Canadian Journal of Pure and Applied Sciences
Vol. 15, No. 3, pp. 5297-5305, Oct 2021
Online ISSN: 1920-3853; Print ISSN: 1715-9997
Available online at www.cjpas.net

LOGIC GATE ARITHMETIC AND QUATERNIONS

Christopher C. O'Neill
Cataphysics Group, Recess, Old Connaught Ave. Shankill, Co. Dublin, Ireland

ABSTRACT

A new method for Logic Gate Arithmetic is proposed allowing sets to be added, subtracted, multiplied, and divided in an novel way leading to an alternate and expanded appreciation of the Set Theory. An additional eleven Logic Operators are derived from the known operators: AND, OR, NOR, XOR, and XNOR, and their relationships described. Utilizing the principles of Dimensional Gate Operators (DGO), the new operators are shown to be directly related to dimensionality, both higher and lower, and the relationship to the Quaternions is given preliminary treatment.

Keywords: Logic Gate Arithmetic, set theory, dimensional gate operators, boolean algebra, quaternions.

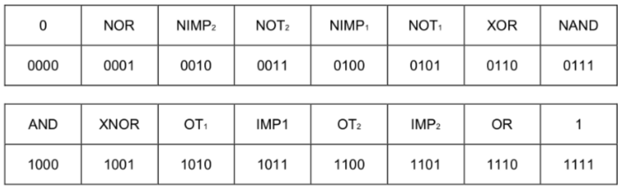
INTRODUCTION, BACKGROUND, AND TERMINOLOGY

Logic Operators are an important part of modern Computer Systems and Set Theory. There are six primary Logic Gate Operators; AND, OR, NOR, XOR, XNOR, and NOT, as they are traditionally perceived. More accurately, it can be said that there are only 3 (AND, OR, and XOR) with the NOT operator being more of a decorator acting on these to create the other 2 (anti-operators) for a total of 5.

O'Neill (2021) explored the possibility that these Logic Operators were incomplete and added 11 more, for a total of 16. Approximately seven months after writing this, the author became aware of a video by N.J. Wildberger called "Implication and 16 logical operations" and realized that Wildberger had arrived at the same conception in 2018 (https://www.youtube.com/watch?v=XkqmuUg_yFs). Wildberger claims to have derived his new set of logic operations in a formal algebraic sense using the Boolean Algebra (Boole, 1847) and proceeds to name a selection of them in relation to the Stoic Logic of Modus Ponens and others in relation to their own internal properties.

For instance, the Logic or Truth table that corresponds to the binary number '0000' (read right to left), he denotes as '0' or 'zero'. It's counterpart '1111' is therefore '1'. 'IMP' stands for implication and OT1 stands for 'Original Term'.

Once a naming convention is in place, it matters little how it got there. However, these new logic operators lack Boolean Algebra symbols, like '^', 'V', 'Δ', and '!Δ' (Bocheński, 1948; Roegel, 2002). To counteract this, a new naming convention will be outlined in this theoretical work. The author developed this method, before realizing that Wildberger had already given names to all the logic operations.



0	NOR	NIMP$_2$	NOT$_2$	NIMP$_1$	NOT$_1$	XOR	NAND
0000	0001	0010	0011	0100	0101	0110	0111

AND	XNOR	OT$_1$	IMP1	OT$_2$	IMP$_2$	OR	1
1000	1001	1010	1011	1100	1101	1110	1111

Fig. 1. Wildberger's 16 Logic Operations, which are almost identical to the Logical Connective operators which have been in use and development for over a hundred years.

In any case, such naming conventions are not too concerning, especially in early stage research such as this. After all, the XOR and XNOR logic gates are sometimes called EOR and ENOR, without much complication, and there are numerous variations used for symbols throughout the literature, including '!' and '¬' for NOT, etc. (Roegel, 2002).

More confusingly, however, is applying the binary numbers to the Truth Tables. Binary numbers are usually written horizontally, whereas Truth Tables are nearly always vertical (see in Appendix 1). When compiling Wildberger's operations into the tables shown in Figure 1, it seemed necessary to list the operators, as though their binary numbers are being read from 'right to left'. To avoid headaches, it will be necessary to continue with this convention. In this paper, all 16 Logic Operators will be derived from the original six.

Corresponding author e-mail: chris.ozneill@gmail.com

Wildberger does not reveal how it was that he developed his version of the 16 Logic Operators. But given their similarities to the Logical Connectives (Bocheński, 1948), as well as Wildberger's extensive knowledge on the subject, it would appear likely he was aware of them in advance. Even so, the author is not aware if this would be classed as an act of plagiarism, given that the information in the form of an online video presentation, as opposed to a scientific paper.

More on logic

The binary numbers '0000' to '1111' can be used to represent all 16 of the Dimensional Gate Operators (DGO). However, some of these operations don't appear all that logical. To alleviate this concern, the author decided to attempt to derive the other eleven operators from the original five.

?	NOR	?	?	?	?	XOR	NAND
0000	0001	0010	0011	0100	0101	0110	0111

AND	XNOR	?	?	?	?	OR	?
1000	1001	1010	1011	1100	1101	1110	1111

Fig. 2. The other two tables: "A rose by any other name?"

The first step is to combine the known logic gates; AND, OR, XOR, NAND, XNOR, and NOR to see if they can generate any knew terms. XOR can be obtained via the OR operator acting on AND and OR:

$$((\wedge)\ V\ (V)) = \Delta$$

However, this is obvious and doesn't lead us out of the recursive loop and into the unknown parts of the DGO. Similarly, $((\wedge)\ \wedge\ (V)) = \wedge$. It is easy to see how this method will lead us in circles.

Taking a step back and reexamining the Truth Tables for these operations, there are clearly two more 'binary' numbers present in each of the tables. Which are: '1010' and '1100'. These numbers are equivalent to Wildberger's OT1 and OT2. In terms of Binary Connectives, these are known as Proposition P and Q. However, this author identified them as RIGHT (or 'R') and LEFT (L), as in Right and Left Multiplication. Applying the NOR operator, !R (NOT RIGHT) and !L (NOT LEFT) are easily obtained. From here, we can get to a new Logic Operator; '0100', which is made from the combination of Δ and !L:

$$((\Delta)\ \wedge\ (!L)) = 0100$$

It was called 0100 or '^U' (meaning AND UP), because it is similar to '^' (1000) shifted 'up' one space. The NOT version of this gate is 1011 giving us; '!^U' (NOT AND UP). Similarly, !V shifted down one space gives us 0010, so this is '!VD' (NOR DOWN).

Therefore, '!VD' is equal to 1101. Once again, the Unary Connectives denotes 0010 and 0100, as the NIMP1 and NIMP2, which is identical to Wildberger's notation and more indication that his work is at least derivative. Together with the NOT versions of these two new Logic Operations, we are able to fill 14 of the 16 arrangements in the tables shown in Figure 3.

N	!V	!VD	!R	^U	!L	Δ	!^
0000	0001	0010	0011	0100	0101	0110	0111

^	!Δ	L	!^U	R	VD	V	!N
1000	1001	1010	1011	1100	1101	1110	1111

Fig. 3. All 16 operations with their names and symbols.

In the process, '0000' and '1111' were named: i.e. "N" and "!N" respectively (standing for "NONE" and "NOT NONE"). These can equally be written as "NOT ALL" (NA) and "ALL" (A). Both 'None' and 'A' are already symbols in the Boolean Algebra, so "NONE" (N) and "NOT NONE" (!N) are used instead. Getting to 'N' and '!N' from any of these gates is a simple matter. We only need apply one gate together with its NOT version:

$$((V)\ \Delta\ (!V)) = N$$

The reverse is given by:

$$((!V)\ \Delta\ (!V)) = !N$$

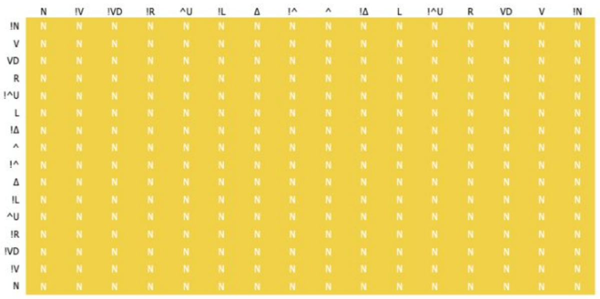
Now all 16 Dimensional Gate Operators and their Boolean Algebra symbols are in place, their relationships can be further explored and they can be written in terms of one another. For example, all these statements are TRUE:

$$((R)\ \Delta\ (L)) = \Delta((!N)\ !\Delta\ (L)) = !L(!VD)\ !\Delta\ (L) = !\wedge$$
$$((\Delta)\ \wedge\ (!L)) = \wedge U$$

This means that the last equation in this list can equally be written as follows:

$$((R)\ \Delta\ (L))\ (!(!VD)\ !\Delta\ !(L))\ ((!N)\ !\Delta\ (L)) = \wedge U$$

To make these relations more easily digestible, the author took the liberty to organize all this information in the following tables shown in Figures 4 to 19 below (the Python code to generate these tables is in Appendix 2).



	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
V	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
R	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!^U	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
L	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!Δ	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
^	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!^	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Δ	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!L	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
^U	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!R	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!VD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
!V	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Fig. 4. N : [0000].

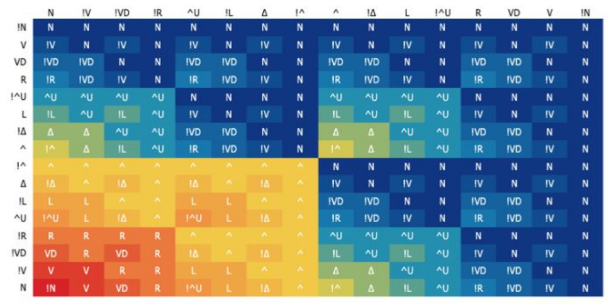


Fig. 5. !V : [0001].

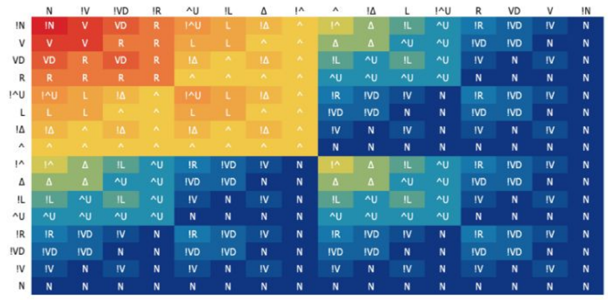


Fig. 6. !VD : [0010].

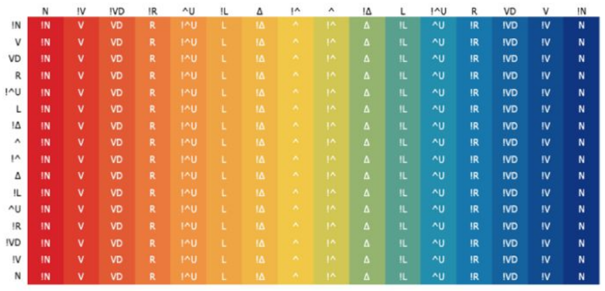


Fig. 7. !R : [0011].

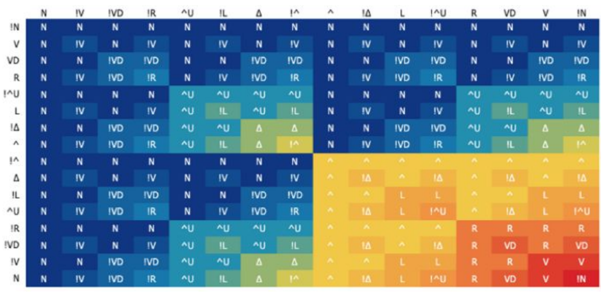


Fig. 8. ^U : [0100].

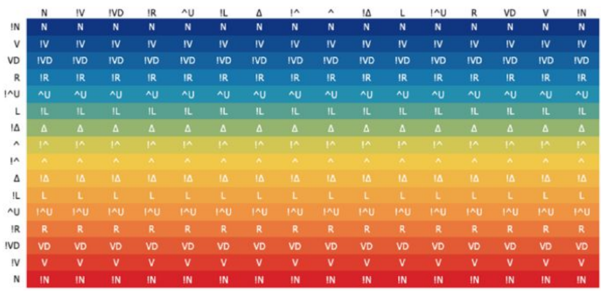


Fig. 9. !L : [0101].

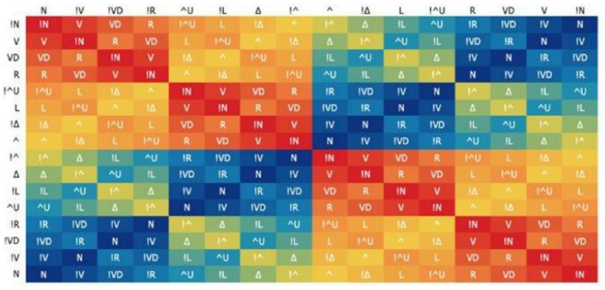


Fig. 10. Δ : [0110].

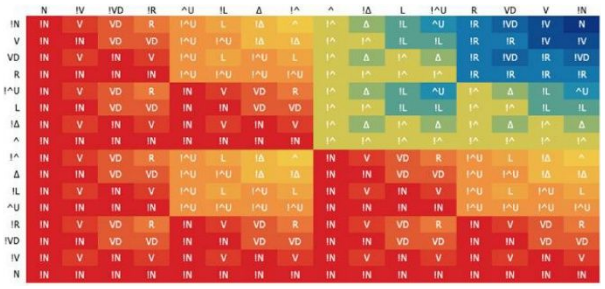


Fig. 11. !^ : [0111].

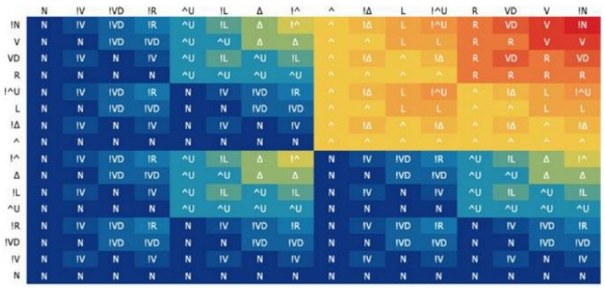


Fig. 12. ^ : [1000].

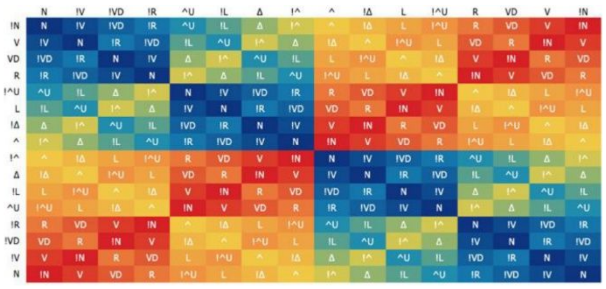


Fig. 13. !Δ : [1001].

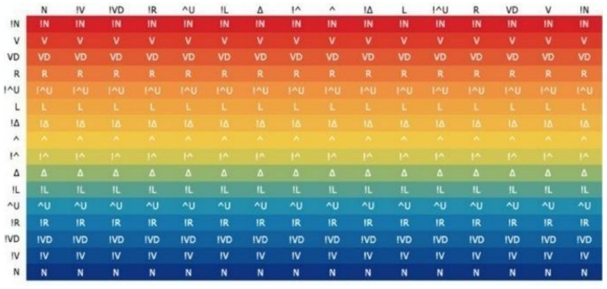
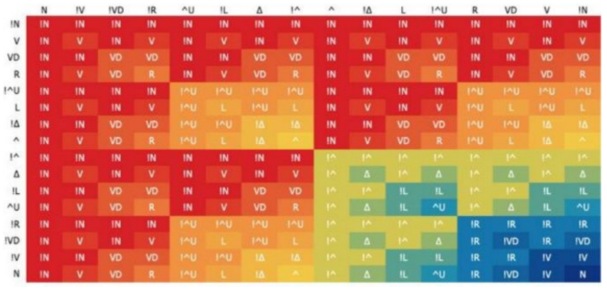
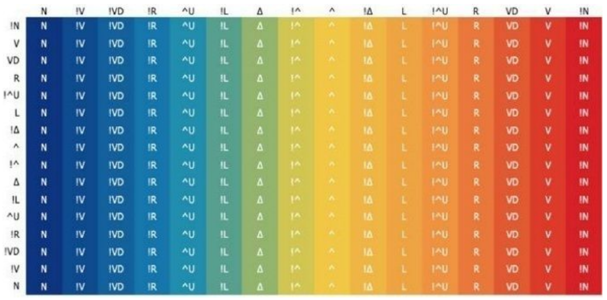


Fig. 14. L : [1010].



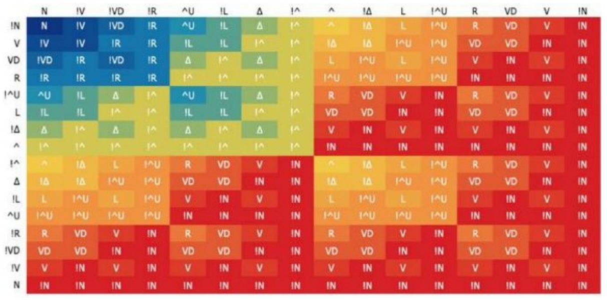
	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V
VD	!N	!N	VD	VD	!N	!N	VD	VD	!N	!N	VD	VD	!N	!N	VD	VD
R	!N	V	VD	R	!N	V	VD	R	!N	V	VD	R	!N	V	VD	R
!^U	!N	!N	!N	!N	!^U	!^U	!^U	!^U	!N	!N	!N	!N	!^U	!^U	!^U	!^U
L	!N	V	!N	V	!^U	L	!^U	L	!N	V	!N	V	!^U	L	!^U	L
!Δ	!N	!N	VD	VD	!^U	!^U	!Δ	!Δ	!N	!N	VD	VD	!^U	!^U	!Δ	!Δ
^	!N	V	VD	R	!^U	L	!Δ	^	!N	V	VD	R	!^U	L	!Δ	^
!^	!N	!N	!N	!N	!N	!N	!N	!N	!^	!^	!^	!^	!^	!^	!^	!^
Δ	!N	V	!N	V	!N	V	!N	V	!^	Δ	!^	Δ	!^	Δ	!^	Δ
!L	!N	!N	VD	VD	!N	!N	VD	VD	!^	!^	!L	!L	!^	!^	!L	!L
^U	!N	V	VD	R	!N	V	VD	R	!^	Δ	!L	^U	!^	Δ	!L	^U
!R	!N	!N	!N	!N	!^U	!^U	!^U	!^U	!^	!^	!^	!^	!R	!R	!R	!R
!VD	!N	V	!N	V	!^U	L	!^U	L	!^	Δ	!^	Δ	!R	!VD	!R	!VD
!V	!N	!N	VD	VD	!^U	!^U	!Δ	!Δ	!^	!^	!L	!L	!R	!R	!V	!V
N	!N	V	VD	R	!^U	L	!Δ	^	!^	Δ	!L	^U	!R	!VD	!V	N

Fig. 15. !^U : [1011].



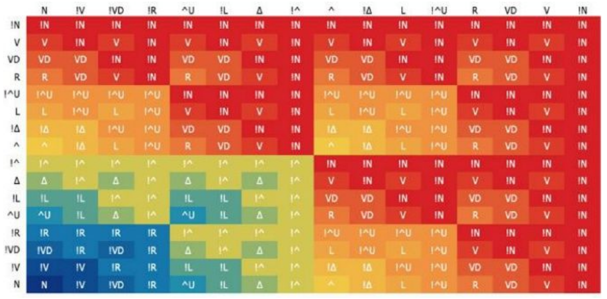
	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!N	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
V	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
VD	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
R	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!^U	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
L	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!Δ	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
^	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!^	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
Δ	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!L	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
^U	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!R	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!VD	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!V	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
N	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N

Fig. 16. R : [1100].



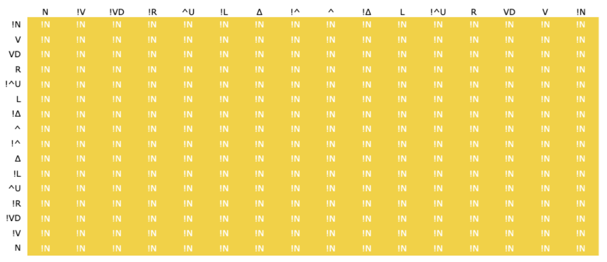
	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!N	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
V	!V	!V	!R	!R	!L	!L	!^	!^	!Δ	!Δ	!^U	!^U	VD	VD	!N	!N
VD	!VD	!R	!VD	!R	Δ	!^	Δ	!^	L	!^U	L	!^U	V	!N	V	!N
R	!R	!R	!R	!R	!^	!^	!^	!^	!^U	!^U	!^U	!^U	!N	!N	!N	!N
!^U	^U	!L	Δ	!^	^U	!L	Δ	!^	R	VD	V	!N	R	VD	V	!N
L	!L	!L	!^	!^	!L	!L	!^	!^	VD	VD	!N	!N	VD	VD	!N	!N
!Δ	Δ	!^	Δ	!^	Δ	!^	Δ	!^	V	!N	V	!N	V	!N	V	!N
^	!^	!^	!^	!^	!^	!^	!^	!^	!N	!N	!N	!N	!N	!N	!N	!N
!^	^	!Δ	L	!^U	R	VD	V	!N	^	!Δ	L	!^U	R	VD	V	!N
Δ	!Δ	!Δ	!^U	!^U	VD	VD	!N	!N	!Δ	!Δ	!^U	!^U	VD	VD	!N	!N
!L	L	!^U	L	!^U	V	!N	V	!N	L	!^U	L	!^U	V	!N	V	!N
^U	!^U	!^U	!^U	!^U	!N	!N	!N	!N	!^U	!^U	!^U	!^U	!N	!N	!N	!N
!R	R	VD	V	!N	R	VD	V	!N	R	VD	V	!N	R	VD	V	!N
!VD	VD	VD	!N	!N	VD	VD	!N	!N	VD	VD	!N	!N	VD	VD	!N	!N
!V	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N
N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N

Fig. 17. VD : [1101].



	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
V	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N	V	!N
VD	VD	VD	!N	!N	VD	VD	!N	!N	VD	VD	!N	!N	VD	VD	!N	!N
R	R	VD	V	!N	R	VD	V	!N	R	VD	V	!N	R	VD	V	!N
!^U	!^U	!^U	!^U	!^U	!N	!N	!N	!N	!^U	!^U	!^U	!^U	!N	!N	!N	!N
L	L	!^U	L	!^U	V	!N	V	!N	L	!^U	L	!^U	V	!N	V	!N
!Δ	!Δ	!Δ	!^U	!^U	VD	VD	!N	!N	!Δ	!Δ	!^U	!^U	VD	VD	!N	!N
^	^	!Δ	L	!^U	R	VD	V	!N	^	!Δ	L	!^U	R	VD	V	!N
!^	!^	!^	!^	!^	!^	!^	!^	!^	!N	!N	!N	!N	!N	!N	!N	!N
Δ	Δ	!^	Δ	!^	Δ	!^	Δ	!^	V	!N	V	!N	V	!N	V	!N
!L	!L	!L	!^	!^	!L	!L	!^	!^	VD	VD	!N	!N	VD	VD	!N	!N
^U	^U	!L	Δ	!^	^U	!L	Δ	!^	R	VD	V	!N	R	VD	V	!N
!R	!R	!R	!R	!R	!^	!^	!^	!^	!^U	!^U	!^U	!^U	!N	!N	!N	!N
!VD	!VD	!R	!VD	!R	Δ	!^	Δ	!^	L	!^U	L	!^U	V	!N	V	!N
!V	!V	!V	!R	!R	!L	!L	!^	!^	!Δ	!Δ	!^U	!^U	VD	VD	!N	!N
N	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N

Fig. 18. V : [1110].



	N	!V	!VD	!R	^U	!L	Δ	!^	^	!Δ	L	!^U	R	VD	V	!N
!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
V	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
VD	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
R	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!^U	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
L	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!Δ	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
^	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!^	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
Δ	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!L	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
^U	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!R	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!VD	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
!V	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N
N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N	!N

Fig. 19. !N :[1111].

The basic rules of arithmetic; like addition, subtraction and/or multiplication and division can easily be applied to these gates. For instance, R Δ L can be rewritten to produce (!^)) as follows:

R Δ L = Δ
(R) Δ (Δ) = ((VD) Δ (!^))
(R) Δ (Δ) / ((VD) Δ) = (!^)

Another more convoluted example, this time rewriting ((Δ) ^ (!L)) for (VD) is:

(Δ) ^ (!L) = ^U
((Δ) ^ (^U)) V ((Δ) V (^U)) = ^U
!(((Δ) ^ (^U)) V ((Δ) V (^U))) = !^U
(!Δ)^(!^U) V (!Δ)V(!^U)/(!Δ) ^ = VD

Later, these methods will be applied to the Set Theory, as a whole.

Some notable gates

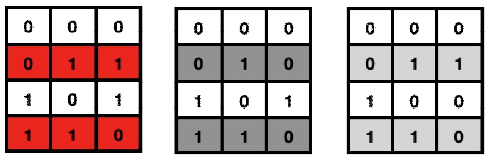
In the previous research paper, operators like Δ (XOR) and !Δ (XNOR) were both encountered (O'Neill, 2021). Each of these represents the rules of Real Number Arithmetic and Complex Arithmetic, according to the DGO. This shift in how to do arithmetic transports us from the ostensibly 1-2 dimensional lines of the Real Numbers and into the 2-3 dimensional realm of the 'Complex Plane'.

Unlike the numbers of the Complex Plane, however, there are no algebraic numbers here. This is because (–1)1/2 (the imaginary unit, i) is equal to ±1 in !Δ. Therefore, they are simply the ordinary numbers with a different arithmetic rule set. The reader might be tempted to think that Δ also governs the rules of the Quaternions and Octonions (Hamilton, 2000). However, this is not the case.

One important feature of the Quaternions is that they are non-commutative; that is A(B) =/= B(A). This attribute is seen in the DGO in places like !VD, !R, !^U, and VD, where A(-B) =/= B(-A). Quaternions are governed by a mix of !VD and ^U logic, because they are non-commutative for different signed quaternions, whilst retaining the !Δ rule set of the imaginary numbers, when the signed values for i, j, k are the same:

i2 = j2 = k2 = – 1
ij = k, ji= – k
jk = i, kj = – i
ki = j, ik = – j

The Quaternions are made from two groups of !VD and one of ^U. This asymmetry allows for the loop to be closed and further explains why these higher-dimensional algebras can only be order 2^n. Below there are three tables that correspond to the following (left to right): Δ, ^U, and !VD.



0	0	0
0	1	1
1	0	1
1	1	0

0	0	0
0	1	0
1	0	1
1	1	0

0	0	0
0	1	1
1	0	0
1	1	0

Dimensional spaces

Returning to Figures 4 to 19 (i.e. sets 0 to 15), it is apparent all of them can fit together like jigsaw pieces. For instance, it is clear that Figures 5, 6, 8, and 12 (i.e. sets 1, 2, 4, and 8) form a set that can be called 'Set B'. Similarly, we can make 'Set C' from Figures 11, 15, 17, and 18 (i.e. sets 7, 11, 13, and 14). The horizontal and vertical lines appear to make another set, 'Set D' (Figures 7, 9, 14, and 16). That just leaves the final four graphs: Figures 4, 10, 13, and 19, namely set {0, 6, 9, 15} or 'Set A'. All the sets are shown schematically in Figure 20. There is a distinct pattern emerging in how these sets are dispersed across the entire binary number line, here represented in their decimal form.

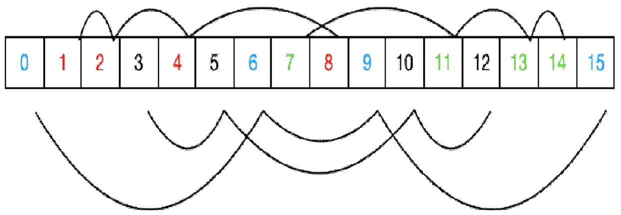


Fig. 20. All four sets: A (blue numbers), B (red), C (green), and D (black). This pattern explains the frequency of the NOT values, as they are represented in the DGO.

The first composite graph (Figure 21) shows the domains of N (in the bottom left-hand corner), Δ immediately above that, !Δ in the lower righthand corner, and !N in the top right. What matters here is not the order, so much, as the grouping and what it reveals about the connectivity of the Logical Connective space. Set A can be rewritten as {N, Δ, !Δ, !N}, which stands for the paths between Dimensions 0, 2, 3, and 1, respectively. To understand this, look at our Quaternion logic gates: ^U and !VD. Figure 22 shows Set B.

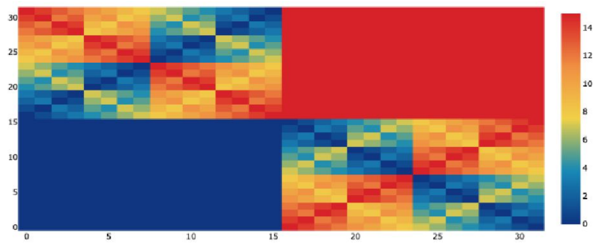


Fig. 21. Set A = {0, 6, 9, 15} or A = {N, Δ, !Δ, !N}.

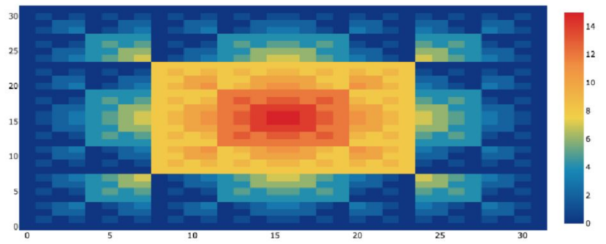


Fig. 22. Set B = {1, 2, 4, 8} or B = {!^, !^U, !VD, !V}.

If we multiply these terms by ¬Δ (the Imaginary rule set) we obtain (!VD) ¬Δ (!^U) = Δ (our Real number logic). This shows the path by which Quaternions collapse down into the rule set of both the Imaginary and Real numbered spaces of dimensions 2-3. We can then continue this process, in the usual manner, multiplying (Δ) !Δ (!Δ) and (Δ) Δ (!Δ) to obtain the dimensions beneath them as follows:

$$(\Delta)\ !\Delta\ (!\Delta) = N$$
$$(\Delta)\ \Delta\ (!\Delta) = !N$$

One possible conclusion is that N and !N refer to dimensions 0 and 1, respectively. In this sense, the Real Numbers and Imaginary Numbers live in Set A, along with dimensions 1 and 0. Whereas the Quaternions live jointly in Sets B and C, along with some other more traditional logic gates. The Octonions, Sedenions, and (potentially) other higher dimensional spaces exist scattered among the other sets, although this is something that must be investigated further.

While this way of thinking about logic gates can provide a method for travelling from one rule set (i.e. dimensional space) to the other, it should not be taken too literally. In one sense, applying the Δ or !Δ rule set to the Quaternions does not lead to Δ or !Δ but rather to hybrid spaces; the Real Quaternions and the Imaginary Quaternions, and neither of these two systems cancel out to Dimension 0 when summed in Δ. More on this in up-coming research.

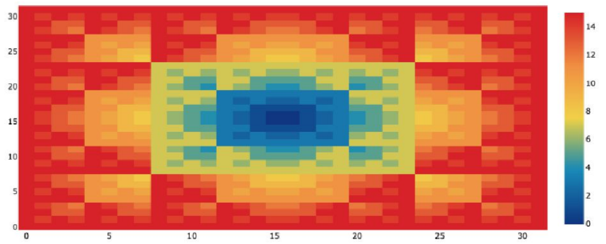


Fig. 23. Set C = {7, 11, 13, 14} or C = {V, VD, ^U, ^}.

Based on the arrangements of the graphs in Figure 21, it is possible to see the placement beginning in the lower left-hand corner and proceeding in a zigzag fashion to the upper-right. The graphs in Figure 22 go in the reverse direction and the pattern is repeated in the next two graphs in Figures 23 and 24.

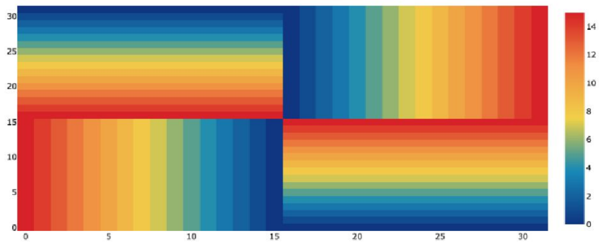


Fig. 24. Set D = {3, 5, 10, 12} or D = {!R, !L, L, R}.

Using the pattern of Figure 21 as the 'base arrangement', Figures 21, 22, 23, and 24 can be arranged into a single graph (Figs. 25, 26, and 27). But note there are 16! possible arrangements, so care must be taken when choosing an arrangement.

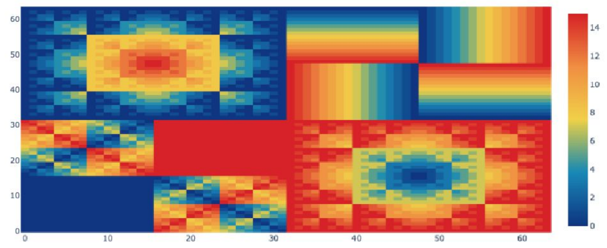


Fig. 25. The preliminary grouping of the sets.

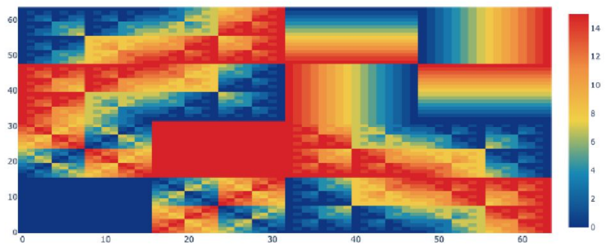


Fig. 26. The final grouping.

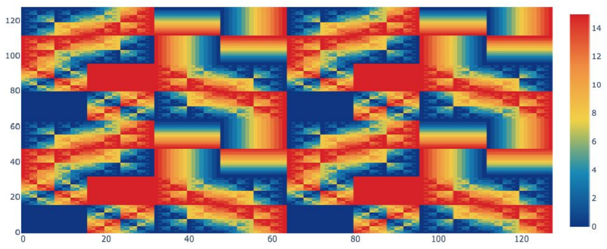


Fig. 27. A more tiled version of Figure 26.

It can be deduced from the lack of connectivity between the different regions that a more accurate arrangement is possible. We shouldn't expect to see such harsh delineating lines from the interrelated sets. Earlier it was stated that the Quaternions and the Imaginary Quaternions follow the rules laid out in Set B, while the Real Quaternions are sitting in Set C, which obviously cannot be right. If the Real Quaternions are moved into Set B with the other Quaternions, this forces all the commonly known logic gates: ^, !^, V, and !V together, which is a much neater result.

Moreover, there is a much greater connectivity between the different regions. Given that this has served as a useful vehicle to explain some of the relationships between the Reals, Imaginary, Quaternions, and Logic Gates, it may be worthwhile exploring this line of inquiry further.

Set Theory Arithmetic

It is possible (at least in principle) to divide and multiply set operators together. And it is also possible to apply these kinds of operations to the actual sets of Set Theory. Unlike traditional methods of arithmetic with sets, where (for instance) the elements of one set are divided into another, here the operators acting on the sets will be divided.

Using the DGO methods on Set Theory will advance new ways to get from one partition of a group of sets to another and will lead to operations, which were not previously possible under the former laws of the Boolean Algebra.

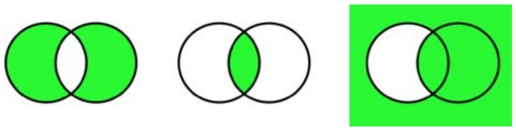
To begin with, it is not clear why this should this be so. After all, the current operators already cover all aspects and combinations of Set Theory. But when it is recalled that the !VD operator governs the arithmetic of the quaternions, this portends the possibility of doing Set Theory in higher order dimensions.

Quaternions are known to be indispensable for describing some aspects of Quantum Mechanics, and these extra Logic Gates simply extend the number of dimensions available to work with. This also forms the basis of the Dimensional Gate Operator Standard Model, which is further developed in (C. O'Neill, "Dimensional Gate Quaternion Multiplication, Quarks & Polyhedra" DOI: 10.13140/RG.2.2.22968.57601/1; "Construction of the 2nd and 3rd Generation Quark Particles in the Standard Model" DOI: 10.13140/RG.2.2.20228.35202, as well as the overview: "Making Sense of the Standard Model" DOI: 10.13140/RG.2.2.34132.12163/1). Essentially, these operators can be used to describe sets of these particles via the rules of the 4-dimensional space (4D space) in which they live. Alternatively, 4D Set Theory might find application in the perplexing world of Quantum Computing, where a bit can be a '1' and '0' at the same time.

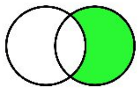
To begin with, a simple example with sets A and B is as follows:

$$(A \Delta B) \wedge (A \,!L\, B) = (A \wedge^{U} B)$$

This operation would correspond with the following Venn Diagrams:

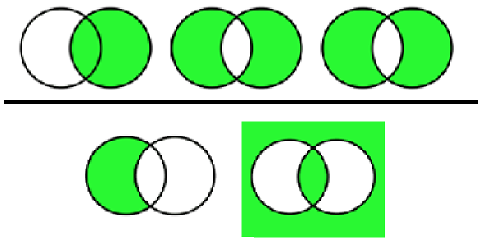


and equals to:

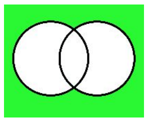


A slightly more complex operation would be the following:

$$(A\ R\ B)\ \Delta\ (A\ !\Delta\ B)\ /\ (!VD)!\Delta = (A\ !V\ B)$$



This equals to



An important aspect of the above two equations is they both feature noncommutative logic gates, specifically: !L, R, and !VD. This means that different outcomes for the equation would be expected depending on which set we choose to be on the right and which appears on the left.

Since sets have no orientation in space and since either one can be on the 'left' or on the 'right', this poses something of a problem, especially as the set arithmetic moves into higher dimensions. Or at least it would be a problem, if it were expected to form a closed space algebra, but as we shall see in the follow-up preprints and research papers, creating closed field algebras might not always be the best approach.

CONCLUSION

The Dimensional Logic Gate Operators have the potential for some very unusual and unexpected applications. They can find application in computer circuitry, set theory, and Boolean algebra, as well as Quantum Computing, 4-dimensional (and higher) Set Theory, and Set Theory Arithmetic.

REFERENCES

Bocheński, JM. 1948. Précis de logique mathématique. F.G. Kroonder, Bussum, North Holland.

Boole, G. 1847. The Mathematical Analysis of Logic, Being an Essay Towards a Calculus of Deductive Reasoning. Cambridge: MacMillan, Barclay, and MacMillan; London: George Bell, UK. pp 82.

Hamilton, WR. 2000. On Quaternions, or on a new system of imaginaries in algebra. The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science. XXV-XXXVI (3rd Series). pp 92. (Hamilton, WR. 1847. XXXVI. On Quaternions, or on a new system of imaginaries in algebra. Proceedings of the Royal Irish Academy. Philosophical Magazine Series 3. 31(207):214) DOI: https://doi.org/10.1080/14786444708645826.

O'Neill, CC. 2021. Reimagining complex numbers. Canadian Journal of Pure and Applied Sciences. 15(2):5261-5268. DOI: https://doi.org/10.13140/RG.2.2.26666.44480/1.

Roegel, D. 2002. A brief survey of 20th century logical notations. [Research Report] LORIA – Université de Lorraine, France. ffhal-02340520 (HAL Id: hal-02340520) https://hal.inria.fr/hal-02340520.

Received: June 28, 2021; Revised: August 18, 2021;
Accepted: Sept 30, 2021

Appendix 1.

The 16 tables.

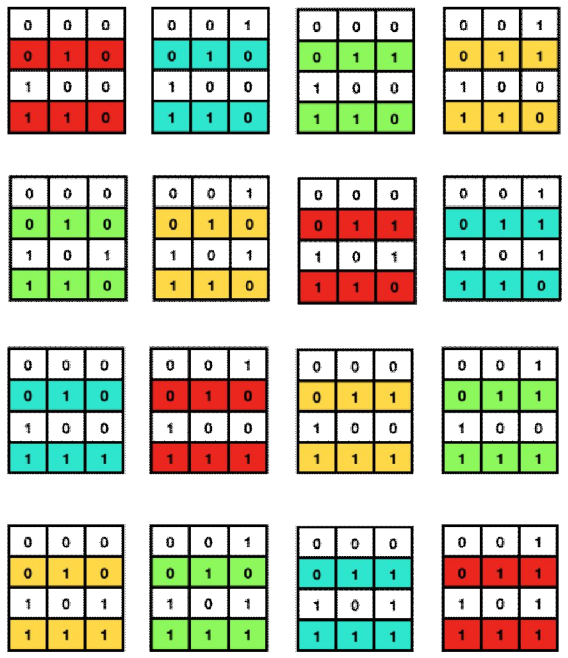


Fig. A1.1. From top left: N, !V, !VD, !R, ^U, !L, Δ, !^, ^, !Δ, L, !^U, R, VD, V, !N. The colour-coding here pertains to the different sets: A = Red, B = Green, C = Blue, and D = Yellow. The Quaternions are created from !VD and ^U and the Real Quaternions are created from !^U and VD.

Appendix 2.
The Python codes for the generation of Figures 4 to 19.

```
import plotly
import  plotly.graph_objs as go
import plotly.figure_factory as ff
import numpy as np
import copy

gates = {'0000':'N', '0001':'!V', '0010':'!VD', '0011':'!R',
'0100':'^U', '0101':'!L',
'0110':'Δ', '0111':'!^', '1000':'^', '1001':'!Δ', '1010':'L',
'1011':'!^U',
'1100':'R', '1101':'VD', '1110':'V', '1111':'!N'}

logic = []
for i in range(16):
   t = str(bin(i)[2:])
   c = len(t)
   j = 4
   while j<= 4:
      g = 4 - c
      k = t.rjust(j, '0')
      logic.append(k)
      j += 1

test1 = []
for i in logic:
   gg = []
   f = list(i)
   gg.append(f)
   test1.append(f)

colourscales = ['Greys', 'YlGnBu', 'Greens', 'YlOrRd',
'Bluered', 'RdBu','Reds', 'Blues', 'Picnic', 'Rainbow',
'Portland', 'Jet','Hot', 'Blackbody', 'Earth', 'Electric',
'Viridis', 'Cividis']

lg = ['N', '!V', '!VD', '!R', '^U', '!L', 'Δ', '!^', '^', '!Δ', 'L',
'!^U', 'R', 'VD', 'V', '!N']

check = []
balance = []

for f in logic:

   m = list(f)

   blogic = []
   for i in logic:
      for c in logic:
         u = 0
         r = copy.copy(i)
         r0 = list(r)
         while u <= 3:
            if i[u] == '1' and c[u] == '1':
               r0[u] = m[0]
            elif i[u] == '0' and c[u] == '1':
               r0[u] = m[1]
            if i[u] == '1' and c[u] == '0':
               r0[u] = m[2]
            elif i[u] == '0' and c[u] == '0':
               r0[u] = m[3]
            u+=1
         r = ''.join(r0)
         blogic.append(r)

   state = []
   for i in blogic:
      state.append(gates[i])

   states= np.array(state).reshape(16, 16)


   po4=[]
   for i in blogic:
      kk = int(i, 2)
      po4.append(kk)

   po3 = np.array(po4).reshape(16, 16)

   po5 = states.tolist()
   check.append(po5)
```

```
    po6 = po3.tolist()
    balance.append(po6)

    fig = ff.create_annotated_heatmap(z = po3, x=lg, y=lg,
annotation_text=states,colorscale=colourscales[10])
    fig.update_layout(title_text=gates[f] + " " + ":" + " " +
"[" + f + "]", title_x=0.5,
              titlefont= {"size": 14},
                font={'color':'black'},
                paper_bgcolor= 'white',
                plot_bgcolor= "white",
                hovermode='closest',)
    fig.show()
```