

Digital Electronics

Aaron Griffiths

Contents

Information table : Page 2

Task 1 : Pages 3 - 10

Task 2 : Pages 11 - 17

Task 3 : Pages 18 - 22

Task 4 : Pages 23 - 27

Task 5 : Pages 28 - 32

Task 6 : Pages 33-34

References : Page 35

Boolean rules for reference

Laws of Boolean Algebra				
	AND version	OR version		
Complement law	$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$		
Identity law	$A \cdot 1 = A$	$A + 0 = A$		
Distributive law	$AB + AC = A(B + C)$	$A + B \cdot C = (A + B)(A + C)$		
Absorption law	$A(A + B) = A$	$A + AB = A$		
Double inversion law	$\overline{\bar{A}} = A$			
Annulment	$A \cdot 0 = 0$	$A + 1 = 1$		
Idempotent	$A \cdot A = A$	$A + A = A$		
Commutative law	$A \cdot B = B \cdot A$	$A + B = B + A$		
De morgan's theorems	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$		
Truth Tables for common gates				
AND	A	B	Q	
	0	0	0	
	1	0	0	
	0	1	0	
	1	1	1	
OR	A	B	Q	
	0	0	0	
	1	0	1	
	0	1	1	
	1	1	1	
NAND	A	B	Q	
	0	0	1	
	1	0	1	
	0	1	1	
	1	1	0	
NOR	A	B	Q	
	0	0	1	
	1	0	0	
	0	1	0	
	1	1	0	
XOR	A	B	Q	
	0	0	0	
	1	0	1	
	0	1	1	
	1	1	0	
NOT	A	Q		
	1	0		
	0	1		

Table 1 Boolean Laws

Task1

I have produced a table using sources gathered in class and my own research and I will refer to this to demonstrate which laws I am using when simplifying each expression. Each and any boolean expression is produced using either AND, OR and NOT gates.

“OR, AND and NOT gates are the three basic logic gates as they together can be used to construct the logic circuit for any given Boolean expression.” (106 Anil Kumar Maini, 2008, Digital electronics and logic design)

Commented [RB1]: Evidence of reading.

Reasons for simplification?

- Simplified circuits are easier to manipulate and input in to a design
- Financial savings by using fewer components/Memory
- Ease of understanding

De Morgans theorem allows us to convert a simplified Boolean expression to a NAND gate only circuit, NAND and NOR gates are universal gates as they have the capability of performing the functions of any of the basic logic gates.

“This is so because a combination of NAND gates or a combination of NOR gates can be used to perform functions of any of the basic logic gates. It is for this reason that NAND and NOR gates are universal gates.”

(106 Anil Kumar Maini, 2008, Digital electronics and logic design)

Why use a Universal gate?

It is far more economical to use a universal gate as the manufacturing process will be considerably more efficient.

I will demonstrate the logic circuits using the Mitsubishi Alpha functional block diagram software

Task1(i)

$$Q = ABC + \bar{A}\bar{B}C + AB\bar{C}$$

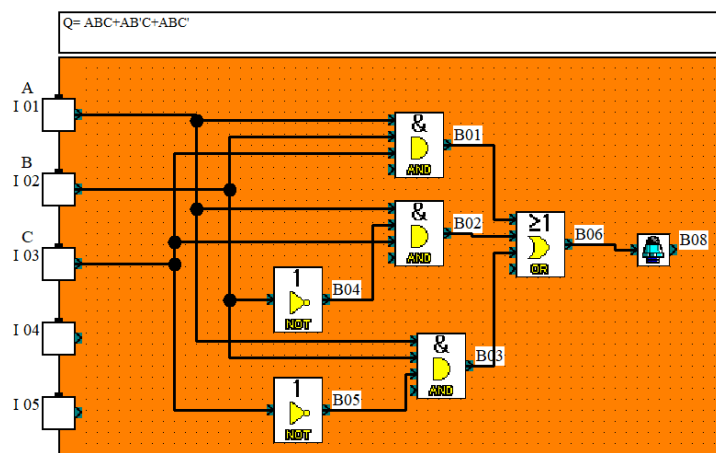


Figure 1 (i) Un-simplified expression

$$Q = ABC + \bar{A}\bar{B}C + AB\bar{C}$$

$$Q = AB(C + \bar{C}) + \bar{A}\bar{B}C \text{ (AND Distributive law)}$$

$$Q = AB \cdot 1 + \bar{A}\bar{B}C \text{ (AND Identity law)}$$

$$Q = AB + \bar{A}\bar{B}C$$

$$Q = A(B + \bar{B}C) \text{ (AND Distributive law)}$$

$$Q = A(B + C) \text{ (OR Complement law)}$$

$$Q = AB + AC \text{ Expanded brackets for final expression}$$



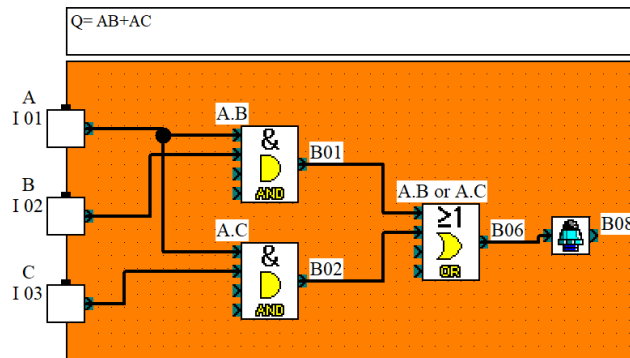


Figure 2 (i) Simplified expression

NAND gate Conversion

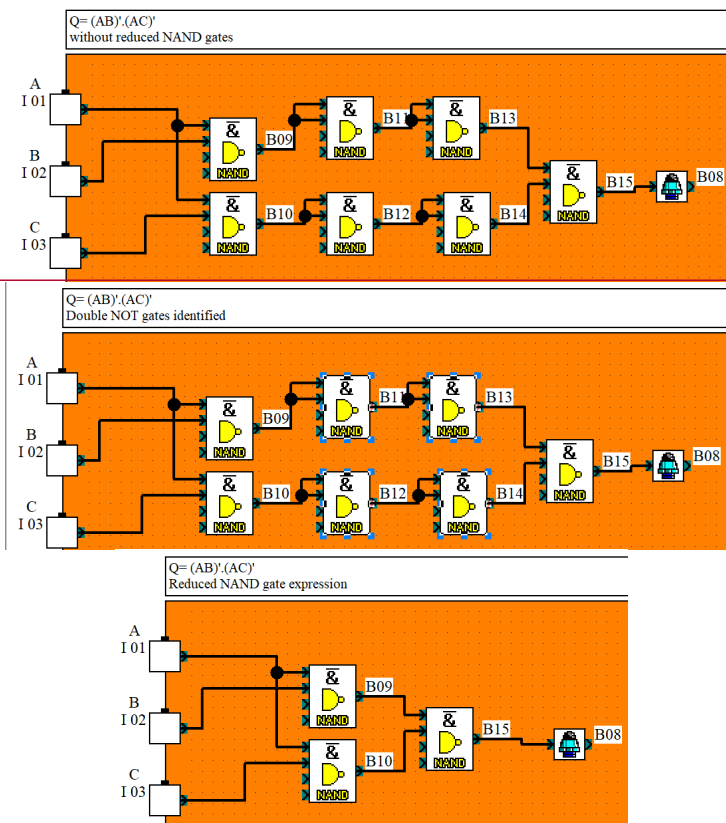
$$Q = AB + AC$$

$$\bar{Q} = \overline{AB + AC} \text{ Double invert both sides of expression}$$

$$Q = \overline{\overline{AB + AC}} \text{ Double inversion of Q is equal to Q itself.}$$

$$Q = \overline{AB} \cdot \overline{AC} \text{ de Morgans AND law}$$

Commented [RB2]: Only one Not is being expanded.



Commented [RB3]: So you are converting each gate into Nand gate first and then identifying the double not gates.

Commented [RB4]: I appreciate your use of other software but showing of the truth table could help more to verify the functionality of the circuit.

Figure 3 (i) Changing of basic gates to universal NAND gates including identifying Double inversion

Task1(ii)

$$Q = ABC + \bar{A}B + AB\bar{C}$$

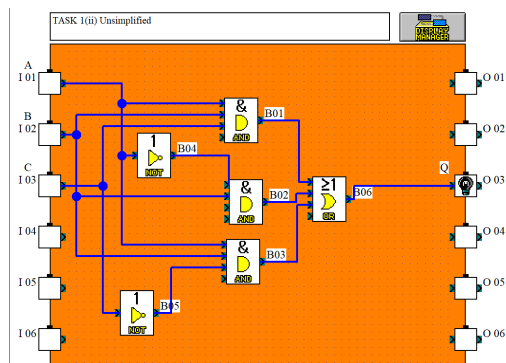


Figure 4 (ii) Un-simplified

$$Q = ABC + \bar{A}B + AB\bar{C}$$

$$Q = AB(C + \bar{C}) + \bar{A}B \text{ (AND Distributive)}$$

$$Q = AB.1 + \bar{A}B \text{ (OR Complement)}$$

$$Q = AB + \bar{A}B \text{ (AND Identity)}$$

$$Q = B(\bar{A} + A) \text{ (OR complement)}$$

$$Q = B.1 \text{ (AND identity)}$$

$$Q = B$$

Instances such as this highlight how important simplification can be, this has turned the use of 3 separate inputs and 6 logic gates to just one direct On/Off switch.

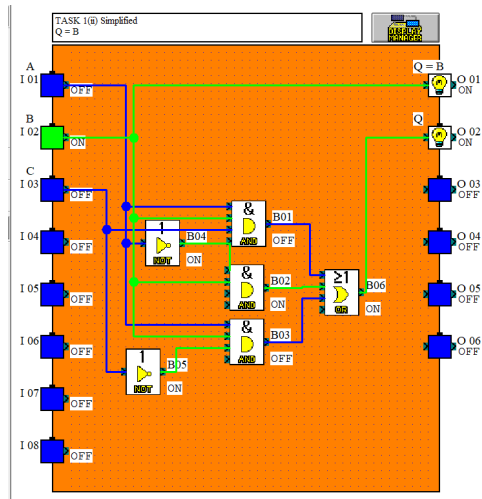


Figure 5(ii) Simplified Expression with simulation

Demorgans Theorem application

$$\bar{Q} = \bar{B} \text{ (Double inversion law)}$$

$$Q = B$$

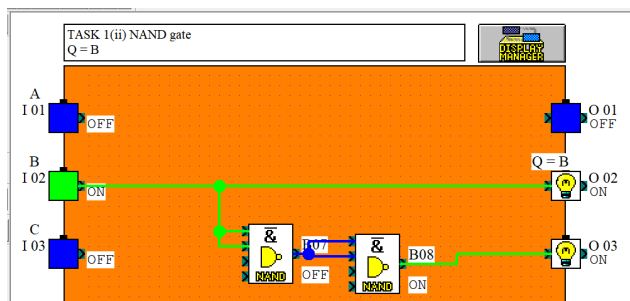


Figure 6 (ii) Solved for NAND gates

Figure 6 demonstrates that a double inversion should never be used as it is not necessary and will only use additional space.



Task1(iii)

$$Q = (A + B)(A + \bar{B})(\bar{A} + C)$$

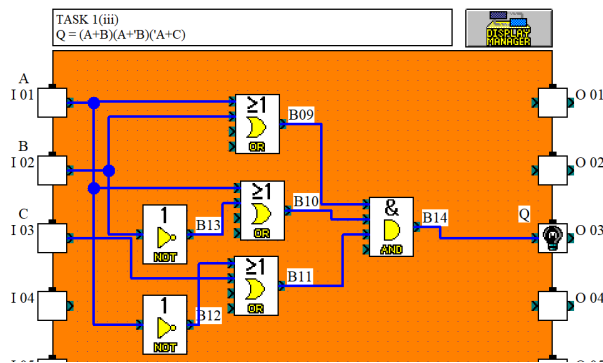


Figure 7 (iii) Un-simplified

$$Q = (A + B)(A + \bar{B})(\bar{A} + C)$$

$$Q = (A(B + \bar{B}))(\bar{A} + C) \text{ (OR Distributive)}$$

$$Q = (A \cdot 1)(\bar{A} + C) \text{ (OR Complement)}$$

$$Q = (A)(\bar{A} + C) \text{ (AND Identity)}$$

$$Q = A(\bar{A} + C) \text{ Expand brackets}$$

$$Q = A\bar{A} + AC \text{ (AND complement)}$$

$$Q = AC$$

$$A + B \cdot \bar{B}$$

$$A + a = A$$

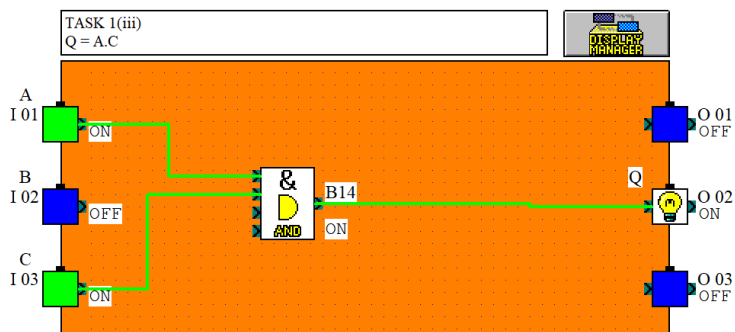
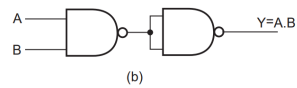


Figure 8 (iii) Simplified

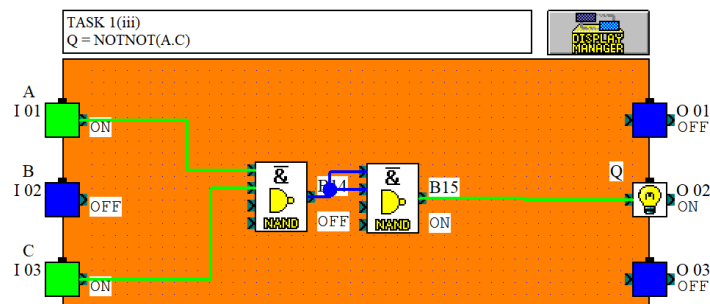
De Morgans Theorem in order to reduce this to NAND gates can be shown as:

$$\bar{Q} = \overline{AC}$$

$$Q = \overline{\overline{AC}} \text{ (Inverted AND gate is an AND gate with a NOT gate at output)}$$



The above image shows how to convert a standard AND expression into the NAND gate logic circuit.



Commented [RB5]: This task shows your ability to use De-Morgan's and Boolean algebra efficiently. However, use of Multisim to verify the result using truth table could have been better.

Task 2

Task 2⁽ⁱ⁾

Inputs	Description	Binary
A	Flow rate ≥ 5 litre/second	1
	Flow rate < 5 litre/second	0
B	Temperature $\geq 30^{\circ}\text{C}$	1
	Temperature $< 30^{\circ}\text{C}$	0
C	Pressure ≥ 1 atmosphere	1
	Pressure < 1 atmosphere	0

a) Flow rate < 5 litre/second **and** Temperature $\geq 30^{\circ}\text{C}$

Or

b) Flow rate ≥ 5 litre/second **and** Temperature $\geq 30^{\circ}\text{C}$ **and** Pressure < 1 atmosphere

Using the Information table above I produced the following truth table:

A	B	C	Q
$\bar{A}$	$\bar{B}$	$\bar{C}$	0
$\bar{A}$	$\bar{B}$	C	0
$\bar{A}$	B	$\bar{C}$	1
$\bar{A}$	B	C	1
A	$\bar{B}$	$\bar{C}$	0
A	$\bar{B}$	C	0
A	B	$\bar{C}$	1
A	B	C	0

As there are 3 occurrences of $Q=1$, to produce the Boolean expression I initially wrote down all outcomes which resulted in $Q=1$.

$$(A = 0, B = 1, C = 0) = 1$$

or

$$(A = 0, B = 1, C = 1) = 1$$

or

$$(A = 1, B = 1, C = 0) = 1$$

Expressed as:

$$Q = 1 = \bar{A}.B.\bar{C} + \bar{A}.B.C + A.B.\bar{C}$$

Commented [RB6]: Good explanation of the usage of given info.

Task 1(ii)

I built the Logic Circuit using the Mitsubishi Alpha PLC Programming software.

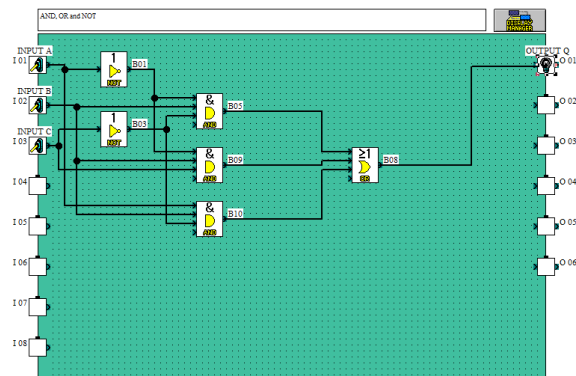


Figure 10 LOGIC circuit using AND, OR and NOT gates

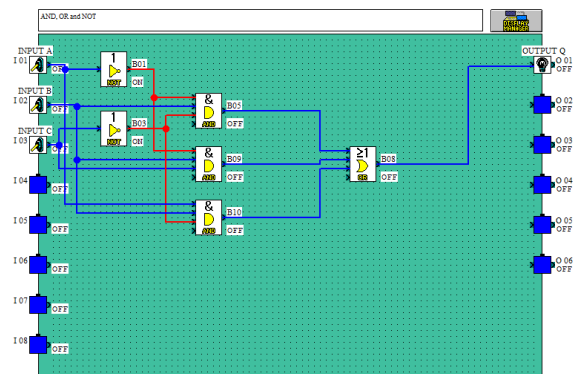


Figure 11 Simulation Start - NA/NB/NC Scenario $Q=0$

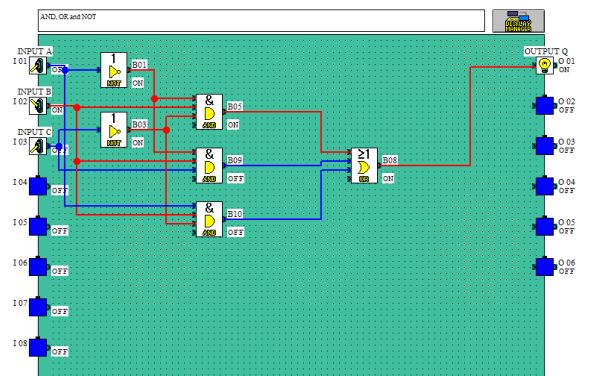


Figure 12 NA/B/NC Scenario $Q=1$

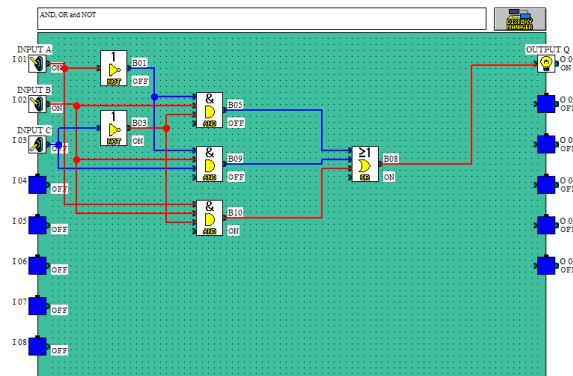


Figure 13 A/B/NC Scenario $Q=1$

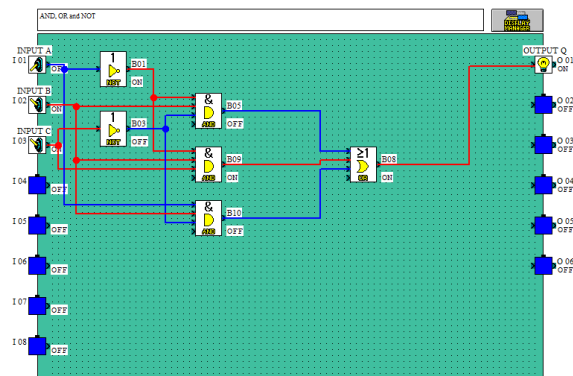


Figure 14 NA/B/C Scenario $Q=1$

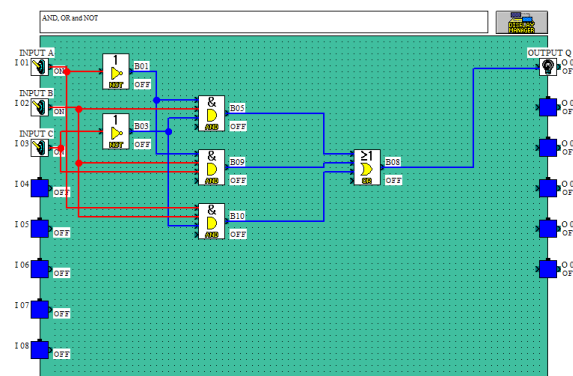


Figure 15 A/B/C Scenario $Q=0$

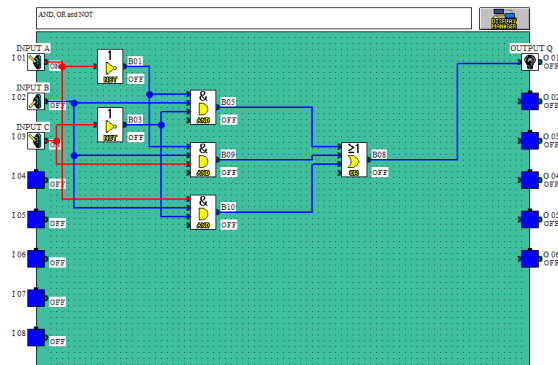


Figure 16 A/NB/C Scenario Q=1

Task 2⁽ⁱⁱⁱ⁾

This expression can be simplified further by using the rules of Boolean algebra.

$$Q = 1 = \bar{A}.B.\bar{C} + \bar{A}.B.C + A.B.\bar{C}$$

$$Q = 1 = \bar{A}.B.(C + \bar{C}) + A.B.\bar{C} \text{ Using **Distributive law** } AB + AC = A(B + C)$$

$$Q = 1 = \bar{A}.B.1 + A.B.\bar{C} \text{ Using **complement law** } A + \bar{A} = 1$$

$$Q = 1 = \bar{A}.B + A.B.\bar{C} \text{ Using **Identity law** } A.1 = A$$

$$Q = 1 = B(\bar{A} + A.\bar{C}) \text{ Using **Distributive law** } AB + AC = A(B + C)$$

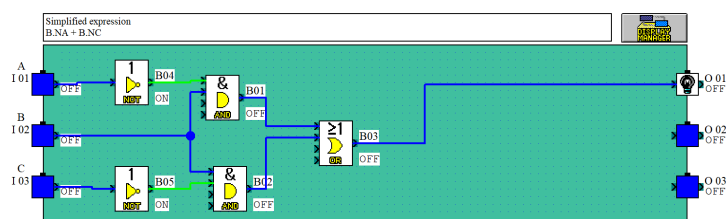
$$Q = 1 = B(\bar{C} + \bar{A}) \text{ Using **Absorption law** } AB + \bar{A} = B + \bar{A}$$

$$Q = 1 = B\bar{C} + B\bar{A} \text{ Expanding brackets for final solution}$$

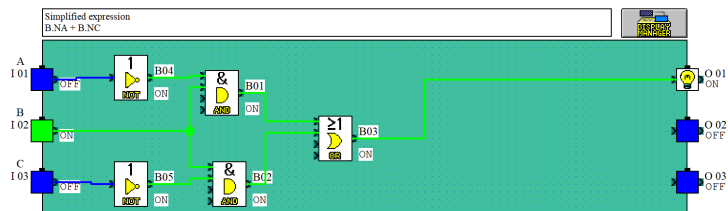
Commented [RB7]: Very elaborative.

Simulations

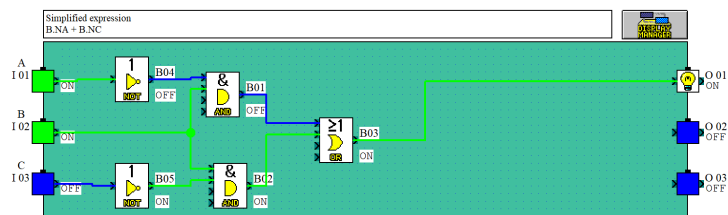
$$\bar{A}, \bar{B}, \bar{C} \ 0,0,0 \ Q=0$$



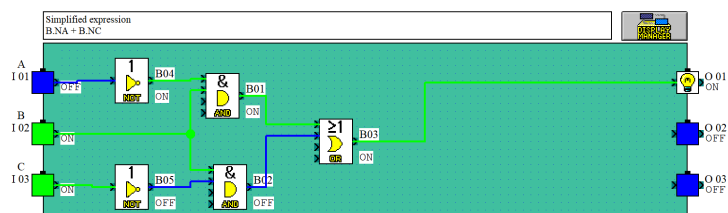
$$\bar{A}, B, \bar{C} \ 0,1,0 \ Q=1$$



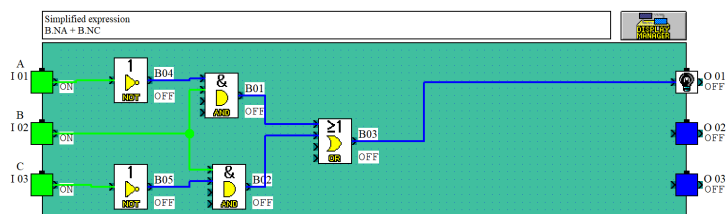
$A, B, \bar{C} \ 1, 1, 0 \ Q=1$



$\bar{A}, B, C \ 0, 1, 1 \ Q=1$



$A.B.C \ 1, 1, 1 \ Q=0$



Applying De-morgans theorem for single NAND gate circuit.

$$\bar{Q} = \overline{\bar{A}.B + B.\bar{C}}$$

$$Q = \overline{\bar{A}.B + B.\bar{C}} \text{ The Double complement law } \bar{\bar{A}} = A$$

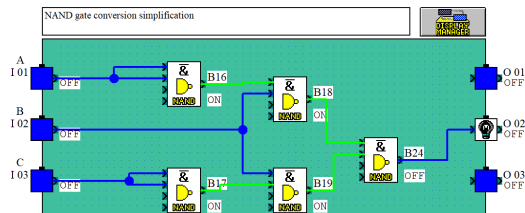
$$Q = \overline{\bar{A}.B} . \overline{B.\bar{C}}$$

$$Q = \overline{\bar{A}.B} . \overline{B.\bar{C}}$$

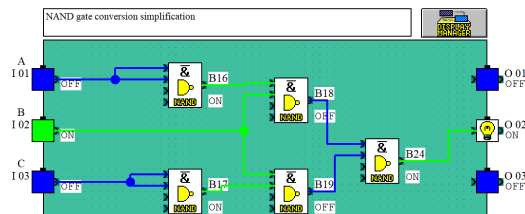
Commented [RB8]: Not right

Simulation

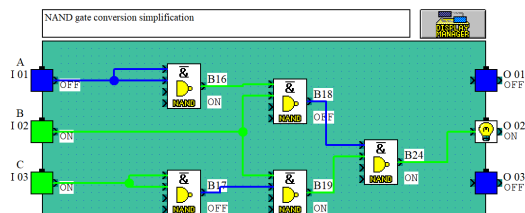
$$\bar{A}, \bar{B}, \bar{C} \ 0,0,0 \ Q=0$$



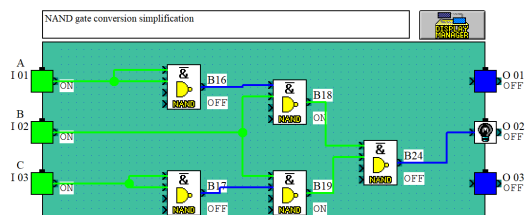
$$\bar{A}, B, \bar{C} \ 0,1,0 \ Q=1$$



$$\bar{A}, B, C \ 0,0,0 \ Q=1$$



$$A, B, C \ 0,0,0 \ Q=0$$



Task 2 shows the importance of simplifying Boolean expressions to reduce logic gates, ease of manufacture, efficient, and cost effective. The conversion from the basic AND, OR and NOT gate functions to solely NAND gate-based circuits shows it's worth as the same outcome can be achieved using a single type of gate, this is

considerably more cost effective as the number of Integrated circuit chips used to produce the desirable circuit is much less.

The Universal NAND and NOR logic gates are produced on a far superior scale to ICs which house only the other basic gates, it does not make sense to produce these chips in the same numbers.

The Logic gates discussed are all created using different types of transistors in different switching configurations to attain a desired output, other basic electronics components are also used to make this possible.

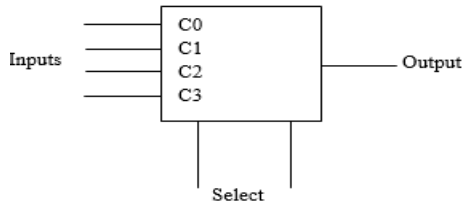
The first circuit produced in this task used a total of 6 gates of 3 different types, this may of used a total of 6 different IC chip CMOS or TTL transistor configurations. The circuit was simplified to use only 5 NAND gates which could limit the number of ICs used to just 2. There are both the TTL 7400 series Quad 2 input NAND gate chip or the CMOS 4011 Quad 2 input NAND gate chip which would be suitable, although CMOS would be the more likely choice, due to its low power dissipation, high fan out per gate characteristics and stable operation across a wide range of temperatures.

*“The **advantage of the CMOS over the TTL chips** is that the CMOS has a higher **density of logic gates** within the same material. TTL chips **consume more power** as compared to the power consumed by the CMOS chips even at rest.”*

- (<https://components101.com/articles/ttl-vs-cmos-ics-and-how-to-choose-between-them>)

Commented [RB9]: Interesting finding.

Task 3



A Multiplexer, or MUX is a Multiple Input / Single Output device which uses inputs known as Select inputs to switch a desired input to be connected to the output. The number of Select inputs is determined by the total number of Optional inputs, A MUX which has only 2 Optional inputs, requires only 1 select inputs, as it only requires two possible outcomes to switch between inputs. Each select input is binary, so is either 1 or 0.

The truth table for the 4-1 MUX above would require 4 possible outcomes to switch between optional inputs – Therefore a minimum of 2 inputs are required for this application.

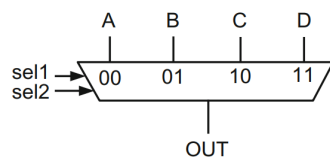


Figure 17 4-1 MUX symbolic diagram

(Electronics for Embedded systems, pdf online, Ahmet Bindal, Springer International Publishing Switzerland, 2017)

Commented [RB10]: Not the right way of citing a reference.

Truth Table:

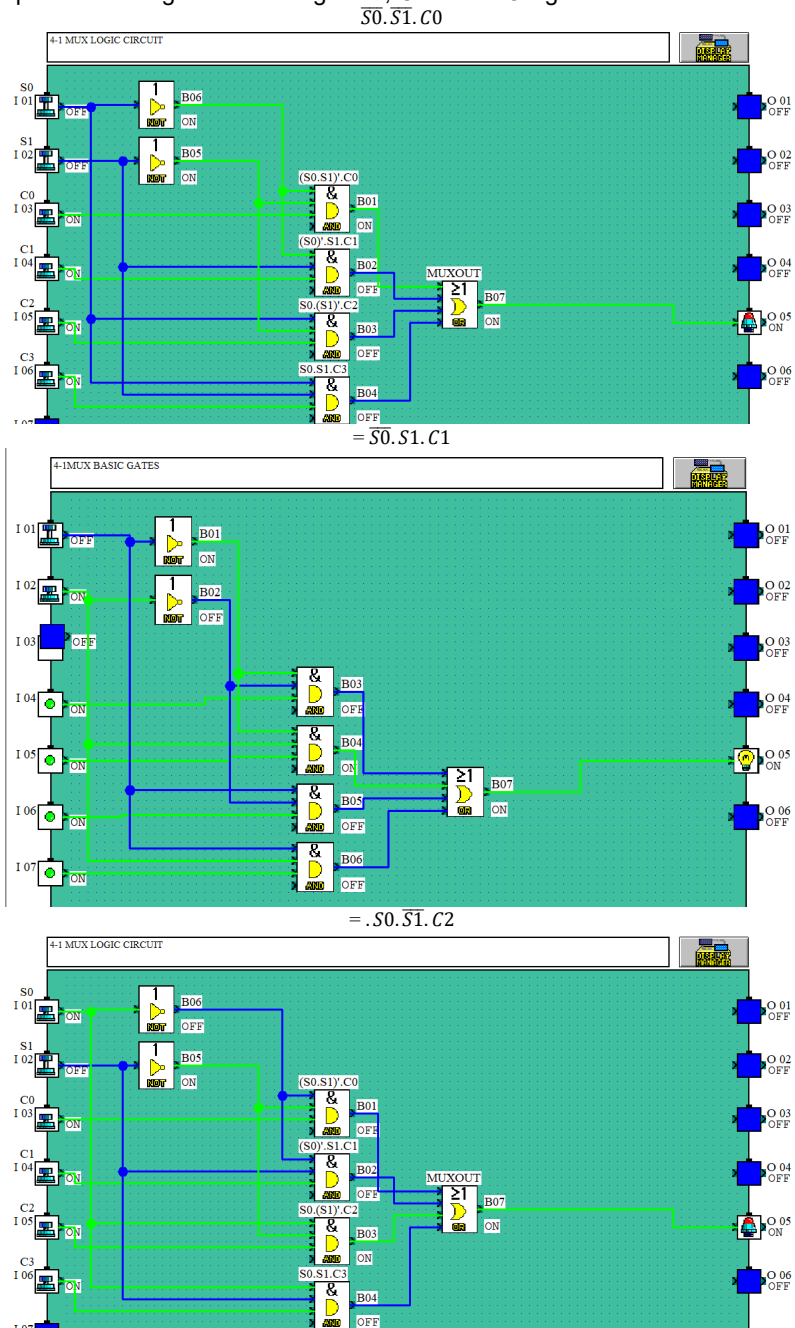
S0	S1	MUXOUT
0	0	C0
0	1	C1
1	0	C2
1	1	C3

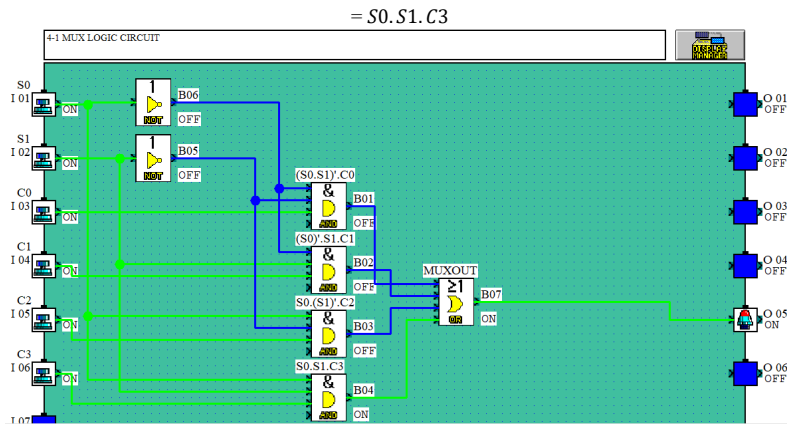
Table 2 4-1 MUX Truth table

Using the Truth table above, I produced the following Boolean Expression:

$$\text{MUXOUT} = \overline{S0}.\overline{S1}.C0 + \overline{S0}.S1.C1 + S0.\overline{S1}.C2 + S0.S1.C3$$

I then produced a logic circuit using AND, OR and NOT gates:





Using De-Morgans theorem, I converted this to use only NAND gate logic.

$$M/O = \overline{S0}.\overline{S1}.C0 + \overline{S0}.S1.C1 + S0.\overline{S1}.C2 + S0.S1.C3$$

$$M/O = \overline{S0}.\overline{S1}.C0 + \overline{S0}.S1.C1 + S0.\overline{S1}.C2 + S0.S1.C3$$

Commented [RB11]: Need to expand one not, to use NAND gates to implement this logic.

To Convert an AND gate to a NAND gate, you must replace the AND gate with a NAND gate and Invert the output. To convert an OR gate to a NAND gate, the preceding inputs must be Inverted.

“so the NAND gate is equivalent to an AND gate followed by an inverter, as shown in Figure 7-8(b). A more appropriate name would be an AND-NOT gate, but we will follow common usage and call it a NAND gate.”

- (Fundamentals of logic Design, 2010, C.H Roth/L.L Kinney, USA)

The figure below shows how this would have been done for the task 3 logic circuit, In the second image I have identified the double inversions and thus would be able to reduce the gates used further as these are not needed according to the Double inversion rule.

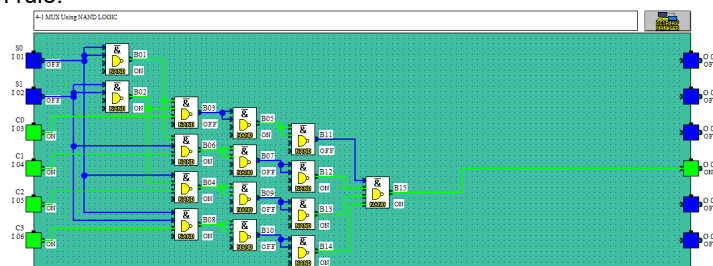


Figure 18 Shows the NAND gate conversions from previous basic logic circuit

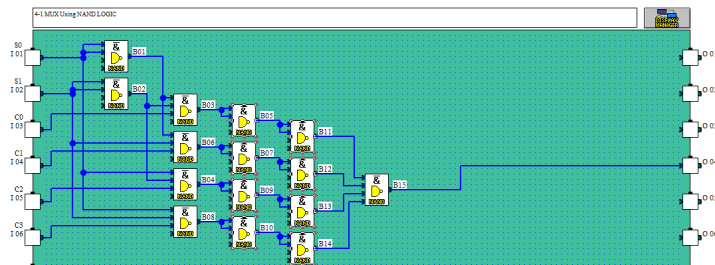
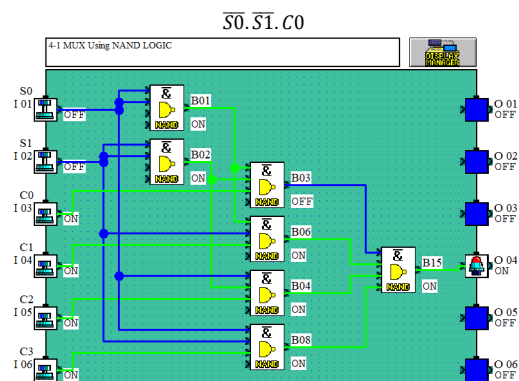
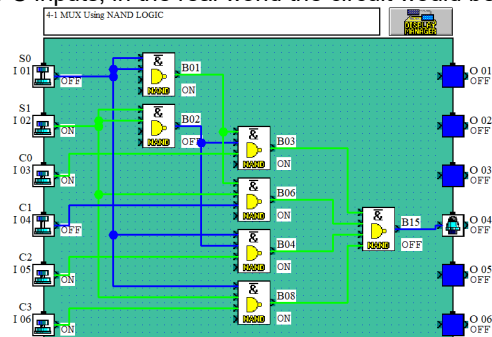
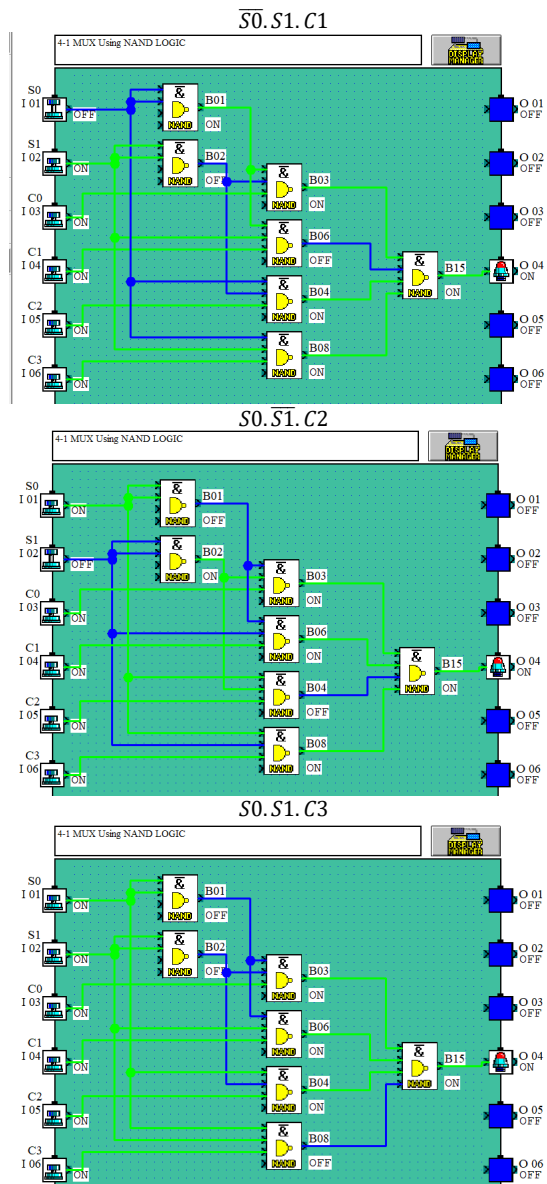


Figure 19 Double Inversions identified for removal



With C1 off but Select inputs correct, this shows that if there is nothing actually at the C inputs, in the real world the circuit would be off.





Task 4

Task4⁽¹⁾

Sequential logic differs from Combinational logic as it uses both the present value of the input and the past value of the input to determine the new output. Flip-Flops are used as the building blocks of Sequential logic circuits, compared to the standard logic gates of Combinational logic.

The combinational circuit is one whose outputs depend only on the present value of the input(s), independent of the past history of the input(s). The sequential circuit is, on the other hand, is a circuit whose output depends not only on the present value of the input but also on the past history of the input

(Foundation of Digital Electronics and Logic Design, [S.K Sarkar][A.K De][S Sarkar], 2014, Taylor and Francis group,[120, 2.2. 1])

Flip-Flops are basic memory devices used in Digital logic circuits and can hold 1-Bit of memory. Flip-Flops are also called Bi-stable Multivibrators, due to them having only two stable states, 0 or 1. When Flip-flops use a clock pulse to set the output according to a certain state depending on both the input state at the point of the pulse and the characteristic of the individual flip-flop.

Converting D Flip-Flop to JK Flip-Flop

To convert the D F-F to J F-F, I first gathered the excitation truth tables of both flip flops to evaluate the characteristics of the current value (Q_n) and next value (Q_{n+1}).

JK FF Excitation Truth Table				
Q_n	J	K	Q_{n+1}	Comments
0	0	0	0	Q_n
0	0	1	0	
0	1	0	1	
0	1	1	1	Toggles previous Q
1	0	0	1	Q_n
1	0	1	0	
1	1	0	1	
1	1	1	0	Toggles previous Q

D FF Excitation TT		
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Commented [RB12]: Excitation table is more about expressing input from the output.

I now used the information in the above tables to produce an excitation truth table which includes the values of D when the Q_n and Q_{n+1} values for the JK flip flop are equivalent.

Q_n	J	K	Q_{n+1}	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

D FF Excitation TT		
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

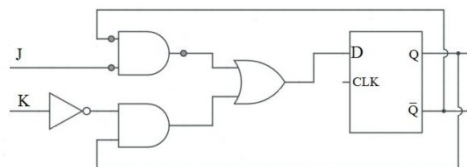
Highlighted in yellow is an example of Q_n and Q_{n+1} being transferred across to the converting TT. I then used a Karnaugh Map to show all resulting outputs of Q_n , J and K which D equals 1.

$Q_n/J.K$	0.0	0.1	1.1	1.0
0	0	0	1	1
1	1	0	0	1

The Boolean expression for the above Karnaugh maps is displayed as:

$$D = Q_n \cdot \bar{K} + \bar{Q}_n \cdot J$$

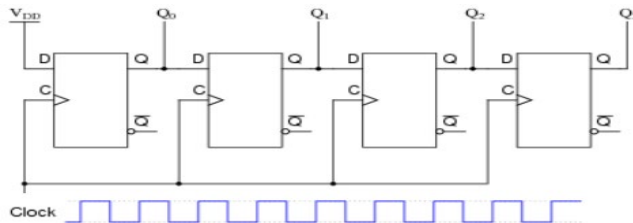
The equation can be used to convert a standard D F-F to a J.K F-F by applying the logic gates to the input of the D F-F, diagram as shown below:



It is very simple to get to this point, and in addition to this you can furthermore adjust it so that only NAND gates are used.

Commented [RB13]: Task in the assignment require you to convert Jk flip flop into D flip flop.

Task 4(ii)



As all Q inputs were set to 0, and we needed all Q outputs to be set to 1 in a sequence of clock pulses. The clock input has a single arrow at its connection point which indicates the output sees the input on the rising edge of the clock pulse. The Most Significant Bit and the Least significant Bit are displayed as:

$$\begin{array}{r} \text{Desired output} = 1111 \\ \text{MSB} \quad \quad \quad \text{LSB} \\ \hline 1 - 1 - 1 - 1 \end{array}$$

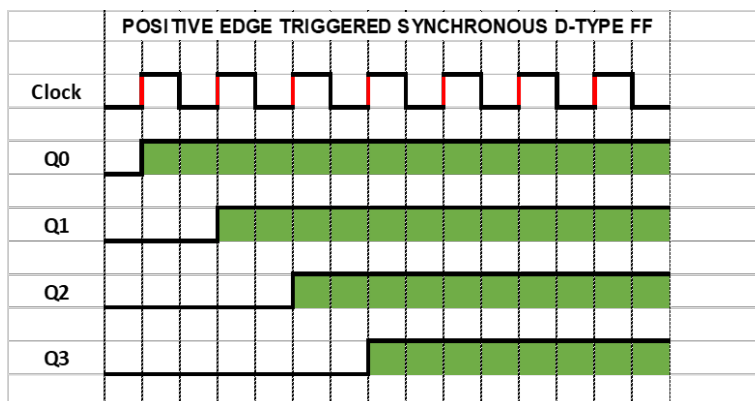
The least significant Bit of the desired output will always start at the DATA input prior to the first clock pulse.

And the Truth Table shows this:

The Data input at clock 4 is irrelevant to this task description

Clock	Data	Q0	Q1	Q2	Q3
initial	1	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	1	0
4	0	1	1	1	1

The timing diagram for the above circuit is as shown as:

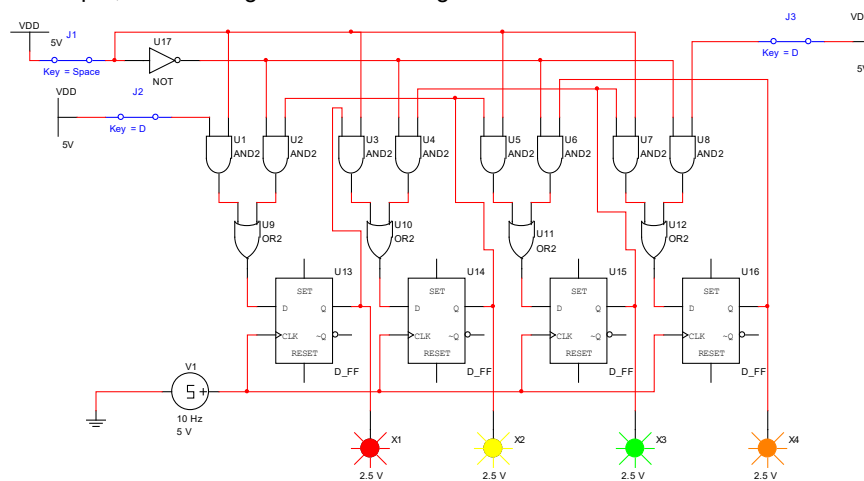


Commented [RB14]: Good

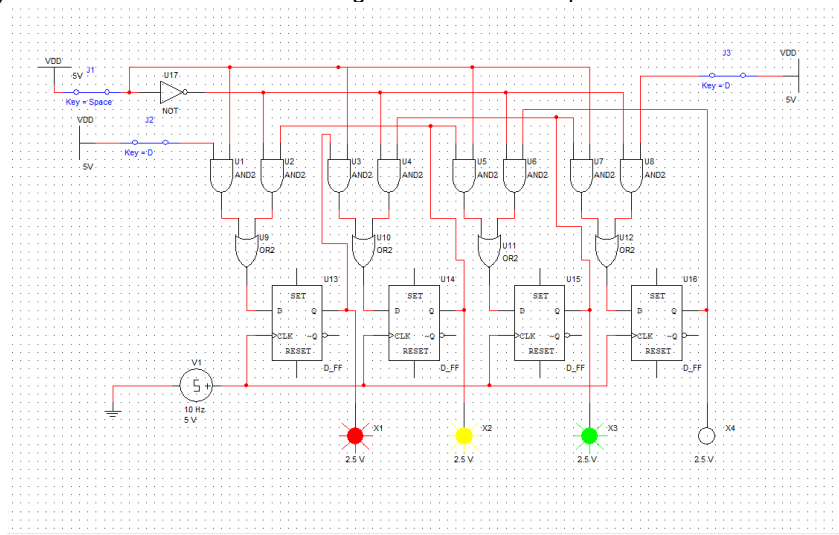
Bidirectional Shift Register (Series In Series Out SISO)

To create the bi-directional shift register, I needed two different Data inputs, one for Left direction and one for Right direction, I would then need a Select or Mode input to switch between directions. The Mode input would allow a logic 1 to the side of the circuitry which direction we desired. As the circuit would either go left OR right, I knew that I would need 2 AND gates (for either data input / Mode input) and an OR gate per Flip-Flop. The Bi-directional characteristic was achieved by connecting the Shift Left data input to the input of the first AND gate, this was then connected so that when the Mode was switched to on, the 4 Q inputs would receive a high (1) and then upon the clock pulse, the value would move from the D Shift left input from Left to Right. When Mode was deselected, the data would be taken from the Shift Right data input, thus moving the data from Right to left.

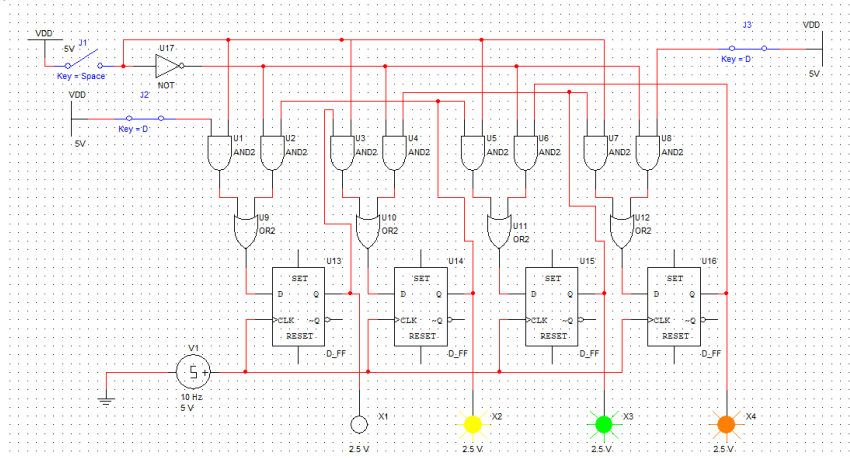
Commented [RB15]: Not clear



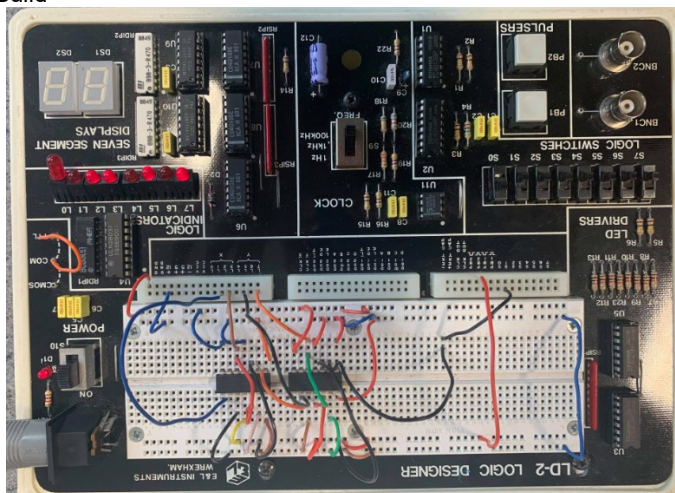
(a) shows the switch from Left to Right with the Mode input closed



(b) shows the Shift from Right to Left with the Mode input Open.



(c) Circuit Build



The Bi-Directional shift register is a useful circuit in practical applications such as expanding the number of Digital inputs and outputs of a microcontroller by optimising the amount of Outputs per I/O. I used the 74HC74 Dual D-type flip flop to create the circuit and referred to the specific datasheet for specific electrical ideals.

Task 5

From the task spec, I have taken the key points:

1. 3 Stage
 - a. This will need 3 Bits, as a JK F-F is capable of holding 1 bit, I know I will need 3.
2. Synchronous
 - a. I know that Synchronous means that all F-F in the circuit are clocked using the same clock pulse.
 - b. Synchronous circuits are more difficult to design but are much more practical than asynchronous.
3. Binary equivalent of 0,1,3,5,7
 - a. The first 8 digits of the numerical binary code can be seen as below, I first produced this to help identify the combinations I required for the circuit. Highlighted in Blue are the identified combinations for each number listed in the task spec. To produce the 8 digits, 3 bits of memory are required.

Binary			Decimal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Commented [RB16]: Could be written in one paragraph to have it presentable.

I produced a table to show the next desired state of the counter.

1 st INPUT			DESIRED OUTPUT		
A	B	C	A+	B+	C+
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	1	0	1
1	0	1	1	1	1
1	1	1	0	0	0

I then used the truth table of a JK flip flop to produce a next state truth table of the table above.

A	A+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

The next state truth table is used to confirm the desired outcome of each J or K input to change or maintain its current state in order to produce the binary counter 0,1,3,5,7.

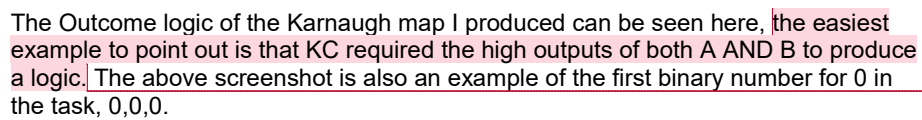
A	B	C	A+	B+	C+	JA	KA	JB	KB	JC	KC
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	1	1	0	1	1	X	X	1	X	0
1	0	1	1	1	1	X	1	1	X	X	0
1	1	1	0	0	0	X	0	X	1	X	1

The values highlighted in yellow above represent the desired inputs of each JK F-F to change the 0,0,0 = 0 value to 0,0,1 = 1, this requires a high at the J input of the 3rd Flip Flop.

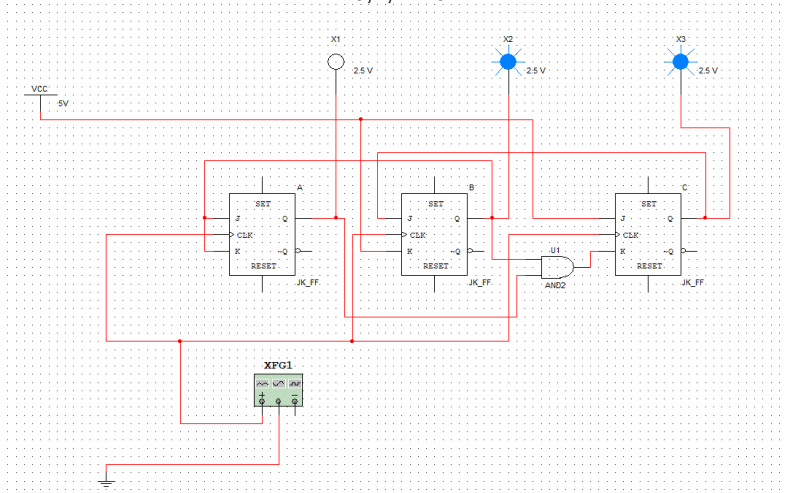
I produced a Karnaugh Map to display the different input gate characteristics required to produce the circuit. All 0s are discarded but need to be put in the map to avoid grouping mistakes.

JA				
A / BC	00	01	11	10
0	0	0	1	x
1	x	x	x	x
JA = B				
7KA				
A / BC	00	01	11	10
0	x	x	x	x
1	x	0	1	x
KA = B				
JB				
A / BC	00	01	11	10
0	0	1	x	x
1	x	1	x	x
JB = C				
KB				
A / BC	00	01	11	10
0	x	x	1	x
1	x	x	1	x
KB = 1				
JC				
A / BC	00	01	11	10
0	1	x	x	x
1	x	x	x	x
JC = 1				
KC				
A / BC	00	01	11	10
0	x	0	0	x
1	x	0	1	x
KC = AB				

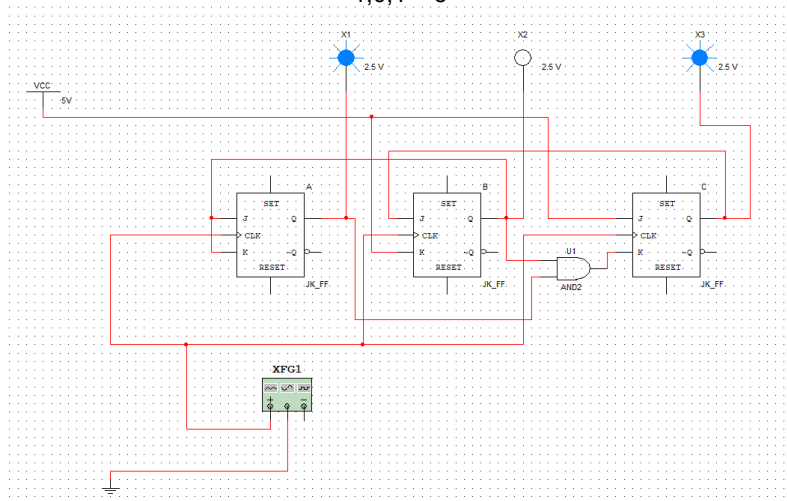
Highlighted in yellow are the groupings within the Karnaugh maps, where the whole group has been highlighted, the desired outcome of the input is always equal to 1.

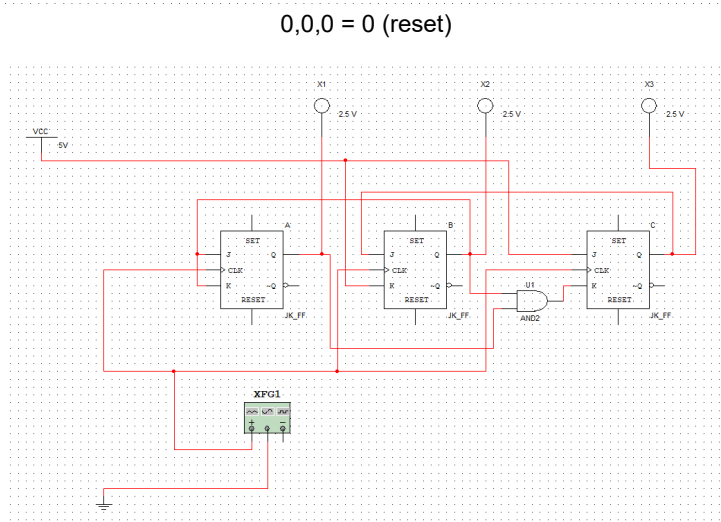
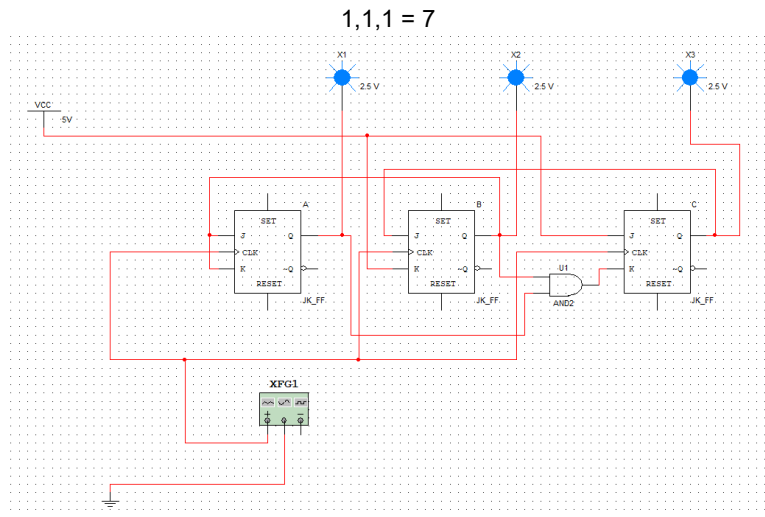

$$0,0,1 = 1$$


0,1,1 = 3



1,0,1 = 5





This circuit was relatively simple to build and would require a the use of two 74LS107 IC chips and a 7408 IC to build practically. If I were to improve the circuit display, I would have tried to include a 7-segment display to display the outputs as a decimal number.

Commented [RB18]: Good suggestion.

Task 6

The fundamentals of the creation of a set of digital circuit where the electrical characteristics are the same are defined as a logic family.

Logic families are extremely significant in the design and build of Digital Electronic circuits as ensuring that all IC's in the design are of the same family will guarantee that the whole circuit will react identically to changes in electrical characteristics.

In instances where the outputs of IC's within one logic family feed in to the inputs of IC's of another logic family, appropriate interface/conversion techniques should be applied to avoid this affecting the overall performance of the circuit.

Digital IC logic come in form of Bipolar families or Metal Oxide Semiconducting (MOS) families. Examples of Bipolar subfamilies include:

Diode Logic (DL)
Resistor Transistor Logic (RTL)
Diode Transistor Logic (DTL)
Transistor Transistor Logic (TTL)
And more.

MOS subfamilies include:

P-channel MOS (PMOS)
N-Channel MOS (NMOS)
Complementary MOS (CMOS)
And more.

Many logic families are now obsolete due to the level of development making some now insignificant and not worth producing.

Logic families that are still in widespread use include TTL, CMOS, ECL, NMOS and Bi-CMOS. The PMOS and I²L logic families, which were mainly intended for use in custom large-scale integrated (LSI) circuit devices, have also been rendered more or less obsolete, with the NMOS logic family replacing them for LSI and VLSI applications.

- (*Digital Electronics, A K MAINI, WILEY, 2007 England*)

As per the above quote I will compare the TTL and CMOS logic families as they are still widely used today.

Commented [RB19]: Intro is not succinct, I am afraid. And it could be presented in a better way.

Characteristic	CMOS 4011 series	TTL 7400 series
Propagation Delay The time taken for a signal to travel from the Input to the Output of a logic gate.	4011 has a propagation delay of up to 250ns, considerably slower than the TTL 7400 series equivalent.	7400 has a propagation delay of either 15 or 22ns depending on whether its running a low to high or high to low output.
Power Dissipation Wasted or lost power of a logic gate due to internal resistance.	Far lower power consumption in general, very efficient, the main reason they are widely used today.	Much higher power consumption making them far less desirable.
	Power dissipation is calculated by $\frac{V_{cc}I_{cc}(avg)}{n}$	
Speed Power Product Amount of speed produced by the lowest power	The SPP of a CMOS chip will likely be far lower (better) than TTL due to the extremely low power dissipation of the ICs.	As high power is consumed within the TTL logic, the value of SPP will almost always be higher than that of the CMOS chip.
	To Calculate SPP = $S_{pd} \times P_d$, expressed in Joules. A singular FET combined gate on a CMOS chip will take far less components to execute than the equivalent gate on a TTL logic chip due to the quantity of components needed.	
Fan In Number of gate inputs	CMOS inputs are neither low or high and if any are not connected to anything, they must be tied to either 0V or the supply, as they are susceptible to noise otherwise.	Up to 8 inputs before a network is required to allow for more
Fan Out Number of gate outputs	Up to 50 or more outputs can be implemented from a single CMOS input, but the higher the input the longer the propagation delay	TTL outputs are either low or high and there is typically 10 of each.
Voltage parameters Voltage threshold at which the logic decides either logic 1 or 0.	Input: 0-1.5 = low 3.5-5 = high Noise margin (Output): 0.05 = 1.45 low 4.95 = 1.45 High	Input: 0-0.8V = low 2-5V = logic high Noise Margin: 0.5-0.8 = 0.3 2 – 2.7 = 0.7

Commented [RB20]: ?

Overall I would be far more likely to design a logic circuit with the use of CMOS based chips as they would be far more efficient where power consumption is concerned, especially in a day and age where computers and the digital logic is the core of most workplaces.

References/Bibliography

Anil Kumar Maini, 2008, <i>Digital electronics and logic design</i>
Bindal, A. 2017. <i>Electronics for Embedded systems</i> . Springer International Publishing Switzerland, 2017
Roth, C., Kinney, L. and John, E., 2010. <i>Fundamentals of logic design</i> .
Sarkar, SK, De, AK, Sarkar, S. 2014, <i>Foundation of Digital Electronics and Logic Design</i> , Taylor and Francis group.

Websites

<https://components101.com/articles/ttl-vs-cmos-ics-and-how-to-choose-between-them>

Word Count : 2277 without Tables/References

STUDENT FEEDBACK FORM

This section details the *extent* to which the assessment criteria are demonstrated by you, which in turn determines your mark. The marks available for each category of skill are shown. Lecturers will use the space provided to comment on the achievement of the task(s), including those areas in which you have performed well and areas that would benefit from development/improvement.

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills	10	
Your work shows engagement with an appropriate range of literature including resources retrieved independently, but with some reliance on the information gained from class contact (especially in task 6). You also need to pay attention to the text citation to improve in this section, I'm afraid.	52% 10	5.2
2. Knowledge and Understanding Skills	30	
Your work demonstrates your competent knowledge and understanding of the basic underlying concepts of digital electronics. You have used appropriate techniques and methods to complete all set tasks given in the assignment. You have demonstrated the excellent knowledge of Boolean algebra, however, few inaccuracies were there in implementing De-Morgan's theorem. Also, at times, detailed explanation of the underlying concepts could be provided such as you could explain how the usage of additional gates can help to achieve the bi-direction shift logic and could also provide one case in counters task to explain its working etc.	68% of 30	20.4
3. Cognitive and Intellectual Skills	10	
You have done a sound evaluation of underlying concepts and principles, showing your ability to devise an argument from evidence. Your supporting statement for the outcome of task 1 and task2 are well appreciated, however, at times, your evaluation in other tasks such as Multiplexer and shift registers could be expanded by relating it to real-world situations.	68% of 10	6.8
4. Practical Application Skills	40	

Aaron the software used to do task 1 and 2 was not appropriate to meet the requirements of the tasks, however for other tasks your usage of software to defend your theoretical results shows your excellent ability to perform simulations and reach a conclusion. Enough screen shots are provided per task to help understanding the working of various circuits, however, explaining the screen shots at the same time could have helped more. Though, your attempt to suggest the improvements in the various circuits is worth appreciation.	68% 40	27.2
5. Transferable Skills for Life and Professional Practice	10	
Most of your work is organised and accurate. All the calculations are well represented using equation editor. However, at times, your work needs structure and sentences are vague, hard to follow and sometimes paragraphs are not complete, especially in the intro of logic families. Proof reading of your work before submission could help you to avoid these mistakes. Overall, your work shows your ability to work independently and as a part of the team.	55% 10	5.5

Assessment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)	Late Submission Penalties (tick if appropriate)		65.1%
	Up to 1 week late (40% Max)		
	Over 1 week late (0%)		