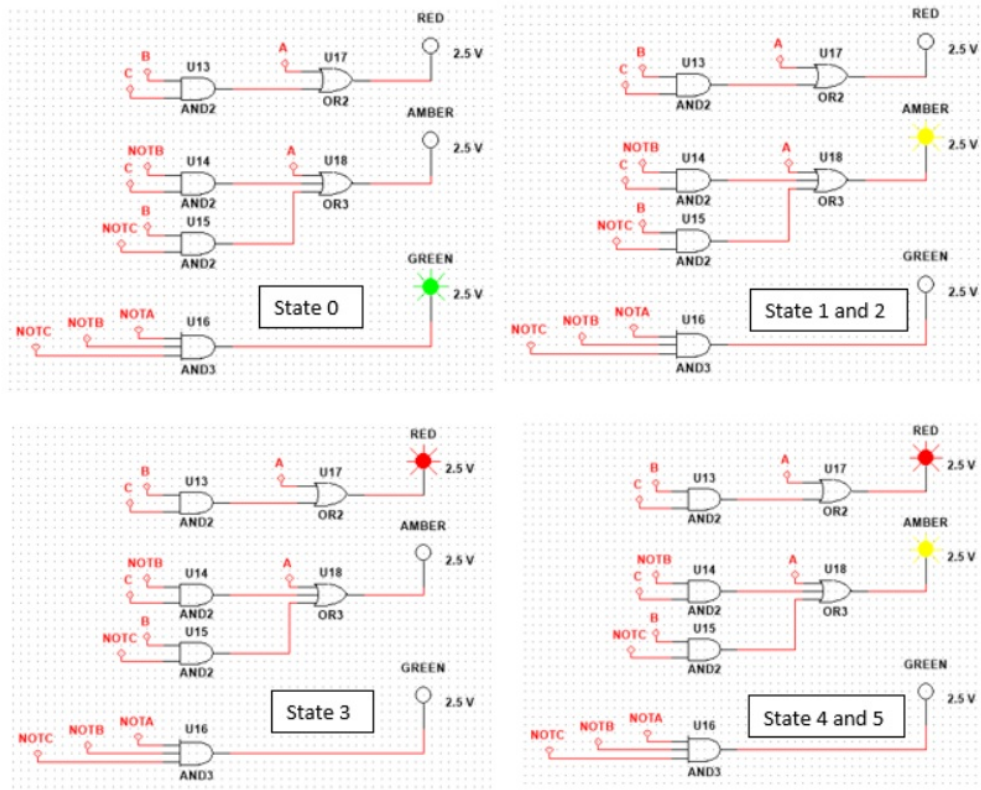


Figure 32 – Sequence Controller Cycling Evidence

The next test was to connect the circuit to a logic analyzer (Figure 33), this is an instrument that captures the electronic signals and then displays it as a timing diagram. The outputs are connected to the analyzer and the diagram will display if its counting through the circuit correctly.

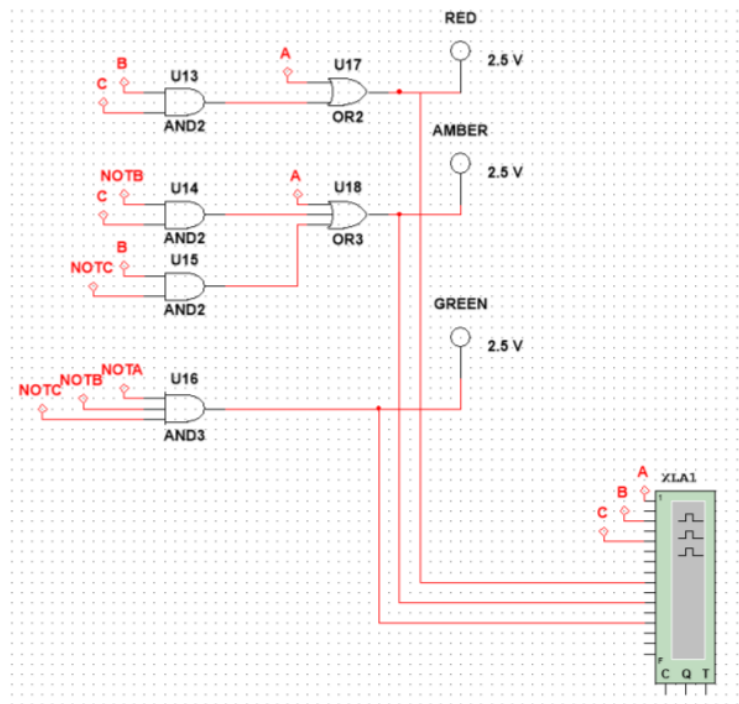


Figure 33 – Sequence Controller with Logic Analyzer

As seen in figure 34, the logic analyzer is counting through the sequence correctly.

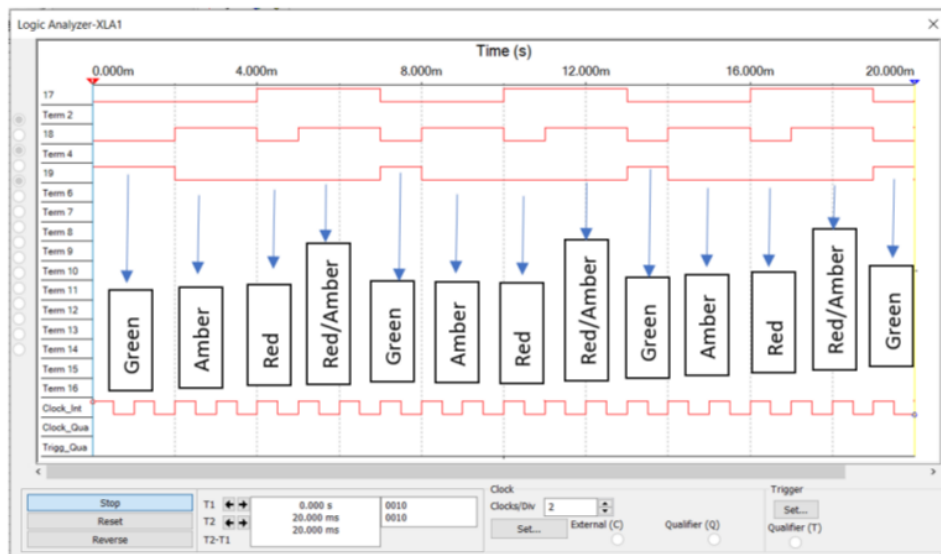


Figure 34 – Sequence Controller Logic Analyzer diagram

The RAG Circuit was then modified into a PLD circuit which was uploaded onto the FPGA board, this was then tested from the board to ensure it worked correctly. (Figure 35)

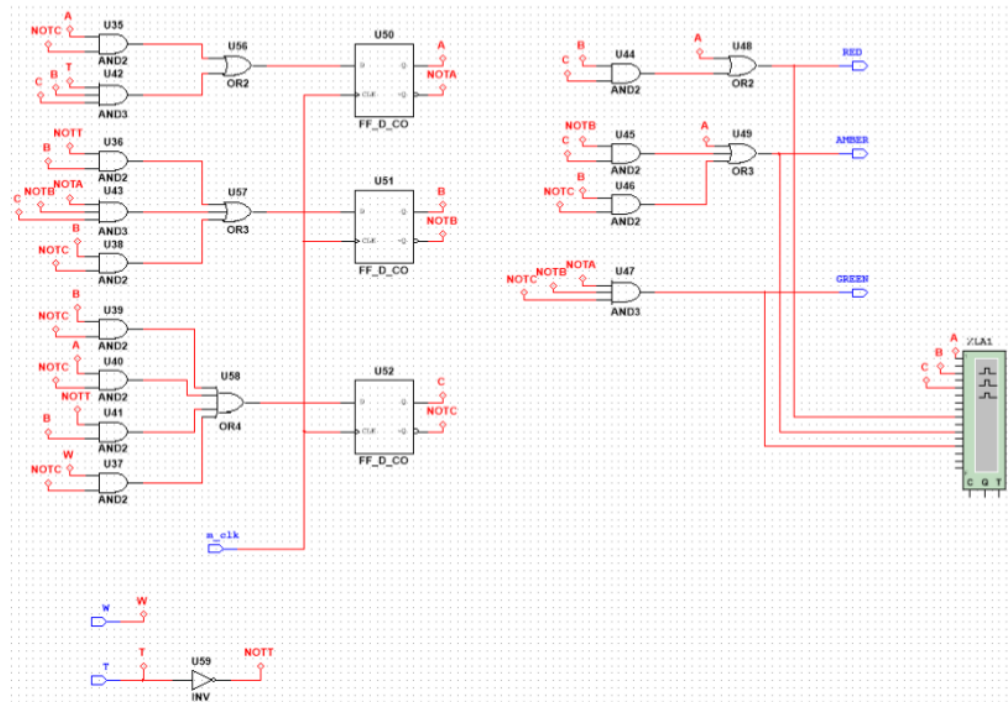


Figure 35 – RAG Controller PLD Circuit

RAG Controller – FPGA LED Test (Press W)		
State	Expected Result	Actual Result
0	Green Illuminates	Green Illuminates
1	Amber Illuminates	Amber Illuminates
2	Amber Illuminates	Amber Illuminates
3	Red Illuminates	Red Illuminates
4	Red/Amber Illuminates	Red/Amber Illuminates
5	Red/Amber Illuminates	Red/Amber Illuminates

Figure 36 – RAG Controller Test Table

As shown in figure 36, the circuit passed the FPGA test as the results matched the expected results, When the wait button was in the low state the light stays at green for the traffic to continue to pass until it is pressed.

4.1.1: 5 Second Timer

Secondly, the 5second timer was tested. For the first test, lights were connected to QA, QB, QC and W, these lights will display the binary count from 000 to 101 in the form of LEDs connected to the outputs. This tests that the circuit is counting as it should be and the output is high when it should be. (Figure 37)

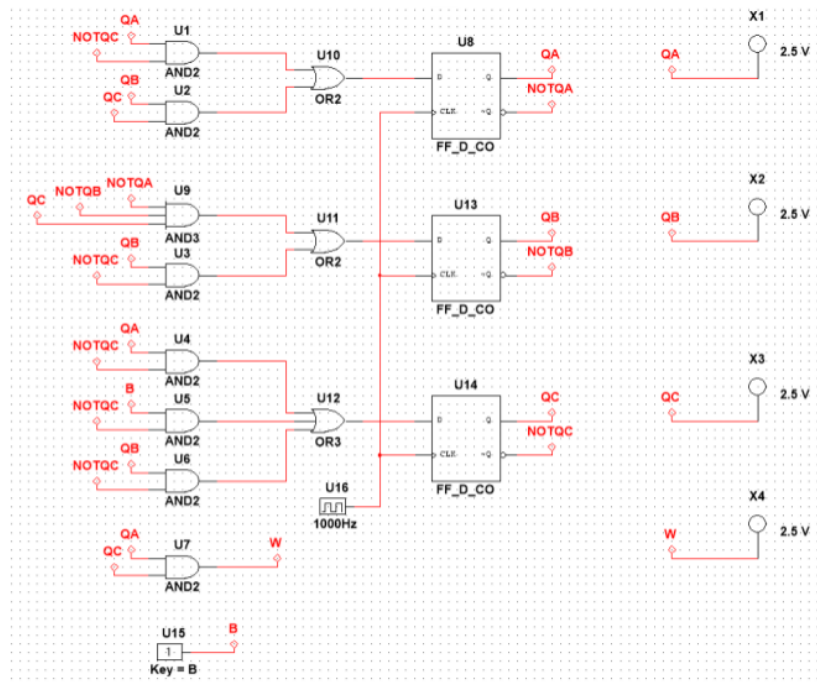


Figure 37 – 5 Second Timer Circuit

Firstly, the circuit was tested when the button was off, no lights illuminated as expected. When the button was pressed the timer counted through the sequence as designed, at 5 seconds the output was high, this is the correct operation. (Figure 38)

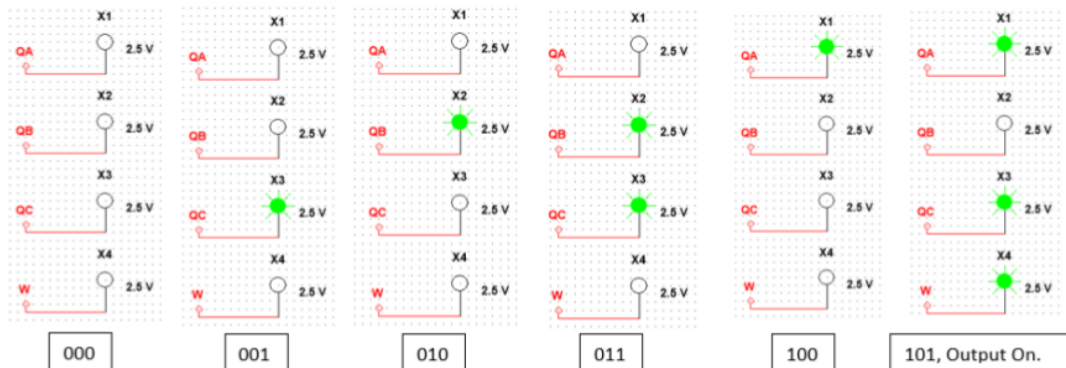


Figure 38 – 5 Second Timer Lamp Test

A logic analyzer was then connected to the circuit to display a timing diagram of the binary count from the logic circuit. The connections from the QA-QC were connected along with the output, it was tested to make sure that “W” is high at 5 seconds (101) (Figure 39)

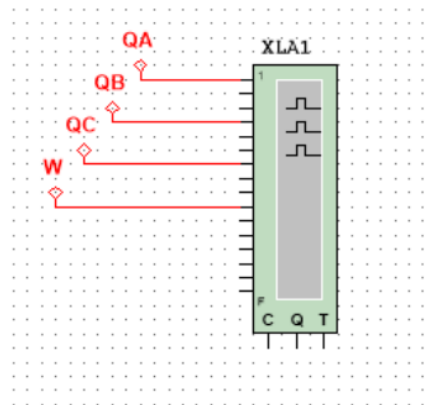


Figure 39 – 5 Second Timer Logic Analyzer Circuit

Figure 40 shows the logic analyzer counting through the sequence correctly.

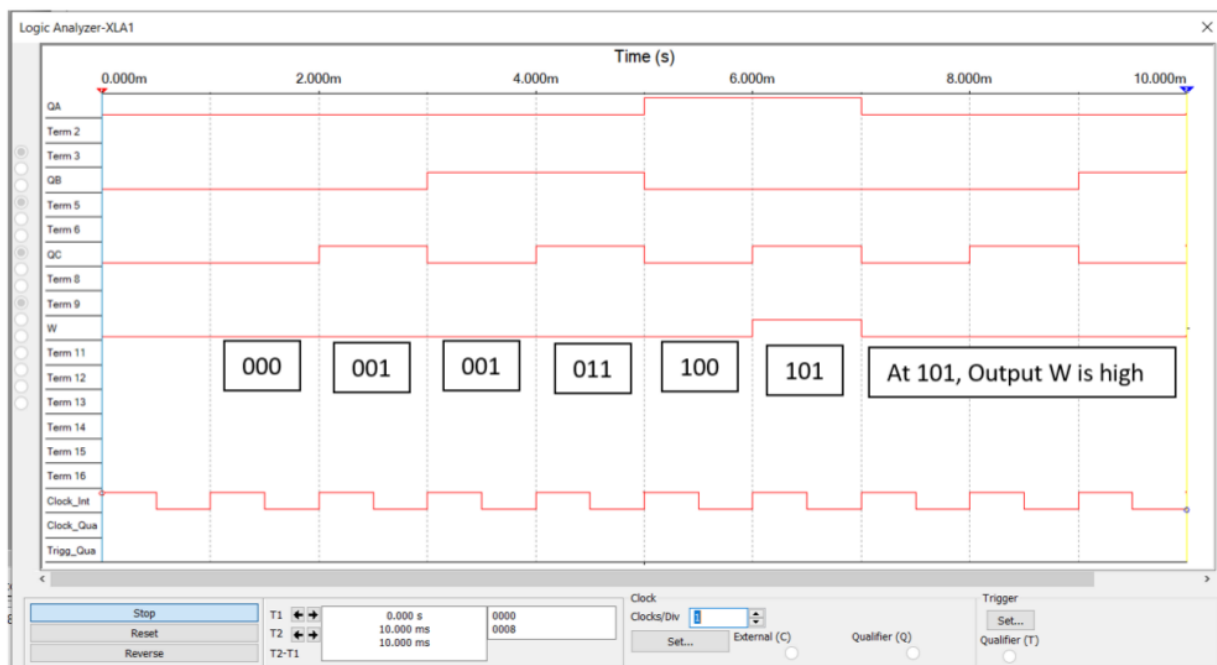


Figure 40 – Logic Analyzer Timing Diagram

Finally, the circuit was modified into PLD mode and uploaded onto a FPGA board, QA, QB, QC and W were assigned to LEDs on the board, as the signals were assigned to LEDs next to each other on the board the binary count was displayed. (Figure 41)

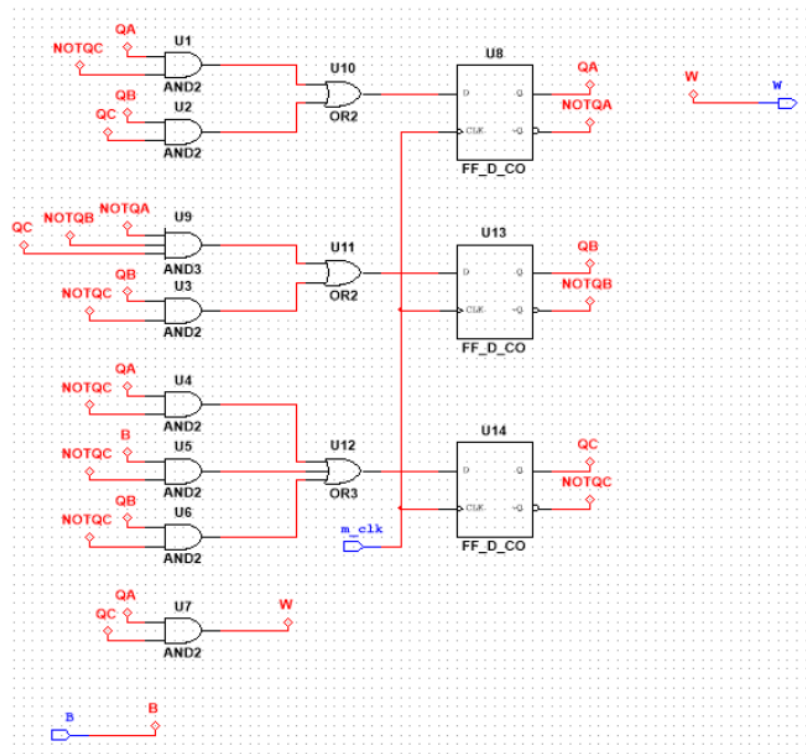


Figure 41 – 5 Second Timer PLD Circuit

5 Second Timer – FPGA LED Test		
State	Expected Results	Test results
0	000	000
1	001	001
2	010	010
3	011	011
4	100	100
5	101	101

Figure 42 – 5 Second Timer FPGA Test

As shown in figure 42 above the test results matched the expected results proving the testing successful.

4.1.2: 15 Second Timer

The final circuit to test is the 15second delay. Similar to the 5second delay, LEDs were connected QA, QB, QC and this time QD, these lights will display the binary count from 0000 to 1111. This tests that the circuit is counting as it should be. (Figure 43)

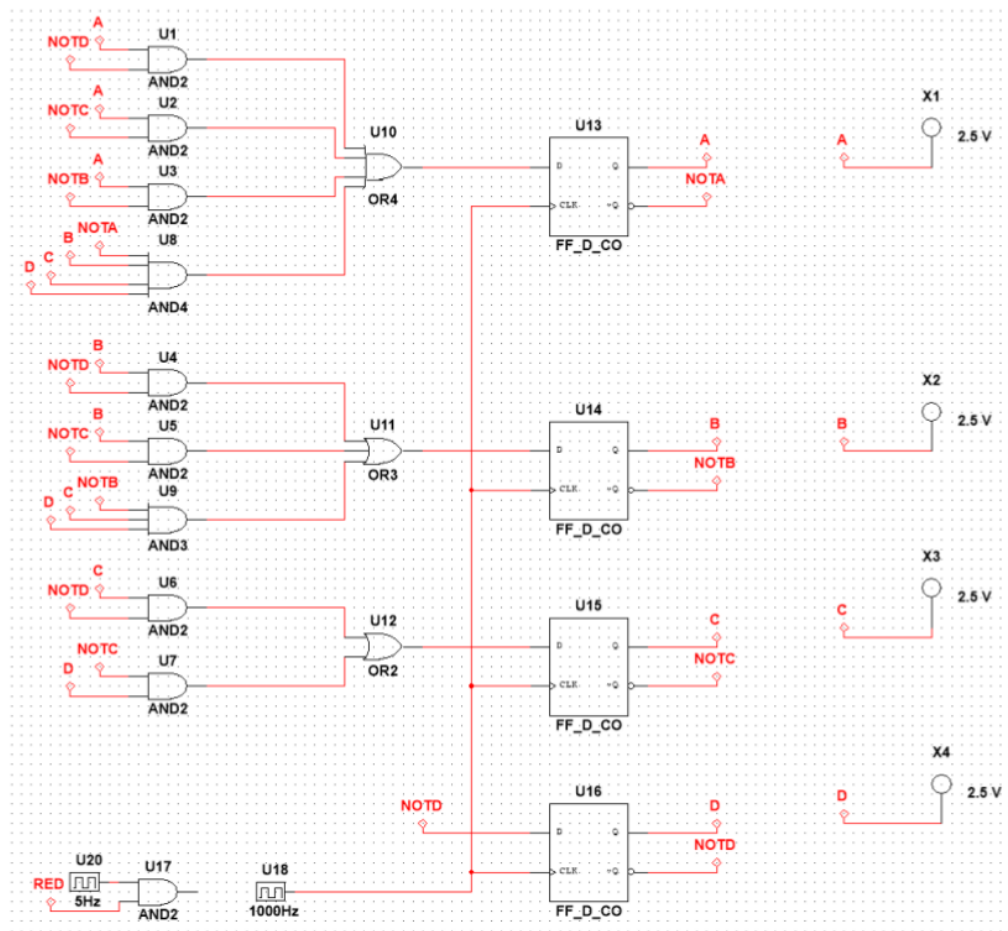


Figure 43 – 15 Second Timer Circuit

The first test for this circuit was to run the circuit to ensure that it is sequencing through the binary count correctly. The evidence of testing is shown in figure 44.

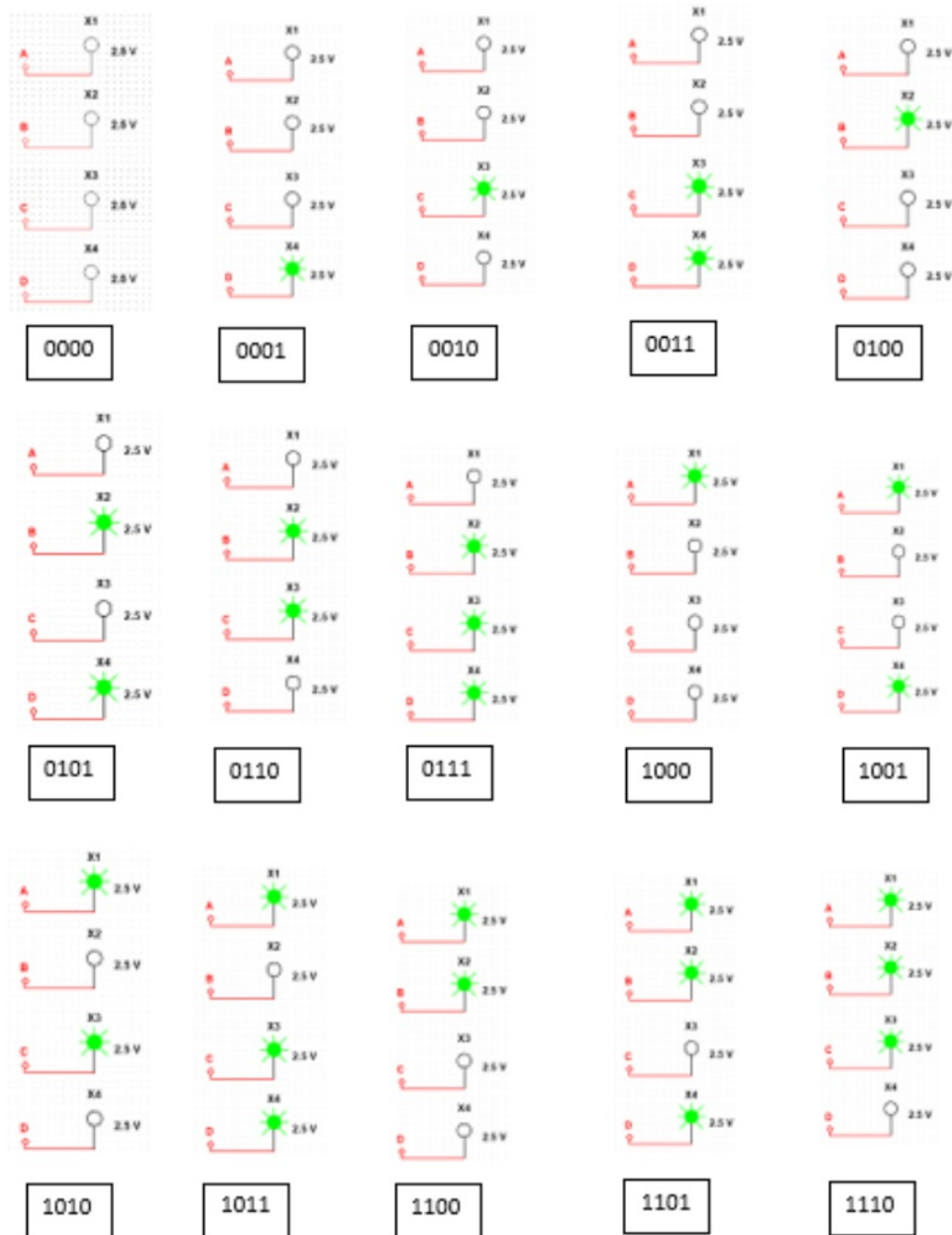


Figure 44 – 15 Second Timer Lamp Tests

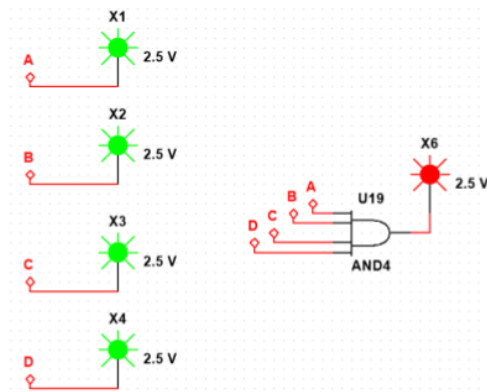


Figure 45 – 15 Second Timer Lamp Test (15s and output on)

This test proved that the circuit was sequencing through the binary count correctly, it was also proved by an AND gate with all outputs connected to prove that at the binary count for 15 the output is high. (Figure 45)

The second test was the logic analyzer test as done previously to obtain a timing diagram of the circuit's operation. The circuit with the analyzer is seen below in figure 46.

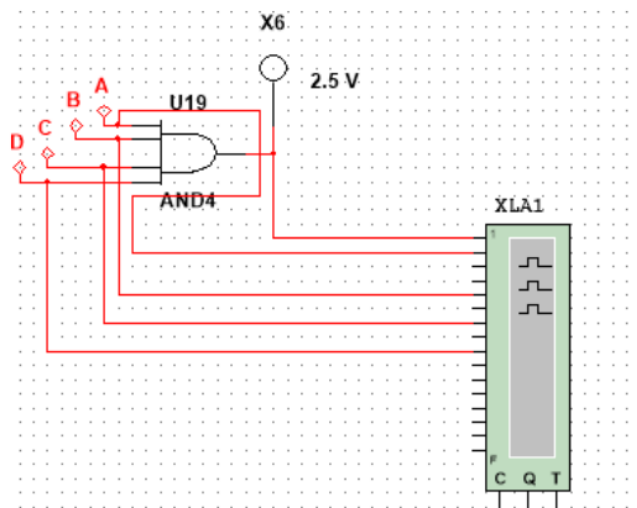


Figure 46 – 15 Second Timer Logic Analyzer Circuit

Figure 47 shows the logic analyzer counting through the sequence correctly, it shows that the output is high at 15seconds.

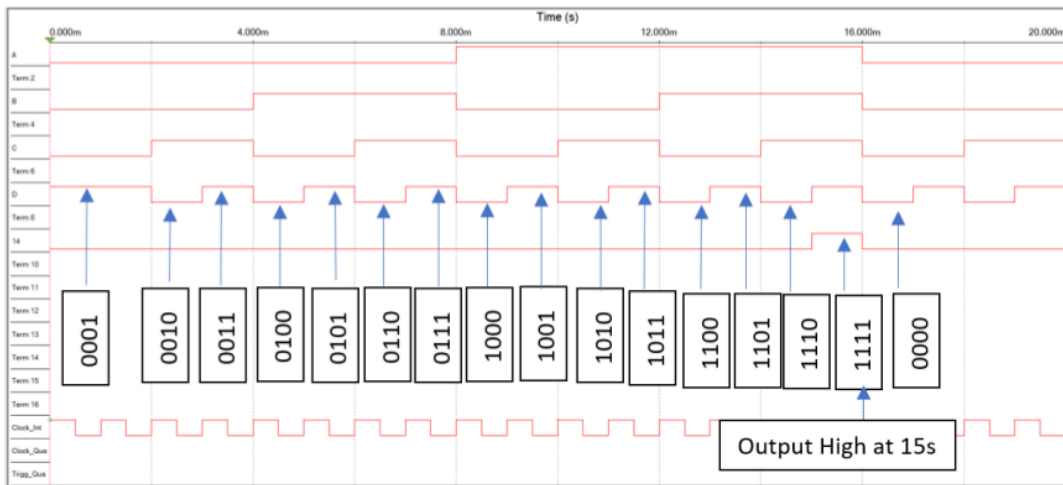


Figure 47 – 15 Second Timer Logic Analyzer Timing Diagram

The 15second circuit was modified into a PLD circuit and was uploaded onto a FPGA board, QA, QB, QC and QD were assigned to LEDs on the board, as the signals were assigned to LEDs, these LEDs displayed the binary count. This was then tested from the board to ensure the sequence was correct on the fpga.

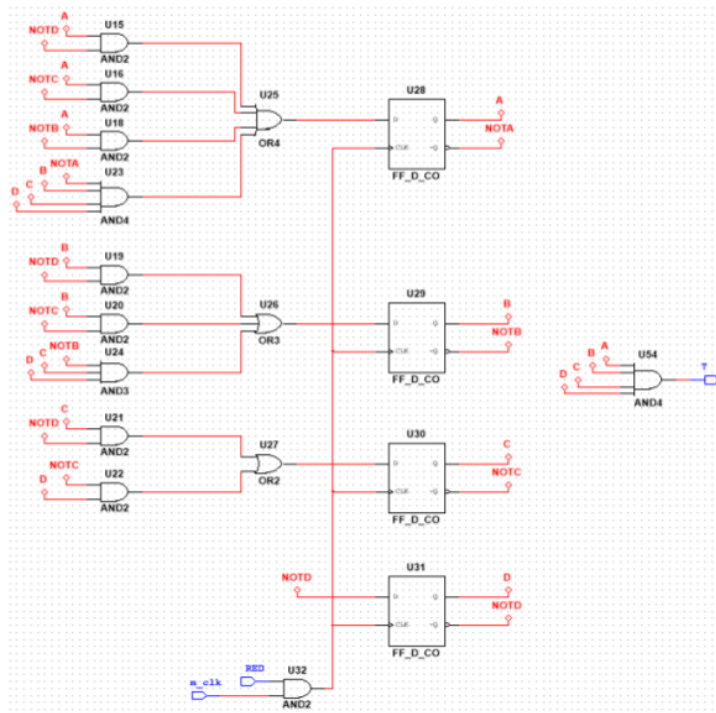


Figure 48 – 15 Second Timer PLD Circuit

15 Second Timer – FPGA LED Test		
State	Expected Result	Test Result
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	1010
11	1011	1011
12	1100	1100
13	1101	1101
14	1110	1110
15	1111	1111

Figure 49 – 15 Second Timer FPGA Test Table

As seen in figure 49, the test results matched the expected results proving that the circuit was working as designed.

4.2: Final Circuit Test

After testing the three sub-circuits and proving that they work correctly, they were all integrated into one circuit, this circuit was then tested on multisim to prove that it is going through the sequence correctly. (Figure 50)

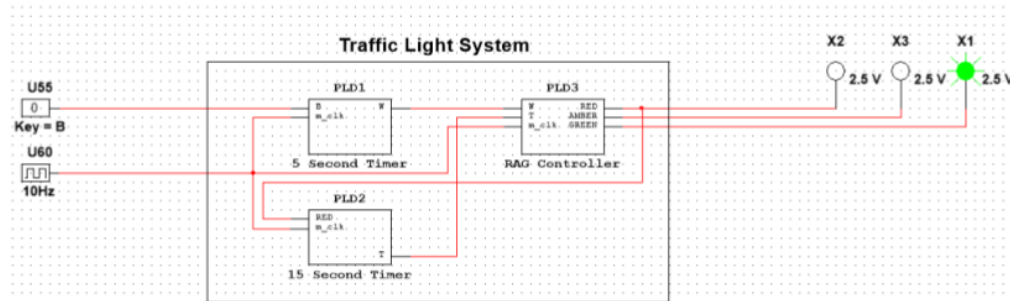


Figure 50 – Final Traffic Light System

When the button is pressed it sends a signal to start the 5second timer circuit, after 5 seconds this circuit sends a signal to the RAG controller to change the traffic light to red. When the traffic light changes to red a signal is sent to the 15 second timer, this timer keeps the traffic lights on red for 15 seconds and the pedestrian lights on green. After the 15 seconds elapses, the traffic lights turn to red/amber and then back to green as designed. Figure 51 shows the final circuit cycling through the sequence correctly.

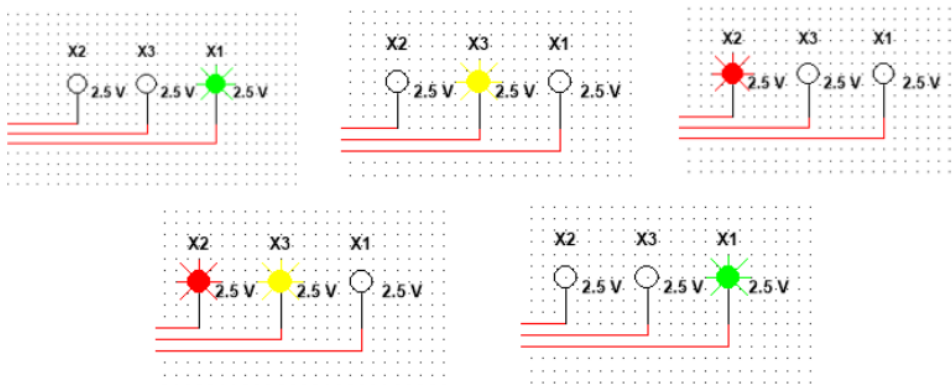


Figure 51 - Final Traffic Light System Lamp Test

The complete circuit was then modified into PLD mode and uploaded onto the Nexys DDR board (Figure 53). A function generator was connected to the board (JA1)(Figure 52), which gave the frequency signal replacing the clock. This has a square step wave of 1Hz so the clock pulses every second. Button C from the Nexys board will start the sequence as it is acting as the pedestrian button.

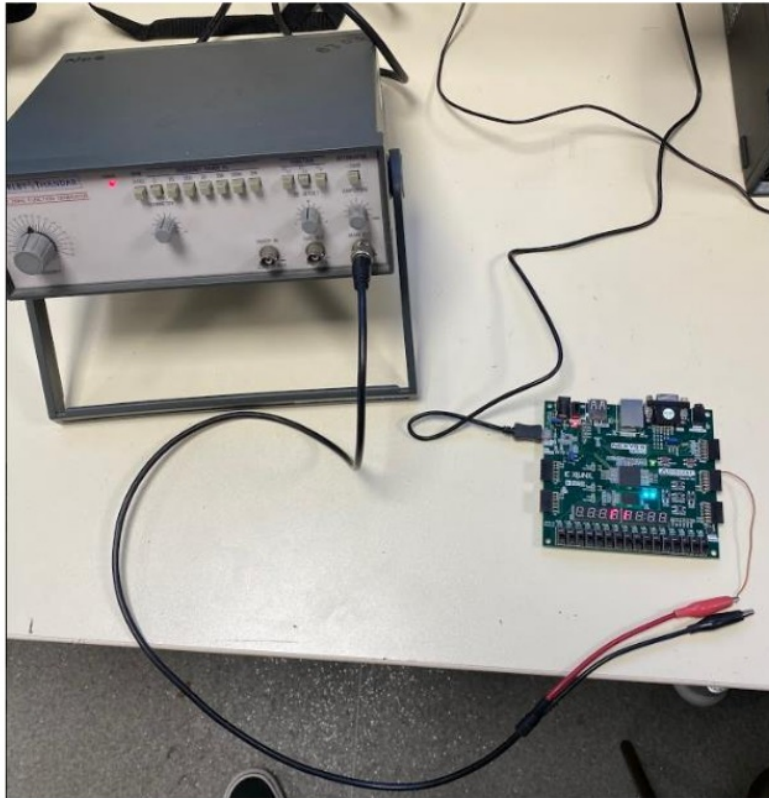


Figure 52 - FPGA Board and Frequency Generator

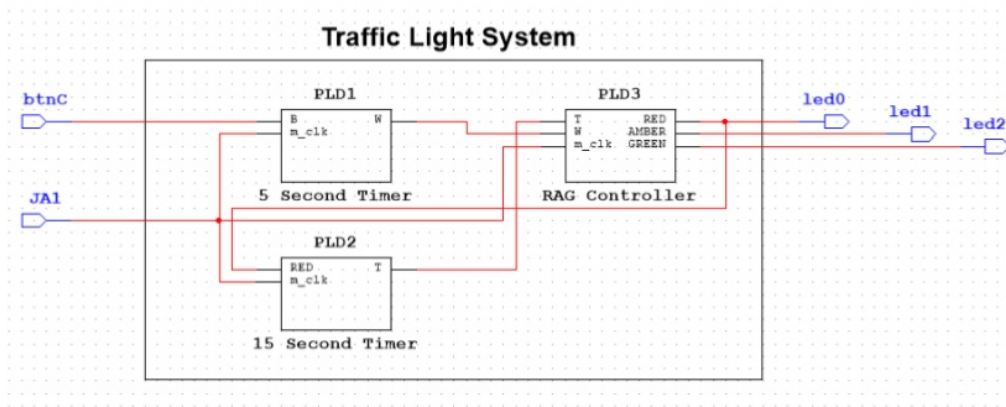


Figure 53 - Final Traffic Light System PLD

A test table was created with tests displaying the test, what should the circuit do, and what happened.

Test	What should happen?	What did happen?
Is the LED assigned to the green light illuminated?	The corresponding LED should be lit.	The correct LED was lit.
Push the wait button.	5 second delay circuit starts.	5 second delay circuit started.
Wait	After the 5 second delay the lights should change from green to red.	After the 5 second delay the lights changed from green to red.
Wait	The sequence should go through traffic light sequence, it should then wait for the input to restart.	The sequence finished through the full sequence.
Push the wait button again	The full sequence should restart.	The full sequence restarted.

Figure 54 - Traffic Light System Final Test

As seen in figure 54, the circuit performed as it should proving the circuit is correct.

4.3: Modifying and Improving the Design

The circuit now needs to be modified; this modification will turn the pelican crossing into a puffin crossing. An extra input will need to be added in order to prove that a pedestrian is waiting to cross the road, this would stop a pedestrian pressing the button and stopping the traffic with no intention of crossing the road.

This extra input will be a pressure pad, the pedestrian will have to press the button and remain on the pressure pad for 5 seconds to start the sequence to cross the road. If the pedestrian moves from the pressure pad then the circuit will reset back to its normal state.

Firstly, a state diagram was created for this modification, this diagram will make it easier to write the truth table as it is visual aid of how the circuit will work in each state. The diagram shows that the Button (B) and Pressure Pad (P) have to be high to move from state 0 to state 5, if there is a pedestrian on the pressure pad (P) at state 5 the 5 second timer will start, if there is no pedestrian on the pressure pad it will return to state 0 and will not hold up any traffic. (Figure S5)

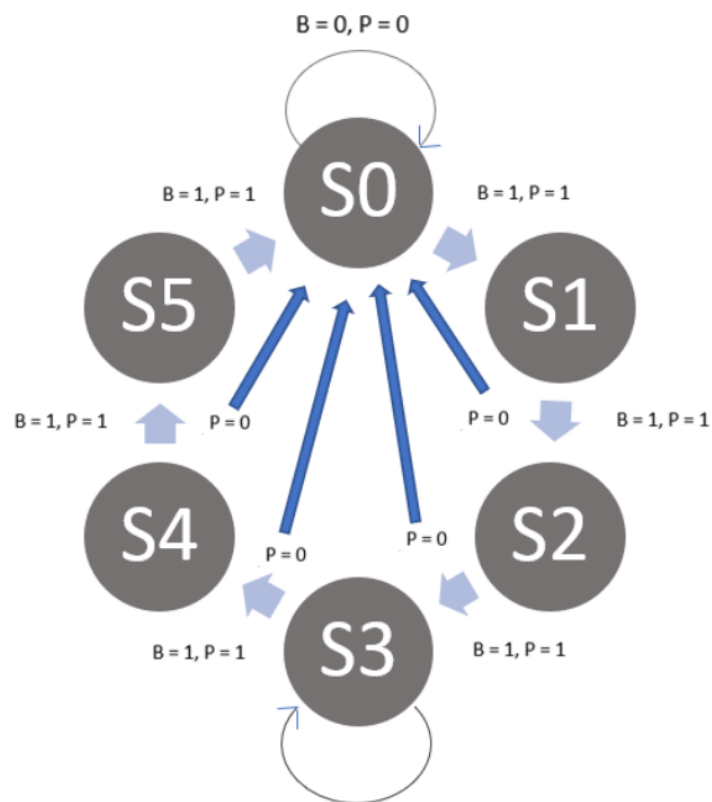


Figure S5 - Puffin Modification State Diagram

With the state diagram designed, the truth table can now be designed, this table is similar to that of the original 5 second timer, however, an extra input has been added for the pressure pad. For this truth table both inputs (B and P) have to be in the high state to count through the sequence, if an input goes from high to low then it will reset back to S0 (Figure 56)

Current	Next				Current			Next												Output (W)
	B = 0		B = 1					B = 0						B = 1						
	P = 0	P = 1	P = 0	P = 1				P = 0			P = 1			P = 0			P = 1			
					QA	QB	QC	QA	QB	QC	QA+	QB+	QC+	QA+	QB+	QC+	QA+	QB+	QC+	
S0	S0	S0	S0	S1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
S1	S2	S2	S0	S2	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	
S2	S3	S3	S0	S3	0	1	0	0	0	1	1	0	1	1	0	0	0	1	1	
S3	S4	S4	S0	S4	0	1	1	1	1	0	0	1	0	0	0	0	0	1	0	
S4	S5	S5	S0	S5	1	0	0	1	0	0	1	1	0	1	0	0	0	1	0	
S5	S0	S0	S0	S0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	
S6	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S7	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

Figure 56 - Puffin Modification Truth Table

From this truth table the Karnaugh maps can be mapped, these were mapped in two separate maps, one is when B is low and the other when B is high. (Figure 57)

B = 0					B = 1				
QA+					QA+				
PQA / QB QC	00	01	11	10	PQA / QB QC	00	01	11	10
00	0	0	1	0	00	0	0	0	0
01	1	0	X	X	01	0	0	X	X
11	1	0	X	X	11	1	0	X	X
10	0	0	1	0	10	0	0	1	0
$QA+ = 'BQBQC + 'BQA'QC + PQA'QC + PQBQC$									
B = 0					B = 1				
QB+					QB+				
PQA / QB QC	00	01	11	10	PQA / QB QC	00	01	11	10
00	0	1	0	1	00	0	0	0	0
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
$QB+ = 'B'QA'QBQC + 'BQB'QC + PQA'QBQC + PQB'QC$									
B = 0					B = 1				
QC+					QC+				
PQA / QB QC	00	01	11	10	PQA / QB QC	00	01	11	10
00	0	0	0	1	00	0	0	0	0
01	1	0	X	X	01	0	0	X	X
11	1	0	X	X	11	1	0	X	X
10	0	0	0	1	10	1	0	0	1
$QC+ = 'BQA'QC + 'BQB'QC + BP'QC$									
W									
QA / QBQC	00	01	11	10					
00	0	0	0	0					
01	0	1	X	X					
$W = QAQC$									

Figure 57 – Puffin Modification Karnaugh Maps

Once the Karnaugh maps were complete the logic circuit was created. As NotB was needed for the circuit “B” was placed through an inverter gate to get “NotB” (Figure 58)

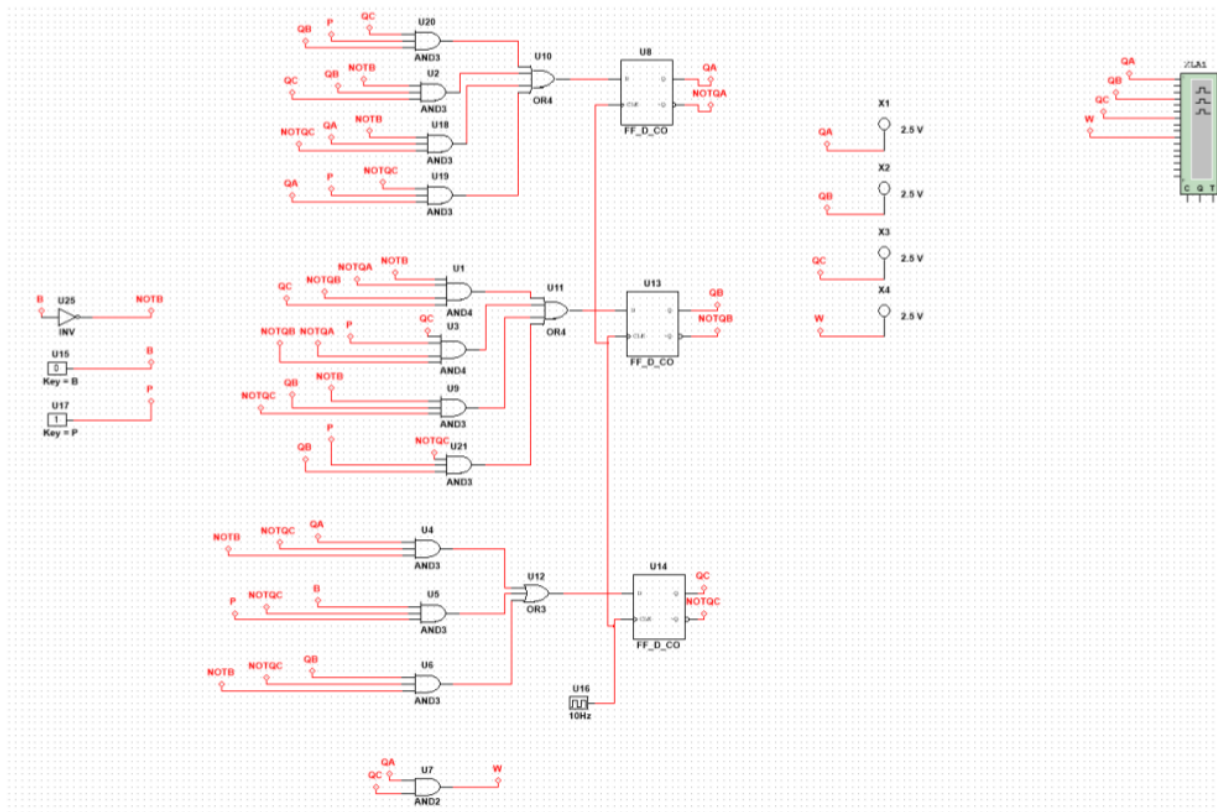


Figure 58 – Puffin Modification Circuit

After the circuit was created it was then tested using a logic analyser to ensure it is counting through the sequence correctly. As shown in figure 59 it was counting through correctly and the output was high at 5seconds.

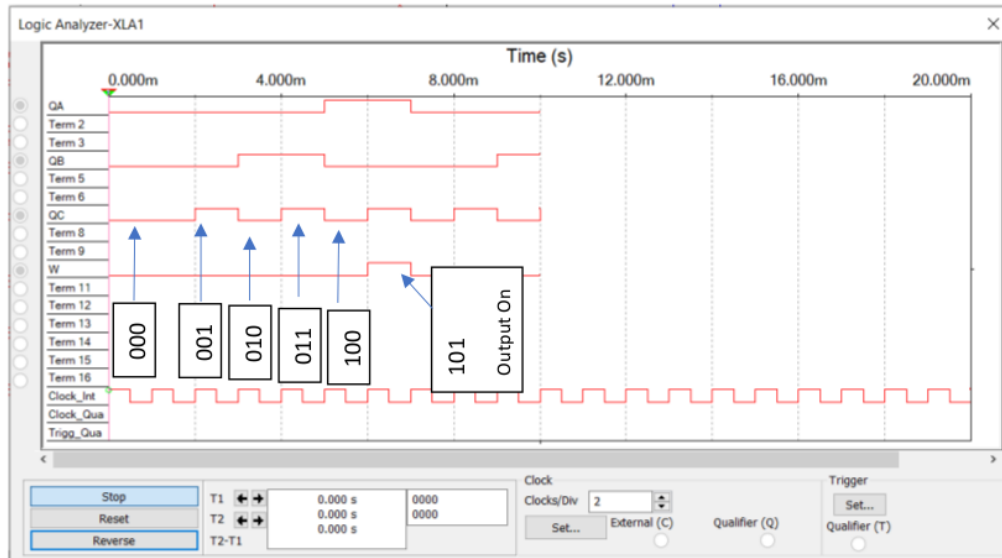


Figure 59 – Puffin Modification Logic Analyzer

The circuit was then checked to see if the correct that the LEDs were lighting up in the sequence.
(Figure 60)

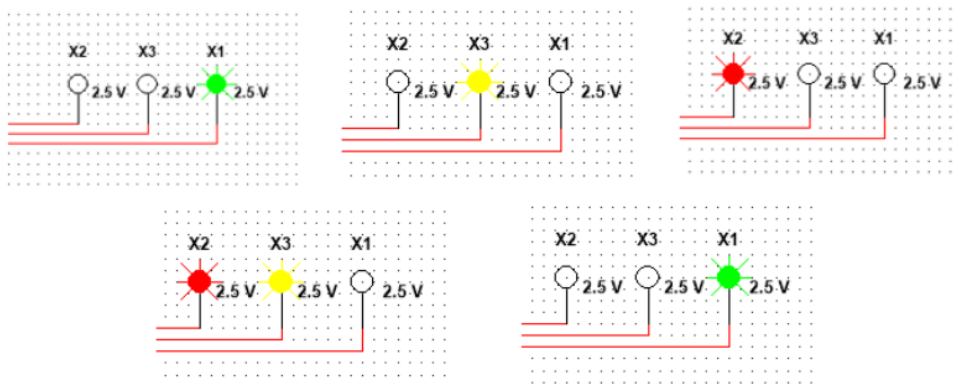


Figure 60 – Puffin Modification Lamp Test

The final stage of the modification is to create a new sub-circuit for the new modified 5 second timer with the extra input from the pressure pad, this is labelled on the sub-circuit as btnP. (Figure 61)

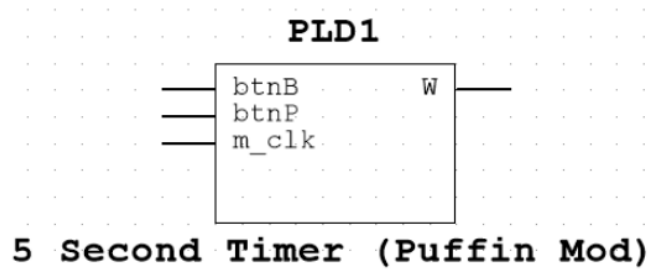


Figure 61 – Puffin Modification PLD

The new sub-circuit was connected to the pre-existing master circuit with the three sub-circuits, clock input signal, buttons for the pressure pad and crossing button. The lights are connected to the RAG controller. (Figure 62)

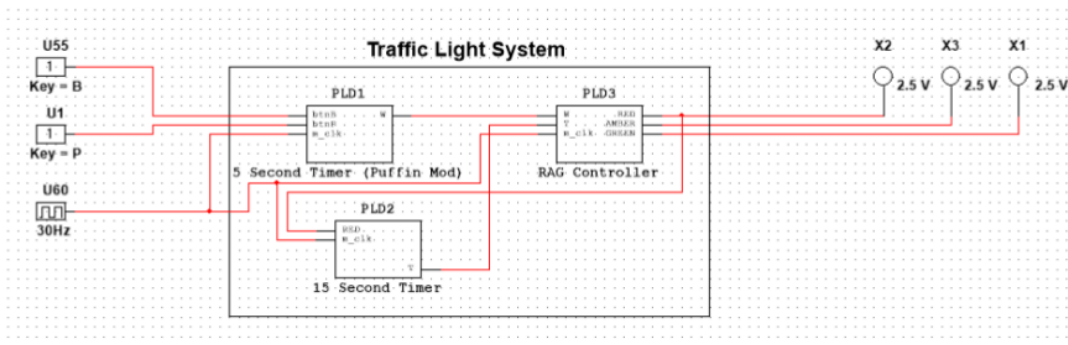


Figure 62 – Final PLD Circuit with Puffin Modification

The final task for the modification is to test its functionality to prove that the modification is working to the designed specification.

Modified Circuit Test		
State	Expected Results	Test results
Press B and P	The timer should run through the sequence and then bring on the output.	The timer ran through the sequence, the output was high at 5 seconds.
Turn off B and P	Nothing should happen, timer should not operate	Timer did not operate
Press B and P, then turn B off.	The timer should count to the end of the sequence.	The timer counted to the end of the sequence.
Press B and P, then turn P off.	The sequence should stop immediately.	The sequence stopped immediately.

Figure 63 – Puffin Modification Test Table

As seen in figure 63, the test results matched the expected results, proving that the circuit meets the specification and that the circuit is working as it was expect.

5. Conclusions and Recommendations

In conclusion the circuit design and testing were successful, the circuit performed as designed in all three sub-circuits and each sub-circuit were seamlessly integrated together to make the full traffic light system whilst meeting the overall specification.

Overall great knowledge was learnt about digital electronics, knowledge about the principles of state diagrams, Karnaugh maps, Boolean expressions and logic circuits was gained and also learning how to utilise PLD mode on multisim in order to test and run the circuits, great experience was also gained from the modification task as knowledge was obtained on how to edit state diagrams and truth tables in order add new aspects to a circuit.

From the research task useful information about sequential and combinational circuits was gained along with the benefits of using a finite state machine, this research was extensively beneficial in the circuit design, testing and modification sections as it gave an underpinning knowledge and key principles on how sequential circuits work.

After researching about FPGAs, it also gave an insight into the Nexys 4 DDR board which was used for the testing of the circuit, this was a great benefit during the testing stage.

The system could be implemented and developed for an ASIC; ASICs are highly complex with extensive functionality with the ability of dealing with 100million+ logic gates. FPGAs are very useful for research and prototyping, however, for high volume designs and implementation such as traffic light systems ASICs are extensively cheaper per unit and they will have a lower “form factor” as it is manufactured to a specific specification. ASICs are also quicker than FPGAs due to their higher clock speeds. (Brogioli, 2012)

ASICs also have many key benefits as seen in figure 64.

1. Higher integration
2. Smaller footprint
3. Lower power consumption
4. Higher functionality
5. Intellectual Property protection
6. Ease of final manufacture
7. Obsolescence guarantee
8. Proven development and logistic routes

Figure 64 - Benefits of using an ASIC

The system as a whole can be integrated to improve sustainable development greatly from the use of ASICs, as the prototype and testing has been completed successfully it can now be modified for an ASIC. By doing this councils will be able to get traffic light systems much cheaper due to the low recurring component cost along with the rugged environments that these chips can operate in which can reduce downtime and will be more eco-efficient as cars will not be on stand-still while traffic lights are down. If an ASIC was to go into fault a new one would easily be installed at a low price.

As a recommendation for the systems improvement, an audible beeper can be integrated into the traffic light system as an output which will be connected to the traffics red light. This would be beneficial to members of the public who are visually impaired as it would be an indication to them that it is safe to cross the road.

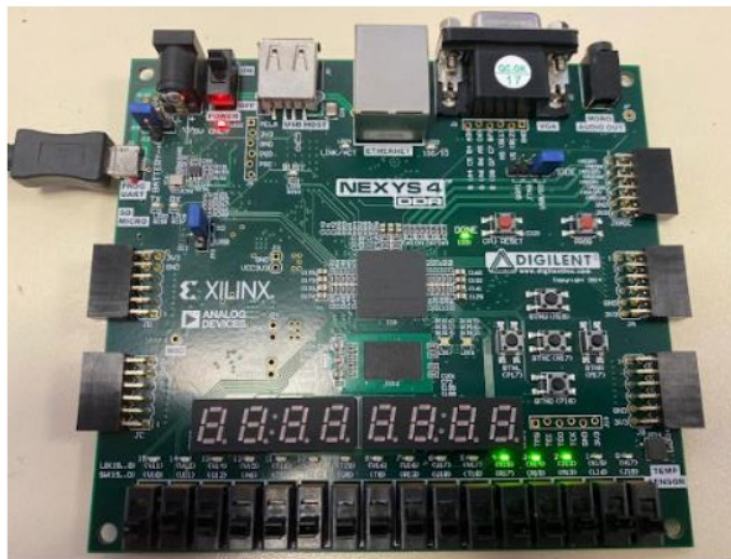
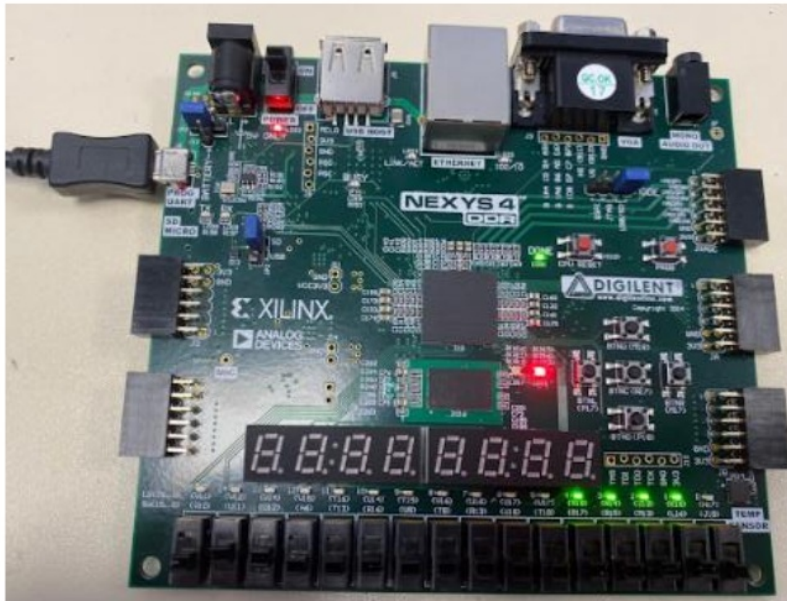
6. Reference list and Bibliography

- Bors, M. L., 2018. *What is a Finite State Machine?*. [Online]
Available at: <https://medium.com/@mlbors/what-is-a-finite-state-machine-6d8dec727e2c>
[Accessed 13 01 2019].
- Broglioli, M., 2012. *The DSP Hardware/Software Continuum*. [Online]
Available at: <https://www.sciencedirect.com/topics/engineering/application-specific-integrated-circuits>
[Accessed 24 01 2020].
- electronics-notes, 2018. *What is an ASIC: application specific integrated circuit*. [Online]
Available at: https://www.electronics-notes.com/articles/electronic_components/programmable-logic/what-is-an-asic-application-specific-integrated-circuit.php
[Accessed 13 01 2020].
- ElectronicsTutorials, 2017. *Sequential logic circuits*. [Online]
Available at: https://www.electronics-tutorials.ws/sequential/seq_1.html
[Accessed 13 01 202].
- elprocus, 2016. *Introduction to Combinational Logic Circuits*. [Online]
Available at: <https://www.elprocus.com/introduction-to-combinational-logic-circuits/>
[Accessed 13 01 2020].
- Singh, R., 2018. *FPGA Vs ASIC: Differences Between Them And Which One To Use?*. [Online]
Available at: <https://numato.com/blog/differences-between-fpga-and-asics/>
[Accessed 13 01 2020].
- Staff, E., 2009. *Using finite state machines to design software*. [Online]
Available at: <https://www.embedded.com/using-finite-state-machines-to-design-software/>
[Accessed 26 01 2020].
- STechies, 2019. *Combinational and Sequential Circuits - Key Differences*. [Online]
Available at: <https://www.stechies.com/difference-between-combinational-sequential-circuits/>
[Accessed 13 01 2020].
- Touger, E., 2018. *What Is an FPGA and Why Is It a Big Deal?*. [Online]
Available at: <https://www.prowesscorp.com/what-is-fpga/>
[Accessed 13 01 2020].
- tutorialspoint, 2018. *Moore and Mealy Machines*. [Online]
Available at: https://www.tutorialspoint.com/automata_theory/moore_and_mealy_machines.htm
[Accessed 13 01 2019].
- Wilson, P., 2015. Introduction. In: *Design Recipes for FPGAs (2nd Edition)*. s.l.:Elsevier.
- Xilinx, 2018. *What is an FPGA? Field Programmable Gate Array*. [Online]
Available at: <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>
[Accessed 13 01 2020].

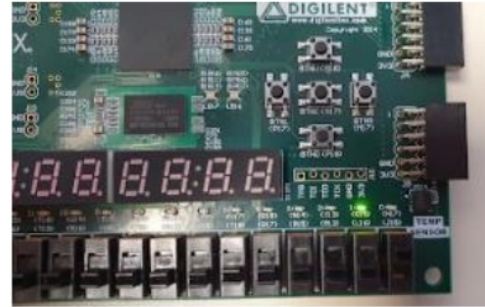
7. Annexe for diagrams, data sheets and results tables

15 Second Timer Evidence

FPGA Board at 15s (Output on)



5 Second Timer Evidence



Unfortunately, when it came to test the final circuit and the RAG controller there were software problems which withheld this.

FINAL GRADE

GENERAL COMMENTS

Instructor

85/100

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

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

GRADING FORM: CSG-RUBIC

THOMAS GRENNAN

85

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 consistently applied and appropriate to the task and your reference list was of an appropriate length for this assignment.

9

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 your understanding of the higher level concepts was demonstrated to a high degree. You have demonstrated a highly competent critical understanding of the main concepts and expanded your knowledge well beyond that contained in the course notes.

27

ANALYSIS



Your analysis of the task was excellent, both in class and as shown in this assignment. You have demonstrated highly competent analytical skills and your approach to the task was highly logical. You carried out a detailed evaluation of the results and included an excellent conclusion relating your practical results to accepted theory. You have demonstrated a capacity for independent thought and your interpretation of the quantitative and qualitative data was excellent with sound judgements based on wholly valid arguments.

27

PRACTICAL APPLICATION AND DEP.



Your application of practical skills for this task was generally excellent. You used both the basic and advanced features of the resources appropriately when designing the circuit and you used appropriate

18

advanced options for testing and evaluating the circuit performance. You managed to design an effective circuit that functioned well within the specification and achieved this by rarely asking for assistance thus demonstrating some advanced problem solving skills. Your results were comprehensive and allowed for a detailed conclusion to be written.

SKILLS FOR PROFESSIONAL PRAC.



Your written report was reasonably well presented in terms of content and layout and was mostly structured in a coherent way, however you should endeavour to use the available word count more effectively in future.

4