A Treatise on the Fundamentals of Ternary Arithmetic and Logic

The definition of ternary arithmetic gates and systems

Etaash Katiyar
Gurgaon, India
etaashkatiyar@gmail.com

Abstract— This paper is part of a larger project to codify a consistent, practical and useful framework of ternary functions (ternary logic gates). This paper will introduce a functional framework for ternary computations – computation system involving 3 discrete values as compared to the modern day standard of 2 (binary). The author will go through certain fundamental gates including the parallels of common binary gates and introduce new gates which are possible in the ternary framework. The applications of certain gates in cryptographically secure (Shannon-secure) systems will be discussed.

Keywords- ternary; base-3; logic-gates; computation;

I. INTRODUCTION

While computational technology advances exponentially following Moore's law, a hard ceiling is incumbent on technology when the transistor becomes small enough for quantum tunneling to severely impact accuracy and reliability. Quantum computing has been devised to allow for computers to increase their capacity, however such logical frameworks only work on certain algorithms and offer little to no gain on other commonplace algorithms. Quantum computing also has the drawbacks of being new and untested as compared to more classical electronic circuits as well as the high maintenance costs it has. Ternary based logic will allow a tremendous boost in computational power while operating on the same underlying physical principles.

Utilizing more than 2 bits offers more efficient computation and storage, scaling by a factor of 3^n instead of 2^n as offered by binary – an exponential gain in computing power. The ternary system does not have a logical framework such as Boolean algebra, which is what this paper will explore. This paper attempts to build a complete system of logic gates and explore some of the more elementary applications of the technology.

The scope for a ternary base computational system is massive, granting an exponential boost in storage and computation. By moving forward from the industry standard of Boolean logic and binary, we can surpass the system's limitations.

II. FOUNDATIONS

A. Difference from Boolean

While Boolean logic is built on top of predicate logic systems where a variable can be assigned a value of 'true' or 'false', ternary based computation does not have to be tied down by similar (albeit useful) restrictions. Instead, the author proposes a system where a variable can take any of the values in the set $\{0,1,2\}$. This is the natural extension from the numerical space of binary, which is $\{0,1\}$. The values do not represent the truth of a variable but are simply integers. This allows us to manipulate the numbers as integers, which is often more practical for computation. The digits are purely mathematical as to provide a more robust and streamlined approach to mathematical and computational needs.

A single unit in binary is called a bit (**bi**nary un**it**). We will extrapolate and use the word '**trit**' to denote a ternary unit, representing a value in $\{0,1,2\}$.

B. Base-three Notation

Before working in ternary, we must familiarize ourselves with the number base of 3. In everyday life we commonly use base 10, a byproduct of our society, which contains digits ranging from 0 through 9. Depending on the placement of the digit in the number, it is assigned a value:

$$4532_{10} = 4 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 2 \times 10^0$$

In ternary notation we use base three. Consequently, the digits range from 0 through 2 and are multiplied by an exponent of three based on their position in the number, illustrated below:

$$21_3 = 2 \times 3^1 + 1 \times 3^0$$

The subscript after a number denotes the base. In this paper we will use this notation. When a subscript is not present, the number is assumed to be in base ten or a self-standing digit.

C. Logic Gates

While a truth table is often the best way to represent binary functions, we will adopt a different approach for ternary. The inputs and corresponding outputs in binary are mapped in a 1-dimensional lookup table. While this is possible to emulate for the purposes of the paper, a truth table becomes cumbersome (listing out 9 values instead of 4 for a 2 bit/trit operation) and the properties of the function are much more evident when using a 2-dimensional lookup table for a function with two inputs and 1 output.

a\b	0	1	2
0	$x_{0,0}$	$x_{0,1}$	$x_{0,2}$
1	$x_{1,0}$	$x_{1,1}$	$x_{1,2}$
2	$x_{1,0}$	$x_{1,1}$	$x_{1,2}$

This way of representing a 2-input and 1-output function is clearer to use when dealing with ternary values. $x_{a,b}$ in the diagram above represents the output for in input of a and b. The functions introduced in the paper will follow this representation.

D. Information and Language

On a side note, using ternary to encode the English alphabet yields significant reductions in the amount of digits needed. By using three trits, we have a total of 27 unique identifiers. This can represent the letters from a to z as well as a character for a space. This by coincidence is an extremely efficient way of storing raw text and is an unexpected but welcome improvement that ternary offers over binary, which would have wasted 5 identifiers (the closest power of two being 32).

III. TERNARY LOGIC

A. Choosing Logic Gates

There are of course infinite possible logic gates, hence I will start from the logic gates with 1 input and output, then the gates with 2 inputs and 1 output as well as discuss a gate with arbitrary inputs and 1 output (which we can consider to be more fundamental).

B. One-to-one gates

There are a total of 27 possible logic gates with 1 input and 1 output. Diagram 2 is a listing of all possible permutations:

Input	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***16***	***17***	***18***
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	2	2	2	0	0	0	1	1	1	2	2	2
2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

19	***20***	***21***	***22***	***23***	***24***	***25***	***26***	***27***
2	2	2	2	2	2	2	2	2
0	0	0	1	1	1	2	2	2
0	1	2	0	1	2	0	1	2

Many of these logic gates are unremarkable. A few notable ones are:

1. Absolute Value gates (equivalents of tautology/contradiction in binary logic): Gates which output only one value, namely the gates at index 1, 14, 27. There isn't much purpose to give these gates a name since they can just be represented by the number that the output; a recommended notation being the corresponding output within a circle.
2. Increment functions: We can define two increment functions, INC1 and INC2, which increment the value by 1 and 2 respectively and loop back round to 0 when passing 2. INC1 is at index 16 and INC2 is at index 20 (INC0 also exists but is redundant).

INC1

0	1
1	2
2	0

INC2

0	2
1	0
2	1

3. Flip gate. There are three gates which flip the inputs “around” one value which remains fixed. This is equivalent to swapping 2 of the output values while keeping the 3rd the same. There are FLP0, FLP1 and FLP2.

FLP0

0	0
1	2
2	1

FLP1

0	2
1	1
2	0

FLP2

0	1
1	2
2	0

C. Two-to-One gates

There are a total of $3^9 = 19683$ two-input-one-output gates; hence we cannot use the same strategy as listing all of them and picking the ones which are interesting. Instead, they are chosen based on their utility, offering the system comprising of the gates the maximum flexibility and utility

1. Equality

The equality gates are gates which represent a certain value when the two inputs are equal and 0 when the inputs are the same. EQU is defined as follows:

EQU			
a\b	**0**	**1**	**2**
0	0	0	0
1	0	1	0
2	0	0	2

2. Min-Max.

The MIN and MAX gates are a set of gates which output either the minimum or the maximum of the inputs. Minimum can be thought of as an interpretation of the AND gate and Maximum as an interpretation of the OR gate. These gates can be generalized to more than 2 inputs, where the output is the lowest or the highest among the input values.

MIN			
a\b	**0**	**1**	**2**
0	0	0	0
1	0	1	1
2	0	1	2

MAX			
a\b	**0**	**1**	**2**
0	0	1	2
1	1	1	2
2	2	2	2

3. TNOR.

Ternary NOR gate is a gate which serves as the universal gate in our model. A universal gate is a gate which can emulate any possible ternary output with an arbitrary numbers of inputs and output through only the use of the universal gate (the discussion of universality is beyond the scope of this paper, however, the author intends to cover it in the subsequent publication). TNOR can be considered a generalization of NOR from binary which is also universal. TNOR is equal to 0 if the inputs are different and equal to INC2(a) if the values are the same:

TNOR			
a\b	**0**	**1**	**2**
0	2	0	0
1	0	0	0
2	0	0	1

4. TXOR and TOX.

Ternary XOR is a cryptographically secure gate. TXOR is remainder of the sum of the two inputs after division by 3 incremented by 1. The gate can be generalized to an arbitrary number of inputs. TXOR is equal to the sum of all the inputs mod 3 (remainder after division by 3). One can remark how creating a cryptographically secure logic gate in this representation is not unlike solving a Sudoku grid.

TXOR			
a\b	**0**	**1**	**2**
0	1	2	0
1	2	0	1
2	0	1	2

TOX takes another approach to consolidate a larger number of inputs, ultimately leaving a differentl structured gate. The gate is not cryptographically secure, however, it defines an interesting pattern. The gate is defined similar to the TXOR but giving alternating positive and negative weights to the inputs. Where t is the output and a_n is the nth input:

$$t = a_1 - a_2 + a_3 - a_4 \dots \pm a_k \bmod 3$$
$$t = \left(\sum_{n=1}^{k} a_n \, (-1)^{n+1} \right) \bmod 3$$

The truth table for the gate for 2 inputs is (first input is ‘a’):

TOX			
a\b	**0**	**1**	**2**
0	0	1	2
1	1	0	1
2	2	1	0

Over the course of this whitepaper we have defined a number of logic gates and a necessary framework for a possible computation system in ternary. The various gates can see their application in a number of algorithms. The logical framework necessary to delve into ternary is what the paper has explored and presented, constructing a coherent and universal system which can constitute any possible gate and therefore lend itself to plethora of practical uses. Symbolically it is equivalent to a turing machine due to the universality of the TNOR gate, however to avoid unnecessary complications when dealing with ever increasing complexity in algorithms based on these gates the other gates are defined.

Further research on this topic can shed light onto the exact usage of the gates in useful algorithms such as a full adder or eventually a ternary based hash. The universality and construction of the more fundamental of these algorithms will feature in the 2nd part of this multi-paper project by the author. Our future work is much simplified by creating a logical and readily understandable framework for ternary based computational systems.

ACKNOWLEDGMENTS

[1] Afolayan A. Obiniyi, Ezugwu E. Absalom, Kwanashie Adako, “Arithmetic Logic Design with Color-Coded Ternary for Ternary Computing”, International Journal of Computer Application Issues, Vol. 26, No. 11, July 2011.

[2] Craig Gidney, “Exploring Universal Ternary Gates”;

[3] “http://twistedoakstudios.com/blog/Post7878_exploring-universal-ternary-gates”

[4] Jorge Pedraza Arpasi, “A Brief Introduction to Ternary Logic”. November 2003.

[5] Ion Profeanu, “A Ternary Arithmetic and Logic,” 2010.