

1809809- Finite State Machines Final

by Rhys Davies

FILE	16479_RHYS_DAVIES_1809809- _FINITE_STATE_MACHINES_FINAL_174902_2058052882.PDF (8.82M)		
TIME SUBMITTED	27-JAN-2020 07:23PM (UTC+0000)	WORD COUNT	5622
SUBMISSION ID	119198761	CHARACTER COUNT	22115

HNC Electrical And Electronic Engineering
SGEE5000 - Electronics II

Assignment - Finite State Machines

Author: Rhys Davies (1809809)
28th January 2020

Word Count: 2516

Table Of Contents:

	Page:
List Of Figures:	3
Executive Summary:	4
Research:	4
Designing a traffic light control system	6
Creating the '5 Second' Timer:	7
Creating the '15 Second' Timer:	12
Creating the 'R.A.G Controller':	17
Recommendations:	22
Conclusion:	24
Reference List:	25

List Of Figures:

	<u>Page:</u>
Figure 1: Combinational Circuit forms (El-Pro-Cus, 2020)	4
Figure 2: FPGA vs ASIC Comparison table (Numato Lab, 2020)	5
Figure 3: Ideal Design Sustainability Cycle (BPF, 2020)	6
Figure 4: RAG Controller State Machine	7
Figure 5: Logic Table & Diagram of a 5 Second Delay	7
Figure 6: State table of the 5 Second Delay timer	8
Figure 7: karnaugh maps of the 5 second timer	8
Figure 8: Logic Gate diagram For '5 Second Timer'	9
Figure 9: Multisim diagram For '5 Second Timer'	9
Figure 10: Multisim diagram For 5 Second Timer With Analyser	10
Figure 11: NEXYS DDR 4 board testing 5 Second Delay timer	10
Figure 12: NEXYS DDR 4 board with 5 Second Delay timer showing S1 to S5 counts with 'W' Output	11
Figure 13: logic table & State diagram For 15 Second Timer	12
Figure 14: State Table For 15 Second Timer	13
Figure 15: karnaugh maps of THE 15 second timer	13
Figure 16: Logic Gate Diagram For 15 Second Timer	14
Figure 17: Multisim diagram For 15 Second Timer	15
Figure 18: Multisim diagram For 15 Second Timer With Analyser	15
Figure 19: Nexys DDR 4 board with 15 Second Delay timer showing 'S15' count with 'W' Output	16
Figure 20: Logic table and diagram For THE 'R.A.G Controller'	17
Figure 21: State table of THE 'R.A.G Controller'	17
Figure 22: karnaugh maps of THE 'R.A.G Controller'	18
Figure 23: Logic Gate Diagram For the 'R.A.G Controller'	19
Figure 24: multisim circuit of THE 'R.A.G Controller'	20
Figure 25: PLD Circuit of the 'R.A.G Controller'	20
Figure 26: real world testing OF THE 'R.A.G Controller'	21
Figure 27: Logic Table & diagram of the 5 Second Delay timer (Modified)	22
Figure 28: State table of the 5 Second Delay timer (Modified)	22
Figure 29: karnaugh maps of THE modified '5 second timer'	23
Figure 30: PLD design of THE modified 'RAG Controller'	24

Executive Summary:

This assignment is a brief look into how the electronics industry design, the processes they use to create 'hardware' that enable sequential circuits with external 'inputs' and 'outputs', to be created.

It also covers an example into the design process of a typical traffic light controller, utilising the use of a FPGA circuit board called a 'Nexys 4 DDR' circuit board. The design process will cover the process from initial design spec, creating logic diagrams, tables and resulting logic gate circuits, together with some programming via MultiSim and then testing with the FPGA board. This assignment covers the 'virtual' side of circuit testing before transferring the circuit to a 'real world' example, showing how the programming of 'input' and 'output' components helps you create external signals to interact with the circuit.

Research:

According to Electronics Tutorials: 'Combinational Logic Circuits are made up from basic logic NAND or NOR gates that are "combined" or connected together to produce more complicated switching circuits.' (Electronics Tutorials, 2020). and 'produce an output based on input variables only.' (University Of Surrey, 2020)

The key feature is that the output is not connected or associated with the input in anyway and as a result, 'are **memoryless** digital logic circuits whose output at any instant in time depends only on the combination of its inputs'. (Electronics Tutorials, 2020)

Combination circuits can be specified in three separate forms:

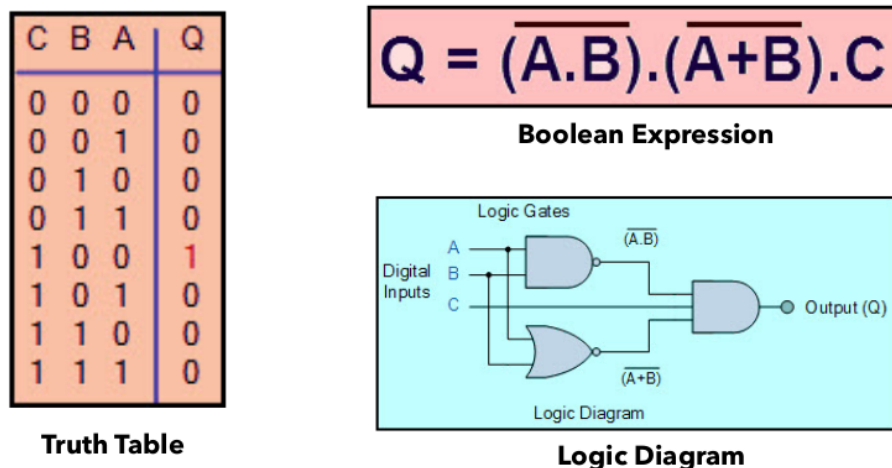


FIGURE 1: COMBINATIONAL CIRCUIT FORMS (EL-PRO-CUS, 2020)

Sequential circuits on the other hand, are similar to combinational circuits but have 'memory' which mean that they 'are able to take into account their previous input state as well as those actually present, a sort of "before" and "after" effect is involved' (Electronics Tutorials, 2020)

Sequential Logic has two modes:

Transparent mode: input propagates to output

Latch mode: the output is stored (also called opaque mode) (ETH Zurich, 2017)

Sequential circuits are also classified under two types of circuits:

Synchronous type circuits rely on an external 'clock' pulsed type of input, which are independent to the control inputs, to advance or change the behaviour of the circuit'

Asynchronous sequential circuits do not use a clock signal as synchronous circuits do. Instead the circuit is driven by the pulses of the inputs.

When it comes to using FSM in design, (Embedded, 2009) says that 'A common approach these days is to apply design patterns when planning and implementing a software project. A finite state machine (FSM) is one design pattern'

The article mentions numerous benefits to the whole design process which include:

- Provides a tool for debugging in case of issues
- Can be valuable at all stages of development of a project
- Creates an easy visual design process which makes logic easier to understand

FSM diagrams are used by numerous industries and professions where their goal was to make a system whilst 'trying to model the human thought process' (ITProPortal, 2020) so the system would be much easier to understand and therefore more adopted as its favoured by its simplicity and effectiveness.

A field-programmable gate array (FPGA) is a circuit/system that is universal and programmable 'on-site', to meet the needs of the consumer. The benefits of this approach is that this approach can be mass produced at a reduced cost, making it very attractive for the DIY enthusiast where the board has many different uses because of its 'programmable' and 'universal' design. This type of board does require additional learning of various programming languages to use this type of board but a development of this language means a more deeper understanding and usage of a FPGA system.

An application-specific integrated circuit (ASIC) is a circuit/system which has been specifically designed to operate in a certain and defined way. The advantages is that they come 'complete' and fully tested with defined limits, tolerances and error functions. As an advantage, the operation of ASIC devices require no programming and very little knowledge to implement.

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

FIGURE 2: FPGA VS ASIC COMPARISON TABLE (NUMATO LAB, 2020)

Jenni Donato states that the importance of sustainable development in design '*meets the needs of the present without compromising the ability of future generations to meet their own needs.*' (BPF, 2020) The use of resources to design and make electronic components are limited, finite and difficult to obtain. According to Malin Pettersson '*E-waste can be highly toxic and contains substances and material such as heavy metals and organics which affects human health and ecosystems negatively when they are not handled in a correct way.*' (Ctrl-Alt-Dev, 2018).

To effectively create a sustainable design process, the drive for sustainability needs to be implemented at the very beginning, to drive this program throughout the whole process and make sure it's actioned and maintained.

As a result, companies are now actively designing their products to ensure that materials can be extracted and recycled at end of life instead of being disposed at landfill.



FIGURE 3: IDEAL DESIGN SUSTAINABILITY CYCLE (BPF, 2020)

Designing a traffic light control system

The circuit design needs to consider 'hardware inputs' of the pedestrian 'wait' button and then using timers and sequencing, to display and change lights to control the flow of traffic to allow the waiting pedestrian to cross.

The circuit utilises 2 timers, a 5 Second and a 15 Second timer.

The 5 Second timer is there to create a 'wait' period for the pedestrian to cross and then create an output every 5 seconds, which will be used to make a timer to trigger the traffic lights to cycle from 'Green', through 'Amber' to 'Red'.

The 15 Second timer is there to create an output every 15 seconds and will be used to reset the 'Red' part of the traffic light sequence to return back to 'Green' through 'Red/Amber'.

The circuit can be displayed as a simplified finite state machine diagram as follows:

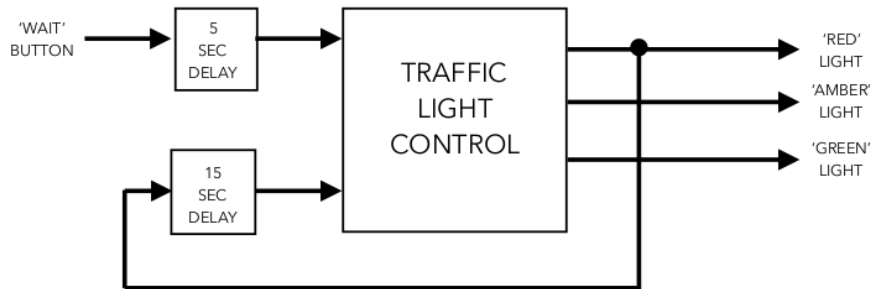


FIGURE 4: RAG CONTROLLER STATE MACHINE

The process now involves creating finite state machines for the 5 second timer, 15 second timer and the 'Traffic Light Control', which will be referred to as the 'RAG Controller'.

Creating the '5 Second' Timer:

This finite state machine will be designed to start counting through 6 states (S_0 to S_5), when the pedestrian 'wait' button (B) is pressed, advancing each state every second then create an output on ' S_5 ' then advance back to ' S_0 ' where it will wait for the next 'wait' button input.

5 Second Delay	
STATE	FUNCTION
S_0	RESET
S_1	1 SECOND
S_2	2 SECONDS
S_3	3 SECONDS
S_4	4 SECONDS
S_5	5 SECONDS

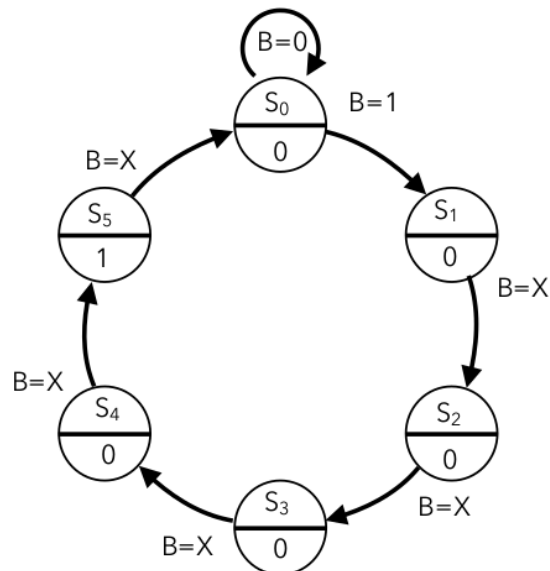


FIGURE 5: LOGIC TABLE & DIAGRAM OF A 5 SECOND DELAY

As 6 states are needed, to determine the amount of 'Flip Flops' we need, we can use the formula:
 $(\text{No of flip flops})^2 = \text{Number of States Available}$

Therefore:

$1^2 = 2 \text{ States Available} / 2^2 = 4 \text{ States Available} / 3^2 = 8 \text{ States Available} / 4^2 = 16 \text{ States Available}$

As we require 6 States (S_0 to S_5) then at least a 3 Flips flop configuration would be adequate.

Assignment - Finite State Machines

State Table:

Present	Next		Present			Next						Output
	B=0	B=1				B = 0			B = 1			
			Q _A	Q _B	Q _C	Q _A ⁺	Q _B ⁺	Q _C	Q _A ⁺	Q _B ⁺	Q _C ⁺	W
S ₀	S ₀	S ₁	0	0	0	0	0	0	0	0	1	0
S ₁	S ₂	S ₂	0	0	1	0	1	0	0	1	0	0
S ₂	S ₃	S ₃	0	1	0	0	1	1	0	1	1	0
S ₃	S ₄	S ₄	0	1	1	1	0	0	1	0	0	0
S ₄	S ₅	S ₅	1	0	0	1	0	0	1	0	1	0
S ₅	S ₆	S ₆	1	0	1	0	0	0	0	0	0	1
S ₆	X	X	1	1	0	X	X	X	X	X	X	X
S ₇	X	X	1	1	1	X	X	X	X	X	X	X

FIGURE 6: STATE TABLE OF THE 5 SECOND DELAY TIMER

Creating Karnaugh maps from the above table for each output (Q_A⁺, Q_B⁺, Q_C⁺, W)

Q_BQ_C

B Q _A \	00	01	11	10
00	0	0	1	0
01	1	0	X	X
11	1	0	X	X
10	0	0	1	0

$$Q_A^+ = Q_B Q_C + Q_A \overline{Q_C}$$

Q_BQ_C

Q _A \	00	01	11	10
0	0	0	0	0
1	0	1	X	X

$$W = Q_A Q_C$$

Q_BQ_C

B Q _A \	00	01	11	10
00	0	1	0	1
01	0	0	X	X
11	0	0	X	X
10	0	1	0	1

$$Q_B^+ = \overline{Q_A} \overline{Q_B} Q_C + Q_B \overline{Q_C}$$

Q_BQ_C

B Q _A \	00	01	11	10
00	0	0	0	1
01	1	0	X	1
11	1	0	X	1
10	1	0	0	1

$$Q_C^+ = Q_B \overline{Q_C} + Q_A \overline{Q_C} + B \overline{Q_C}$$

FIGURE 7: KARNAUGH MAPS OF THE 5 SECOND TIMER

From these tables and groupings, a logic gate diagram for the 5 Second Timer can now be drawn up:

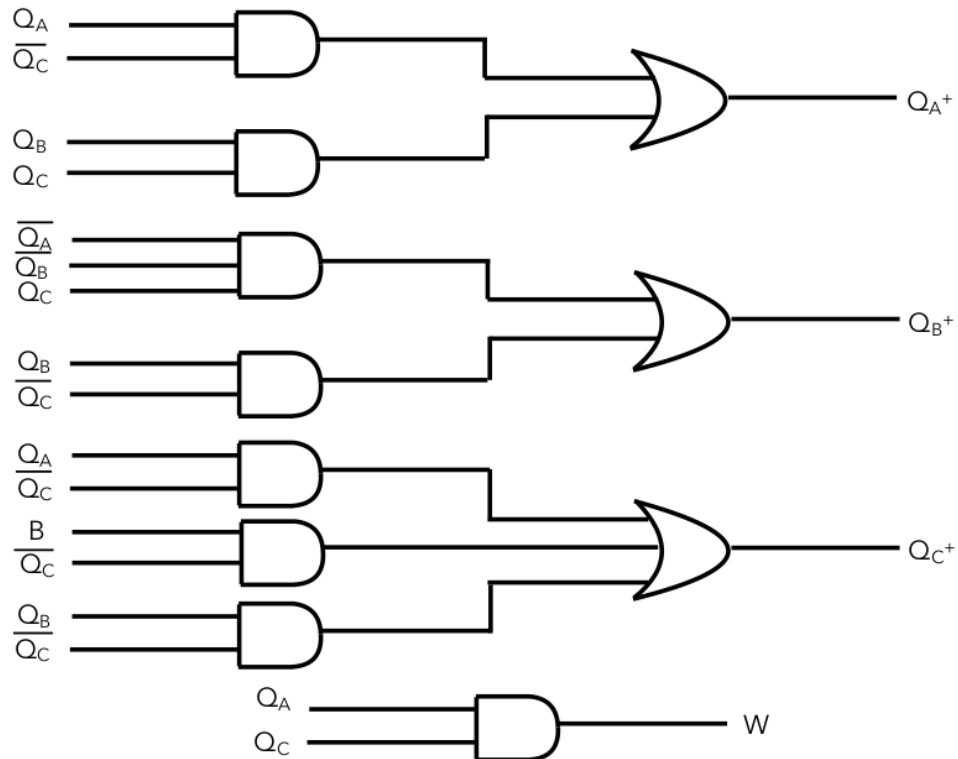


FIGURE 8: LOGIC GATE DIAGRAM FOR '5 SECOND TIMER'

This logic gate diagram can now be simulated in Multisim, together with the necessary JK Flip Flops to create the $\overline{Q_A}$, $\overline{Q_B}$, $\overline{Q_C}$, Q_A , Q_B , Q_C and W outputs required.

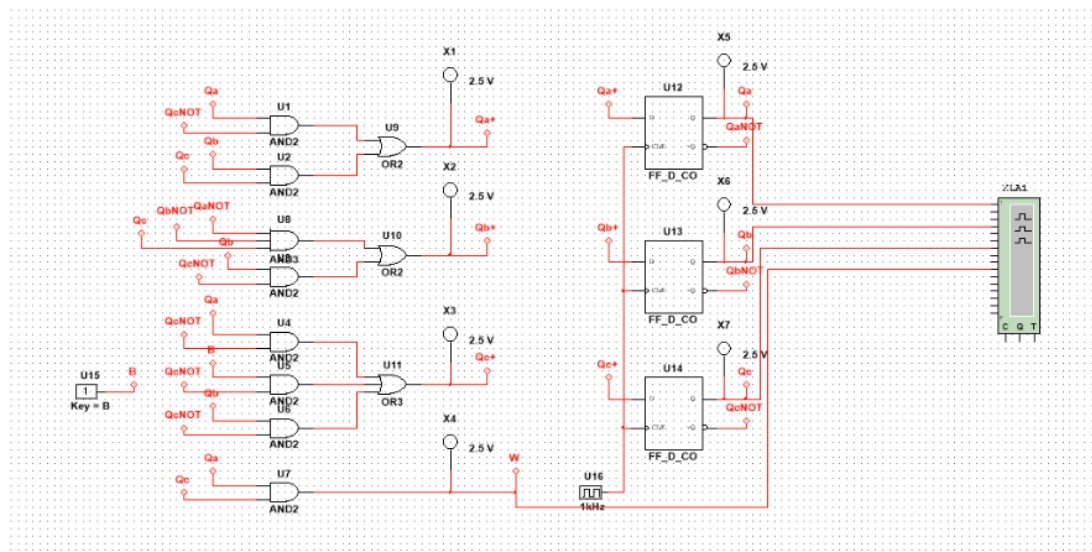


FIGURE 9: MULTISIM DIAGRAM FOR '5 SECOND TIMER'

Assignment - Finite State Machines

Simulating the circuit with test inputs, we can use a Logic Analyser to see the binary count from 0 to 5 with a resulting 'output' on Step '5'. (See figure below)

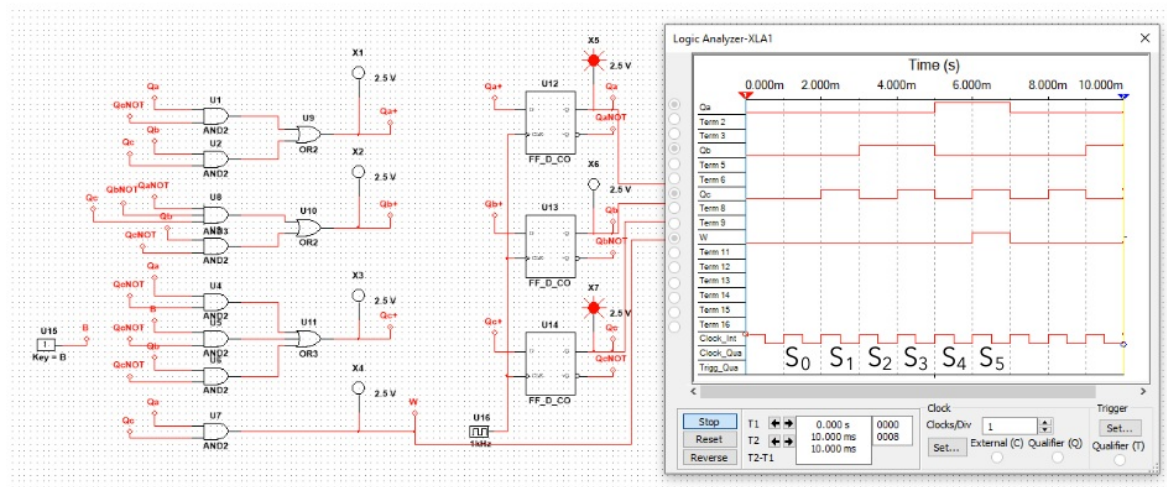


FIGURE 10: MULTISIM DIAGRAM FOR 5 SECOND TIMER WITH ANALYSER

All simulated inputs/outputs can now be converted to Nexus 4 DDR compatible input/output 'tags' and the program can be downloaded and tested, using onboard buttons as a simulation in place of a 'counter' to switch the Flips Flops ON/OFF and advance the steps.

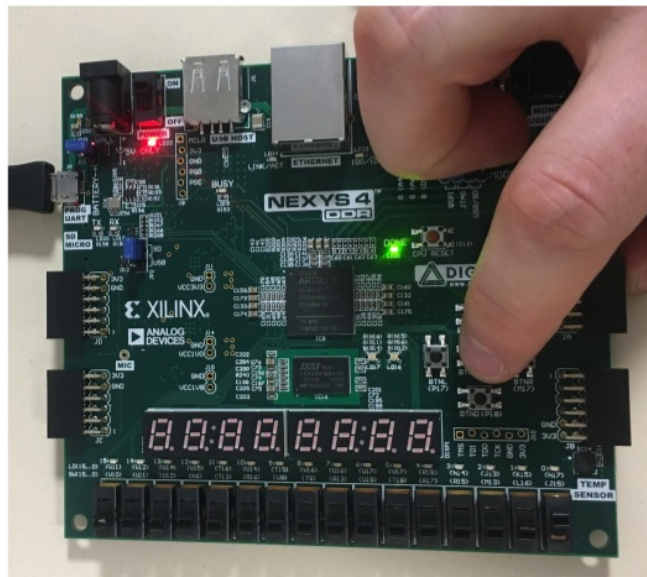


FIGURE 11: NEXYS DDR 4 BOARD TESTING 5 SECOND DELAY TIMER

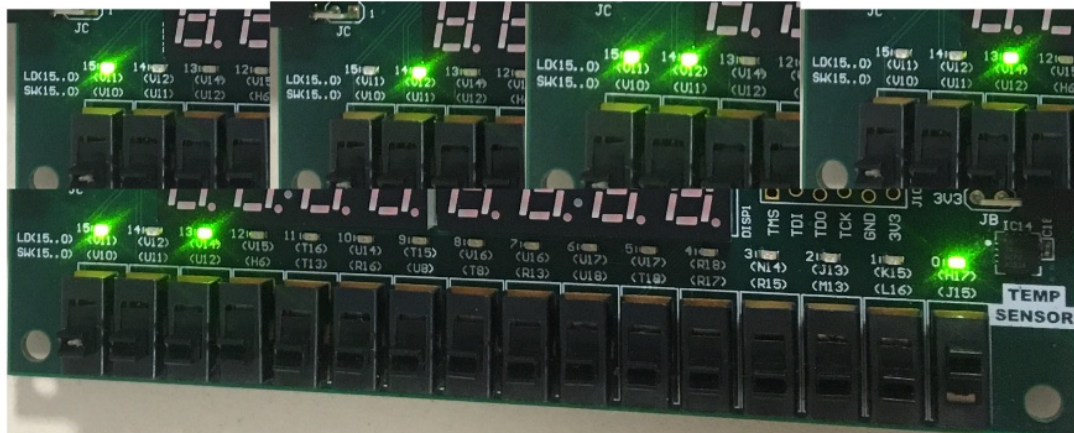


FIGURE 12: NEXYS DDR 4 BOARD WITH 5 SECOND DELAY TIMER SHOWING S1 TO S5 COUNTS WITH 'W' OUTPUT

The use of virtual testing became very useful tool in diagnosing any issues with the binary count of the sequence and testing of the 'clock' gave a successful output at 'Step 5' ONLY.

Physical testing of the circuit showed that despite the clock input was pulsed to simulate a 1Hz pulse, the circuit binary count stayed on 'S₀' until 'SW0' (mapped as the 'wait' button) was turned on. Although the choice to use a push button as a 'pulsed input' for a clock was initially a good one, we did have issues as glitches in the button would cause the sequence to advance 2 or even 3 three times on individual pushes. The use of a flip switch would have improved this part of testing but overall, the circuit performed well.

Creating the '15 Second' Timer:

This part of the finite state machine will be designed to start counting through 16 states (S_0 to S_{15}), when the RAG Controller 'Red' signal (R) is inputted, advancing each state every second then create an output on ' S_{15} ' then advance back to ' S_0 ' where it will wait for the next 'Red' RAG controller input. A table showing each state and its function, together with a state diagram to show how the 'R' input will affect the sequence.

15 Second Delay	
STATE	FUNCTION
S_0	RESET
S_1	1 SECOND
S_2	2 SECONDS
S_3	3 SECONDS
S_4	4 SECONDS
S_5	5 SECONDS
S_6	6 SECONDS
S_7	7 SECONDS
S_8	8 SECONDS
S_9	9 SECONDS
S_{10}	10 SECONDS
S_{11}	11 SECONDS
S_{12}	12 SECONDS
S_{13}	13 SECONDS
S_{14}	14 SECONDS
S_{15}	15 SECONDS

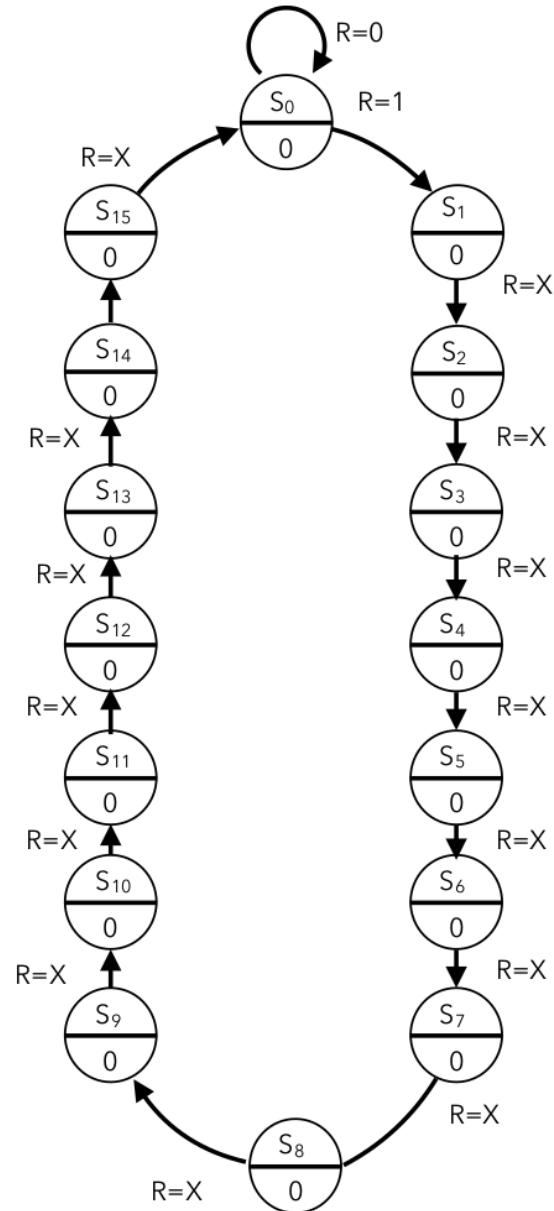


FIGURE 13: LOGIC TABLE & STATE DIAGRAM FOR 15 SECOND TIMER

Using this state diagram, we can now develop a state table and resulting Karnaugh Maps.

As 15 states are needed, to determine the amount of 'Flip Flops' needed, we can use the formula:

$$(\text{No of flip flops})^2 = \text{Number of States Available}$$

Therefore:

Assignment - Finite State Machines

$1^2 = 2$ States Available / $2^2 = 4$ States Available / $3^2 = 8$ States Available / $4^2 = 16$ States Available

As we require 15 States (S_0 to S_{15}) then a 4 'Flip flop' configuration is required.

State Table For 15 Second Timer:

Present	Next	Present				Next				Output
		Q_A	Q_B	Q_C	Q_D	Q_A^+	Q_B^+	Q_C^+	Q_D^+	
S_0	S_1	0	0	0	0	0	0	0	1	0
S_1	S_2	0	0	0	1	0	0	1	0	0
S_2	S_3	0	0	1	0	0	0	1	1	0
S_3	S_4	0	0	1	1	0	1	0	0	0
S_4	S_5	0	1	0	0	0	1	0	1	0
S_5	S_6	0	1	0	1	0	1	1	0	0
S_6	S_7	0	1	1	0	0	1	1	1	0
S_7	S_8	0	1	1	1	1	0	0	0	0
S_8	S_9	1	0	0	0	1	0	0	1	0
S_9	S_{10}	1	0	0	1	1	0	1	0	0
S_{10}	S_{11}	1	0	1	0	1	0	1	1	0
S_{11}	S_{12}	1	0	1	1	1	1	0	0	0
S_{12}	S_{13}	1	1	0	0	1	1	0	1	0
S_{13}	S_{14}	1	1	0	1	1	1	1	0	0
S_{14}	S_{15}	1	1	1	0	1	1	1	1	0
S_{15}	S_0	1	1	1	1	0	0	0	0	1

FIGURE 14: STATE TABLE FOR 15 SECOND TIMER

Now Karnaugh maps can be created from the above table for each output (Q_A^+ , Q_B^+ , Q_C^+ , Q_D^+ , W)

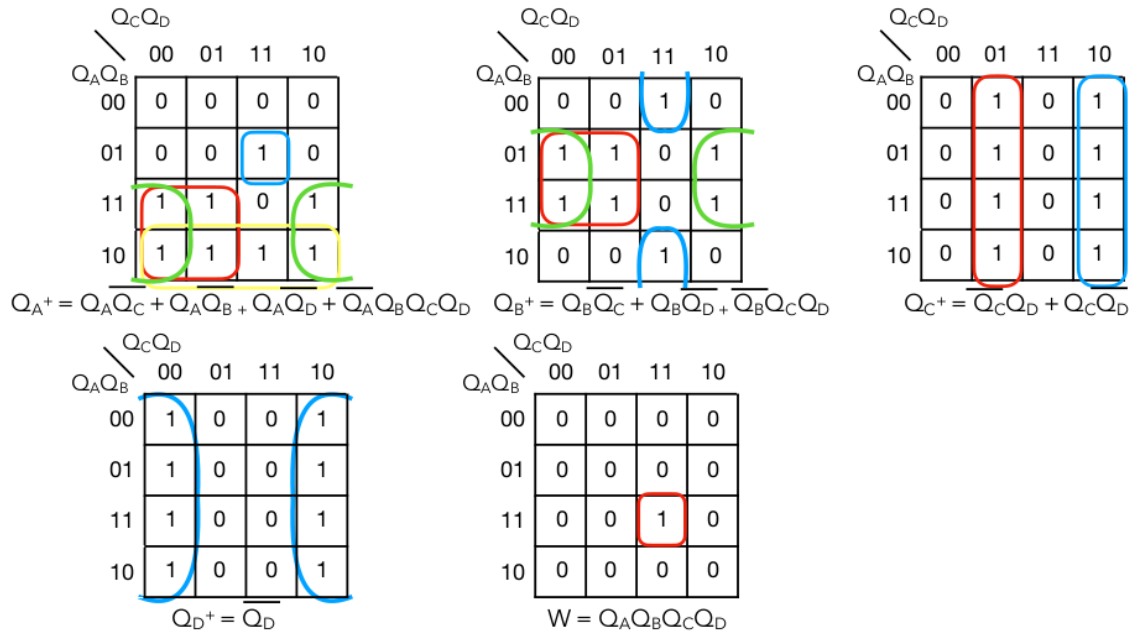


FIGURE 15: KARNAUGH MAPS OF THE 15 SECOND TIMER

From these tables and applicable groupings, a logic gate diagram for the 15 Second Timer can be drawn up:

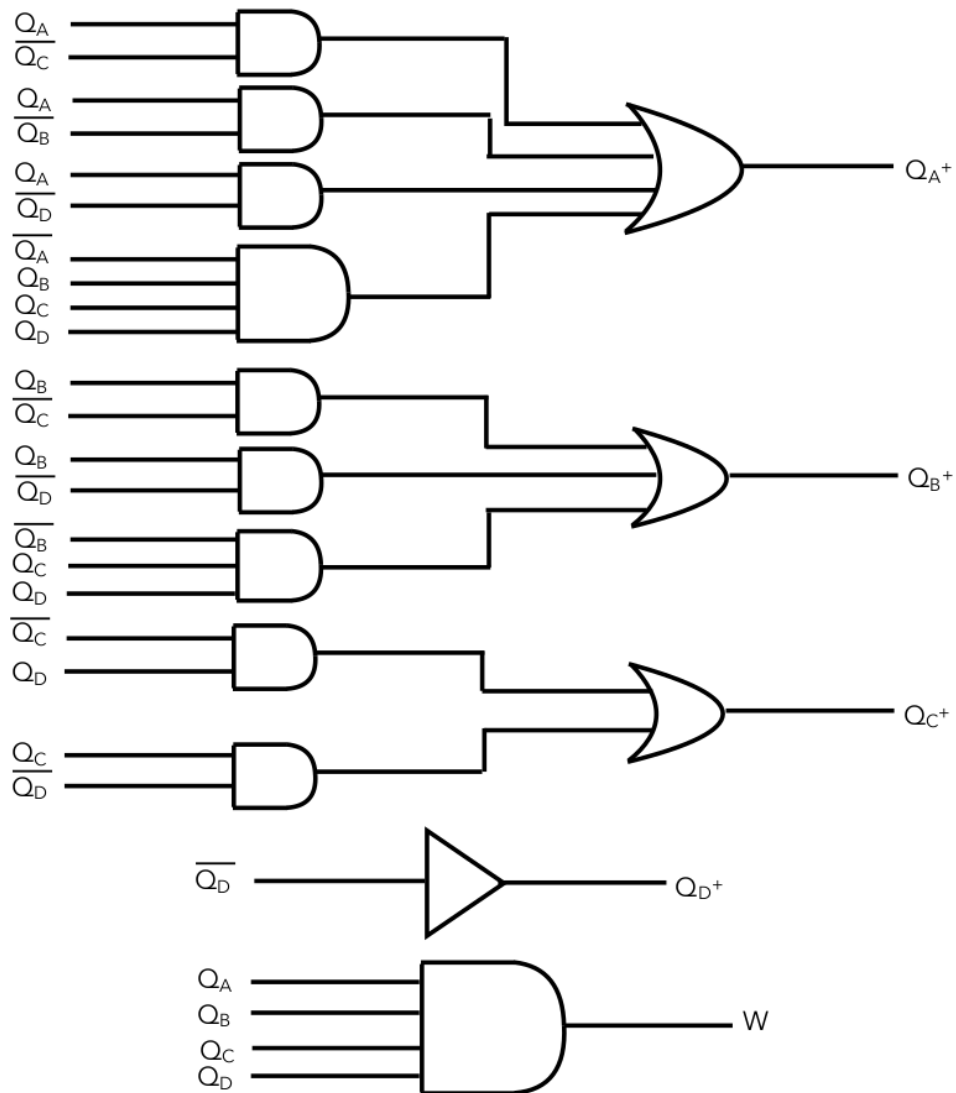


FIGURE 16: LOGIC GATE DIAGRAM FOR 15 SECOND TIMER

Assignment - Finite State Machines

This logic gate diagram can now be simulated in Multisim, together with the necessary JK Flip Flops to create the $\overline{Q}_A$, $\overline{Q}_B$, $\overline{Q}_C$, $\overline{Q}_D$ and Q_A , Q_B , Q_C , Q_D outputs required.

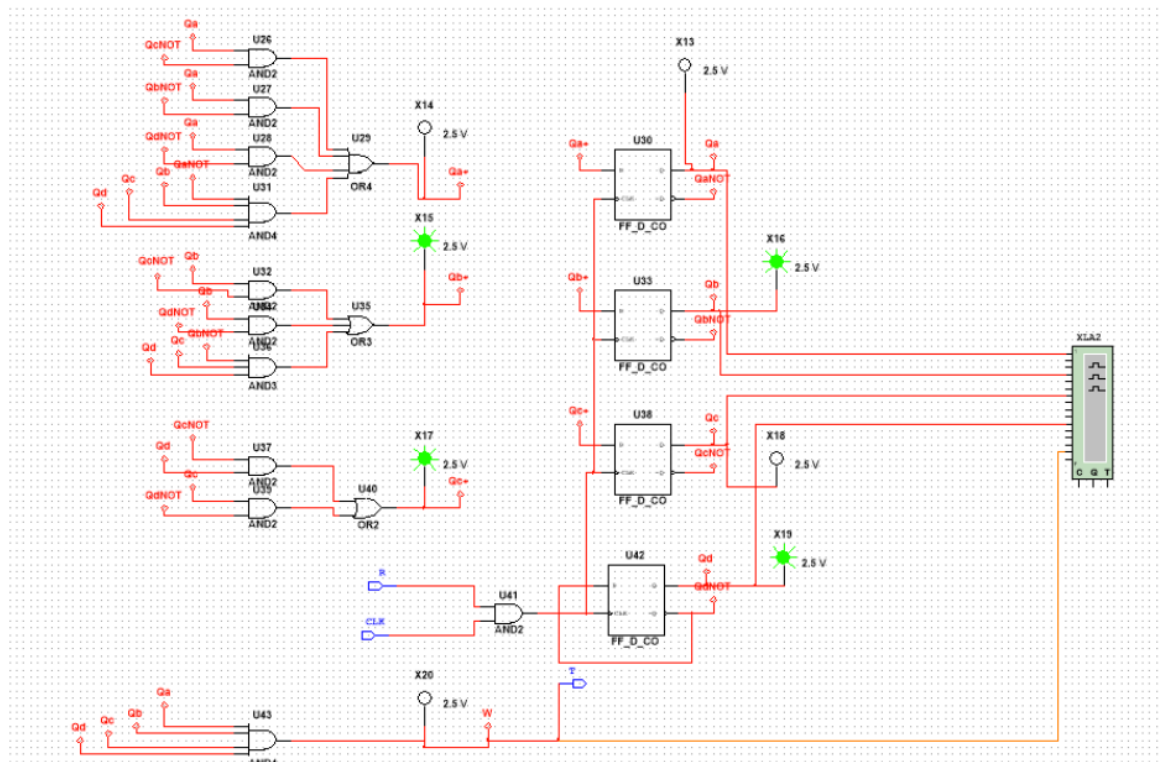


FIGURE 17: MULTISIM DIAGRAM FOR 15 SECOND TIMER

Simulating the circuit with test inputs, we can use a Logic Analyser to see the binary count from 0 to 15 with the resulting output (W) only showing on Step '15'. (See figure below)

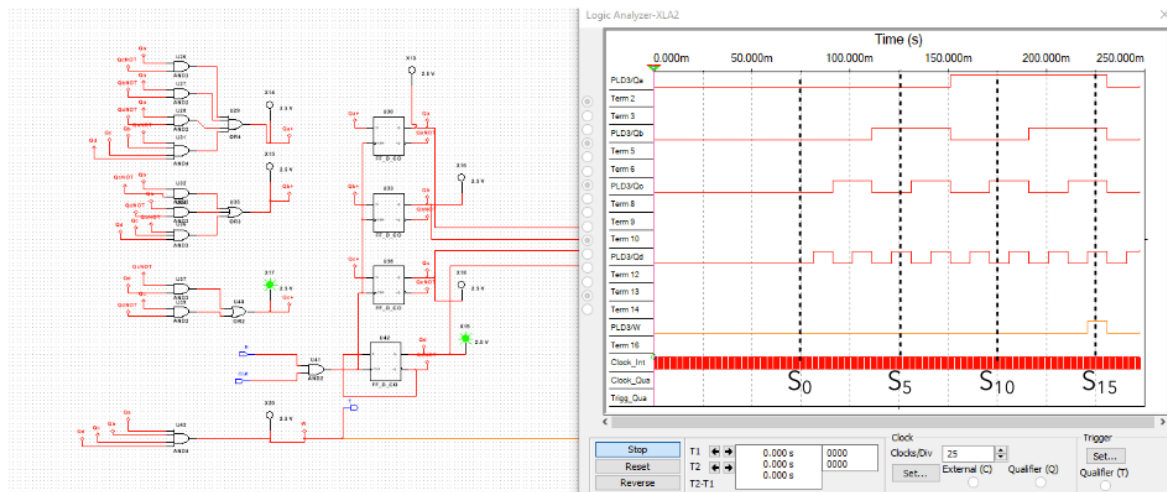


FIGURE 18: MULTISIM DIAGRAM FOR 15 SECOND TIMER WITH ANALYSER

Assignment - Finite State Machines

All simulated inputs/outputs on the Multisim program can now be converted to Nexus 4 DDR compatible input/output 'tags' and the program can be downloaded and tested to the board, using onboard push buttons as a simulation in place of a 'clock' to switch the Flips Flops ON/OFF and advance the steps.

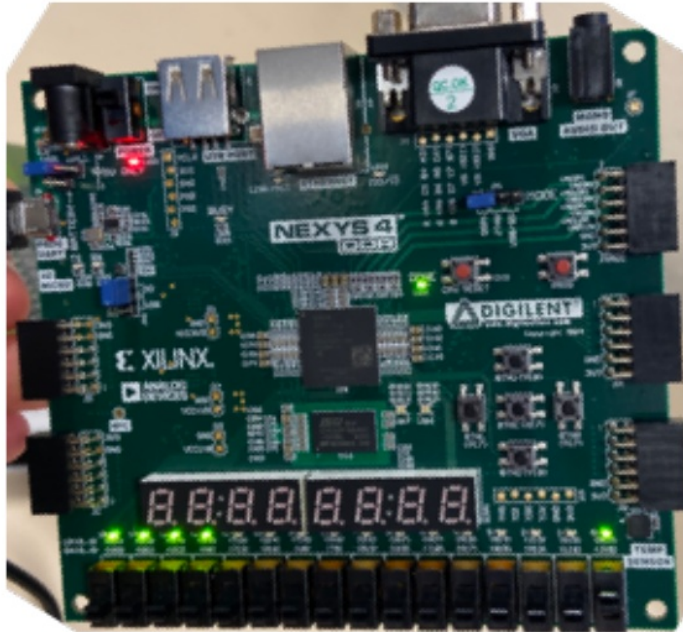


FIGURE 19: NEXYS DDR 4 BOARD WITH 15 SECOND DELAY TIMER SHOWING 'S15' COUNT WITH 'W' OUTPUT

Testing of the circuit gave a successful output at 'Step 15'. The nexus board provided a lot of extra LEDs meaning we could program LED0 to LED3 to show the binary layout of the step sequence (Q_A to Q_D) as a quick way to debug/monitor the step sequence and see if the output only showed at Step 15. As the nexus board does not provide real time monitoring back to Multisim, this was the only way of displaying the current status of the sequence.

Creating the 'R.A.G Controller':

This finite state machine will be designed to start counting through 3 states (S_0 to S_3), when the '5 Second Timer 'output' signal (R) is inputted, advancing each state every second and stop at ' S_3 '. It will then wait for the 15 Second Timer Output (T) to continue onto ' S_4 ' then advance to ' S_5 ' and create the output, where it will continue to advance back to ' S_0 ' where it will wait for the next 'Red' RAG controller input.

RAG Controller	
STATE	FUNCTION
S_0	GREEN
S_1	AMBER
S_2	AMBER
S_3	RED
S_4	RED/AMBER
S_5	RED/AMBER

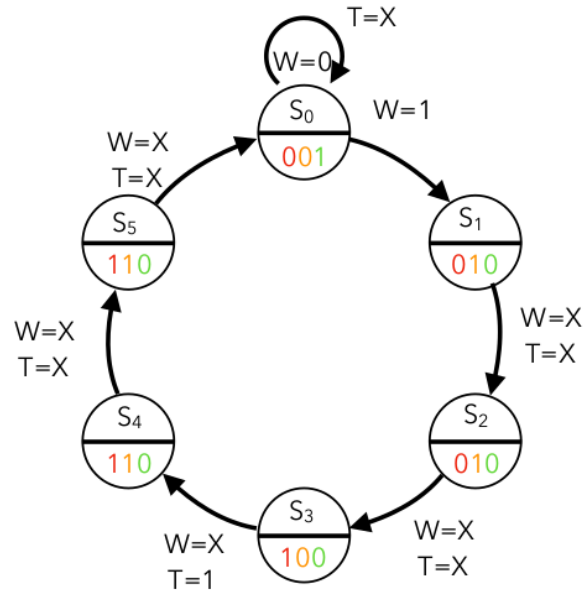


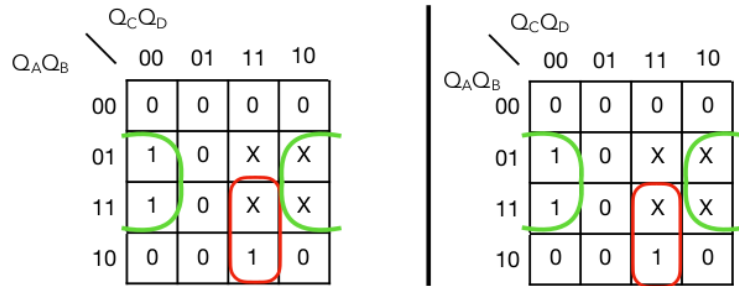
FIGURE 20: LOGIC TABLE AND DIAGRAM FOR THE 'R.A.G CONTROLLER'

Current	Next				Current			Next												OUTPUT			
								W=0						W=1									
	W=0		W=1		T = 0			T = 1			T = 0			T = 1									
	T=0	T=1	T=0	T=1	Q _A	Q _B	Q _C	Q _A +	Q _B +	Q _C +	Q _A +	Q _B +	Q _C +	Q _A +	Q _B +	Q _C +	Q _A +	Q _B +	Q _C +	R	A	G	
S ₀	S ₀	S ₀	S ₁	S ₁	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1
S ₁	S ₂	S ₂	S ₂	S ₂	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0
S ₂	S ₃	S ₃	S ₃	S ₃	0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0
S ₃	S ₄	S ₄	S ₄	S ₄	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0
S ₄	S ₅	S ₅	S ₅	S ₅	1	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0
S ₅	S ₀	S ₀	S ₀	S ₀	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X
S ₆	X	X	X	X	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
S ₇	X	X	X	X	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

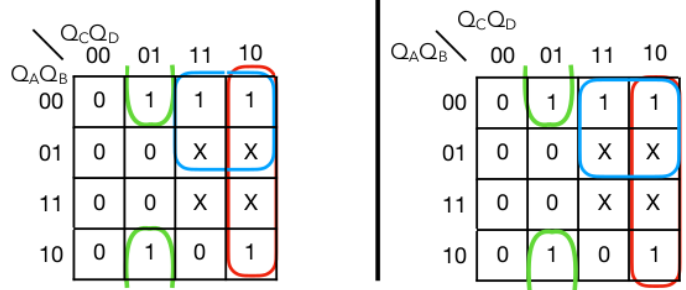
FIGURE 21: STATE TABLE OF THE 'R.A.G CONTROLLER'

The groupings of the '0's and '1's from the above table form the Karnaugh maps, as shown below.

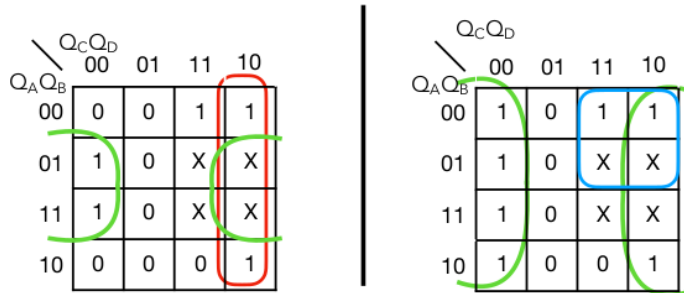
Assignment - Finite State Machines



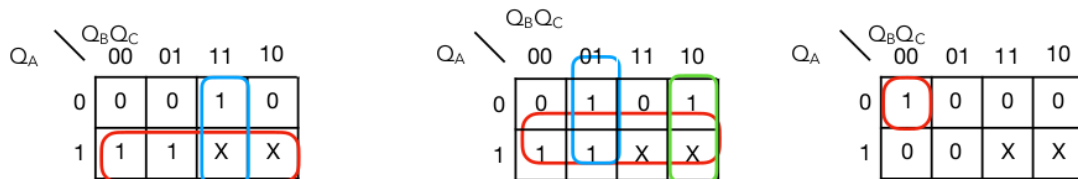
$$Q_A^+ = Q_A \overline{Q_C} + T Q_B Q_C$$



$$Q_B^+ = \overline{T} Q_B + Q_B \overline{Q_C} + \overline{Q_A} \overline{Q_B} Q_C$$



$$Q_C^+ = Q_A \overline{Q_C} + \overline{T} Q_B + Q_B \overline{Q_C} + W \overline{Q_C}$$



$$R = Q_A + Q_B Q_C$$

$$A = Q_A + \overline{Q_B} Q_C + Q_B \overline{Q_C}$$

$$G = \overline{Q_A} \overline{Q_B} \overline{Q_C}$$

FIGURE 22: KARNAUGH MAPS OF THE 'R.A.G CONTROLLER'

Using the groupings formed from the karnaugh maps, a logic gate diagram can now be derived:

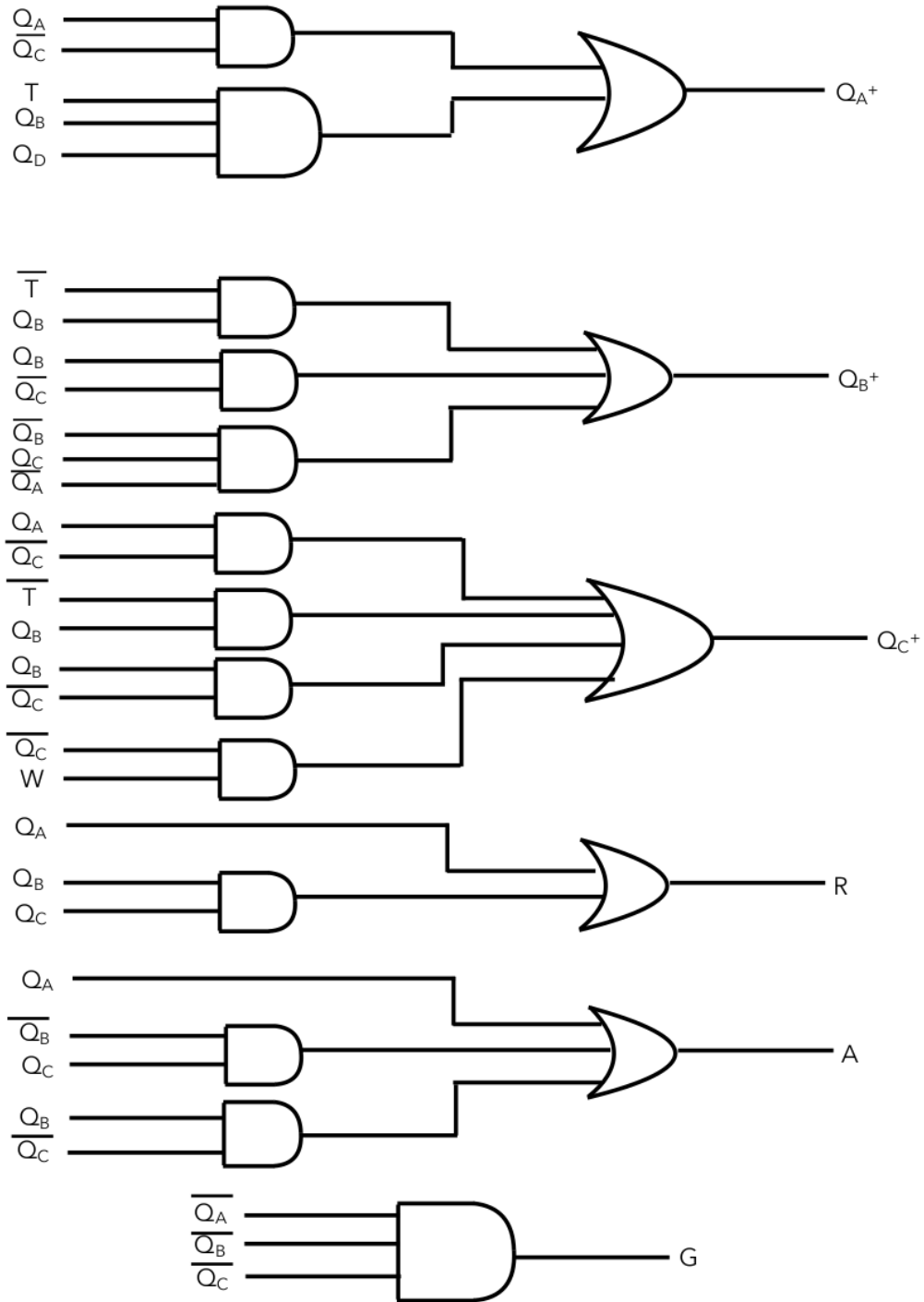


FIGURE 23: LOGIC GATE DIAGRAM FOR THE 'R.A.G CONTROLLER'

With the logic gate diagram now formed, the circuit can now be simulated in Multisim, together with the necessary JK Flip Flops to create the ' $\overline{Q_A}$, $\overline{Q_B}$, $\overline{Q_C}$, Q_A , Q_B , Q_C , R , A and G ' outputs required.

Assignment - Finite State Machines

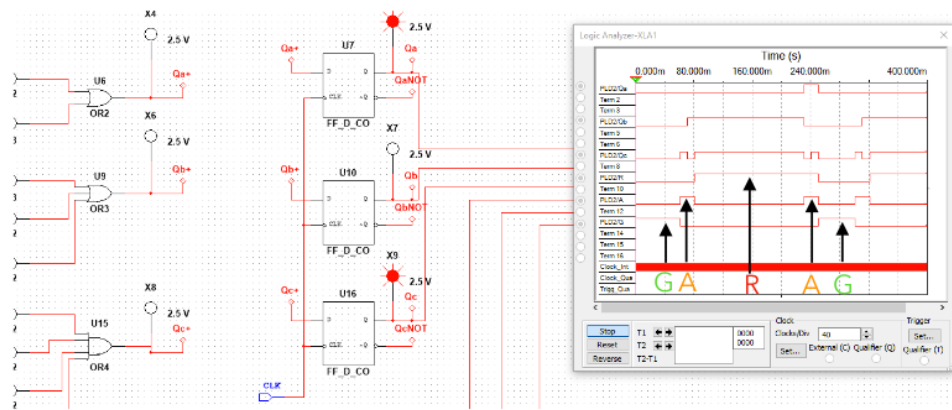


FIGURE 24: MULTISIM CIRCUIT OF THE 'R.A.G. CONTROLLER'

Using the '5 second timer', '15 second timer' and 'RAG controller' circuits, the circuits were tied together using a PLD design and then virtually tested to ensure the 'R,A,G sequence' follows with correct timings and delays. (See figure above for proven R.A.G sequence)

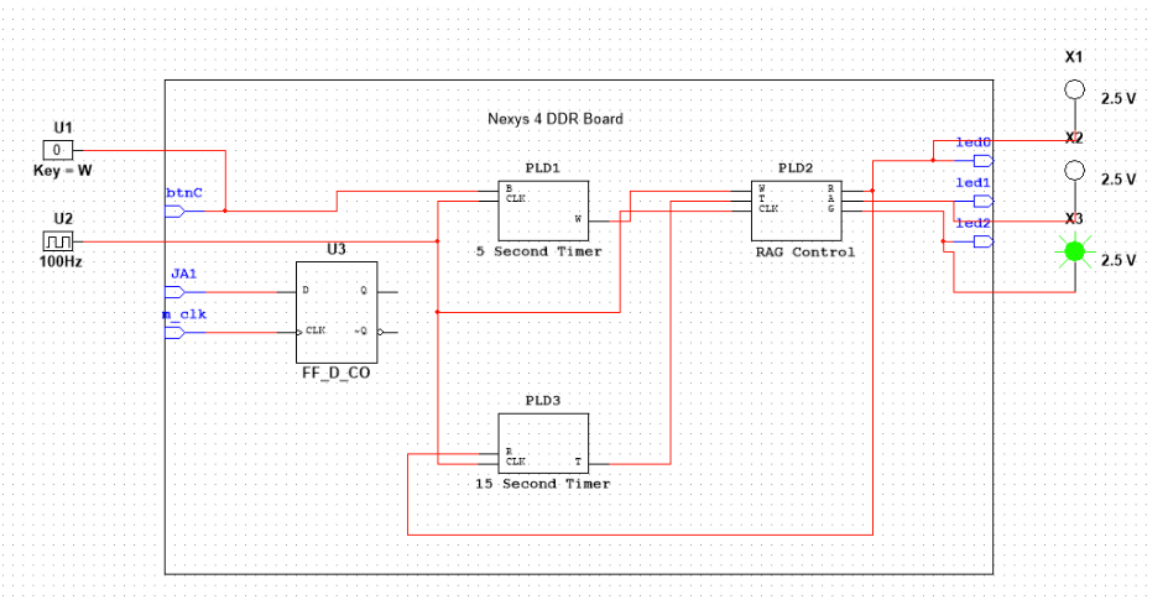


FIGURE 25: PLD CIRCUIT OF THE 'R.A.G. CONTROLLER'

Using the finalised circuit, it was tested virtually on Multisim with a simulated clock to ensure the RAG lights were operating in the right sequence and observed the programmed delays before converting the clock input to an external connection and then using an external pulse generator to generate the 1Hz signal needed to advance the various state machines in the 5 second, 15 second and RAG controller logic circuits by 1 second increments.

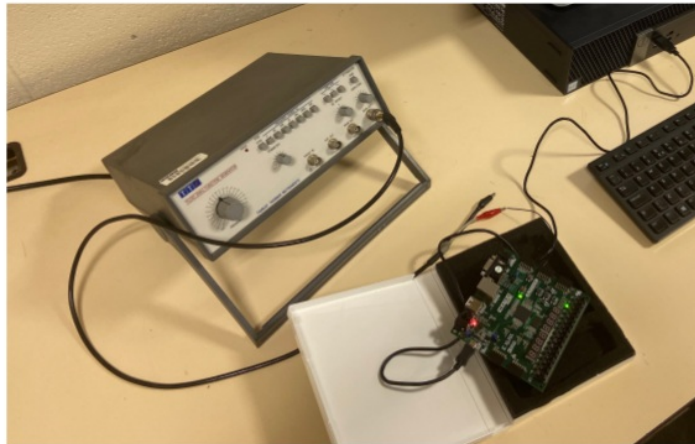


FIGURE 26: REAL WORLD TESTING OF THE 'R.A.G CONTROLLER'

Using the Nexys DDR 4 FPGA circuit board, the whole PLD design circuit was downloaded from Multisim with now only 2 external inputs (Wait button and external clock input) and 3 external outputs (R,A and G lights) programmed.

Testing of this circuit gave a successful outcome to the original brief, creating a sequence of Red, Amber and Green lights that sequenced through at the desired layout and with correct timings, especially when using the external clock to provide a smooth and timed sequence.

Recommendations:

As the initial 5 second 'wait' timer waits for the initial input of the Wait button (W) and then counts 5 steps, outputting a signal of the 5th step, the state diagram was modified so the counter counts as normal, but confirms a positive signal of the pedestrian pressure pad (P) at the 4th step before advancing to 'S₅', and resulting output of '1'.

This method was chosen as the confirmation of the pressure pad will only be required at 'S₄'. If there are any glitches in the input from the pressure pad due to steps or issues with the sensor, this will not result in annoying resets of the counter throughout the timing sequence.

5 Second Delay (Modified for Pressure Pad)	
STATE	FUNCTION
S ₀	RESET
S ₁	1 SECOND
S ₂	2 SECONDS
S ₃	3 SECONDS
S ₄	4 SECONDS
S ₅	5 SECONDS

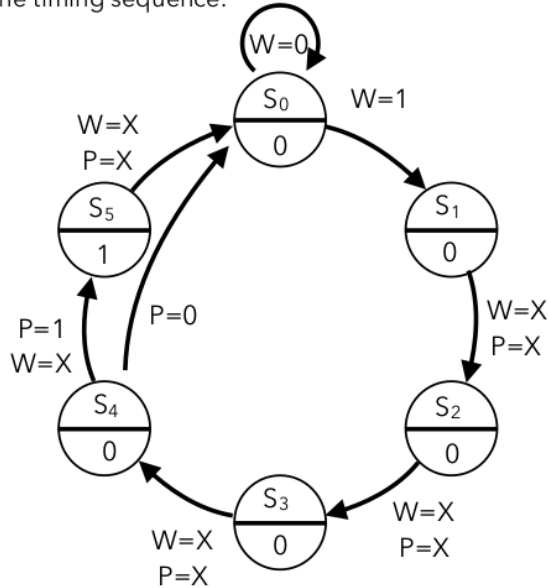


FIGURE 27: LOGIC TABLE & DIAGRAM OF THE 5 SECOND DELAY TIMER (MODIFIED)

Using the state diagram, a modified state table can now be drawn up and Karnaugh Maps can be extracted.

Current	Next				Current			Next												Output
								W=0						W=1						
	W=0		W=1					T = 0			T = 1			T = 0			T = 1			W
	T=0	T=1	T=0	T=1	Q _A	Q _B	Q _C	Q _A ⁺	Q _B ⁺	Q _C ⁺	Q _A ⁺	Q _B ⁺	Q _C ⁺	Q _A ⁺	Q _B ⁺	Q _C ⁺	Q _A ⁺	Q _B ⁺	Q _C ⁺	
S ₀	S ₀	S ₀	S ₁	S ₁	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
S ₁	S ₂	S ₂	S ₂	S ₂	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0
S ₂	S ₃	S ₃	S ₃	S ₃	0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0
S ₃	S ₄	S ₄	S ₄	S ₄	0	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0
S ₄	S ₅	S ₅	S ₅	S ₅	1	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0
S ₅	S ₀	S ₀	S ₀	S ₀	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
S ₆	X	X	X	X	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X
S ₇	X	X	X	X	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X

FIGURE 28: STATE TABLE OF THE 5 SECOND DELAY TIMER (MODIFIED)

Assignment - Finite State Machines

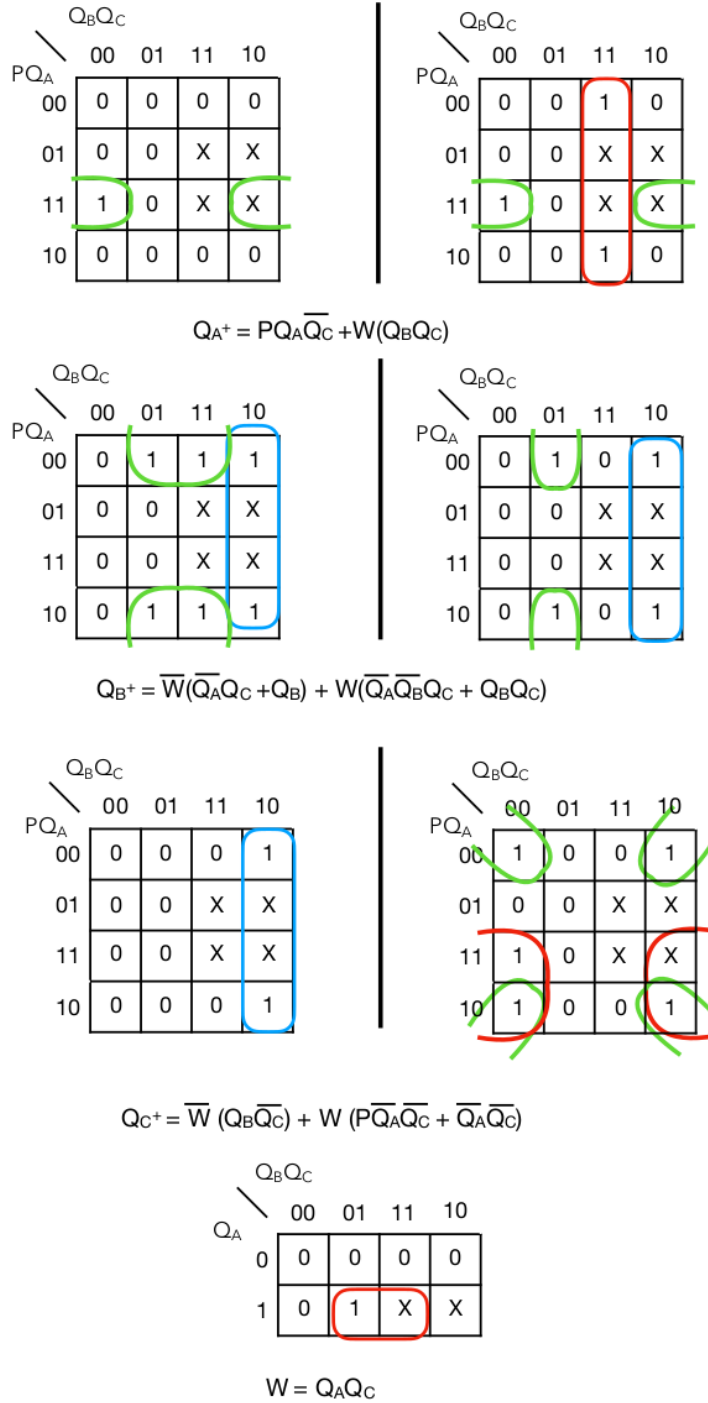


FIGURE 29: KARNAUGH MAPS OF THE MODIFIED '5 SECOND TIMER'

In comparison to the original '5 Second timer' design, this logic circuit now has 2 inputs and only makes a 'decision' at 'S₄', whether to move to 'S₅' or reset back to 'S₀', making the sequence more easy to understand.

Further recommendations could include a slight alteration to the PLD design of the layout, where a 2nd '5 second' timer could be utilised a second time and the pedestrian 'Wait' button would extinguish 5 seconds after the Red light illuminates, together with an audible beacon. This would ensure that the traffic has an opportunity to stop when the Red light is showing, before allowing the pedestrians to cross, thus improving safety and overall satisfaction that the system is doing it's job.

The 15 second counter would then allow 10 seconds for the pedestrians to cross before returning back through the Red/Amber - Green sequence, which I think is sufficient for pedestrians to cross.

To aid accessibility for visual impaired, a knurled rotating knob is positioned at the bottom of both 'wait' button stations, which rotate when the wait light extinguishes and an audible beacon starts, to alert people that its ready to cross.

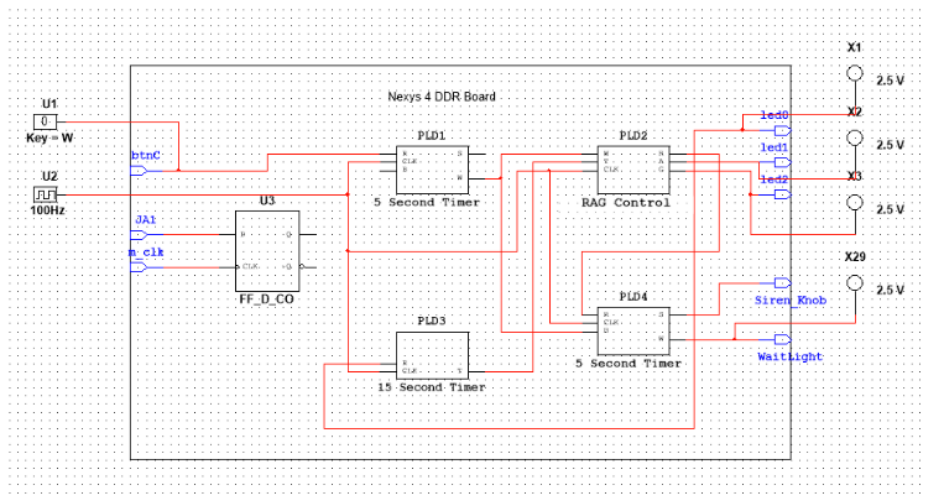


FIGURE 30: PLD DESIGN OF THE MODIFIED 'RAG CONTROLLER'

Conclusion:

The circuit and resulting logic tables/karnaugh maps created outputs which eventually designed a circuit that fulfilled the original design spec where the sequence of the traffic Lights (Red, Amber and Green) followed the designed sequence and with the proposed 'delays' achieved.

The Multisim program provided an easy to design and follow layout which allows for easy testing using 'internal' lamps as a visual aid to what outputs are HIGH or LOW.

Using the Research, the circuits that were designed are classified as Sequential Synchronous circuits, as there is internal feedback (Flip Flops) and external reliance on a clock to advance the sequence with external pulses from this clock.

As the circuit/system is now tested and working, the circuit could now be designed as an ASIC with marked PCB inputs, making installation easier to follow and less prone to installer error. Using the earlier Figure in Research, the cost drivers for this type of circuit would be:

- Reliability - as it needs to run 24/7 remotely)
- The need for a quick replacement - if there are any potential issues in the field
- Standardised operation - timing needs to be agreed and kept to without possibility of change by the installer.

As a result, the circuit would be preferred as an **ASIC**.

FPGA relies upon a user programmable memory which can be corrupted and damaged, internal memory can be lost through power loss and this circuit would be favoured as an ASIC.

Reference List:

(Electronics Tutorials, 2020) [online] Available at: <https://www.electronics-tutorials.ws/combo/combo_1.html> [Accessed 18th January 2020]

(University Of Surrey, 2020) [online] Available at: <http://www.ee.surrey.ac.uk/Projects/Labview/Sequential/Course/03-Seq_Intro/Intro.html> [Accessed 18th January 2020]

(ETH Zurich, 2017) [online] Available at: <https://ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/system-security-group-dam/education/Digitaltechnik_17/0b_SequentialCircuits.pdf> [Accessed 18th January 2020]

(El-Pro-Cus, 2020) [online] Available at: <<https://www.elprocus.com/introduction-to-combinational-logic-circuits/>> [Accessed 18th January 2020]

(Embedded, 2009) Author: Ted Carmely [online] Available at: <<https://www.embedded.com/using-finite-state-machines-to-design-software/>> [Accessed 20th January 2020]

(ITProPortal, 2020) [online] Available at: <<https://www.itproportal.com/features/finite-state-machines-better-than-flowcharts/>> [Accessed 20th January 2020]

(Numato Lab, 2020) [online] Available at: <<https://numato.com/blog/differences-between-fpga-and-asics/>> [Accessed 21st January 2020]

(Ctrl-Alt-Dev, 2018) [online] Available at: <<http://wpmu.mah.se/nmict182group1/e-waste-a-threat-to-sustainable-development/>> [Accessed 21st January 2020]

(BPF, 2020) [online] Available at: <https://bpf.co.uk/sustainable_manufacturing/design/Designing_Sustainability.aspx> [Accessed 21st January 2020]

1809809- Finite State Machines Final

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

91 / 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

GRADING FORM: CSG-RUBIC

RHYS DAVIES

91

RESEARCH-INFORMED LITERATURE



There was evidence of considerable further reading and research from highly credible and reliable sources. Your interpretation and understanding of the additional research was exceptional and demonstrated a deep underlying appreciation of the subject. Your references were consistently applied to a high level of competency and your reference list was extensive for this assignment.

9

KNOWLEDGE AND UNDERSTANDING



The knowledge and understanding of the subject demonstrated by this assignment was generally

27

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.

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 exceptional. You used both the basic and advanced features of the resources appropriately when designing the circuit and you used appropriate 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

18

assistance thus demonstrating some advanced problem solving skills. Your results were comprehensive and allowed for a detailed conclusion to be attempted.

SKILLS FOR PROFESSIONAL PRAC.



In this assignment you have demonstrated skills in working individually and also collaboratively. Your written report was exceptionally well structured and written in a highly 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.

10