

A Comparative Analysis of the Performance of Emulated Ternary vs Binary Logic Systems

Literature, Technology and Data Survey

Thomas Timmons
Bachelor Hons. Computer Science
tlt33@bath.ac.uk

December 4, 2024

1 Introduction

Binary Logic has been the defacto standard for digital systems since the inception of the transistor. However, due to the imminent end of Moore's Law, researchers have been looking for alternative methods to increase the performance of digital systems. This paper will attempt to realise the theoretical performance gains of ternary logic within an emulator. We can then measure and project the performance on the assumption it should be possible to implement these circuits as hardware.

2 Literature and Technology Survey

James' Feedback

Technology Survey:

Ways to implement emulators in software.

Review Tools and explain why implementation by yourself is more comparable.

This report will provide a comprehensive review of the literature and technology surrounding the following topics:

- Ternary Algebra
- Ternary Logic Gates
- Previous Ternary Implementations
- Emulation of Circuits

Each of these areas will be explored in detail while assessing the credibility and motivation for the work. The report will aim to build the foundation on which the project will be based.

2.1 Ternary Algebra

The cost or complexity of number systems can be evaluated through representation efficiency, computational overhead, and implementation requirements.

Representation efficiency refers to how digits or symbol are required represent any given range of values N , with the base of the number system represented by R and the necessary number of digits required is d (rounded upwards to the nearest integer):

$$N = R^d \quad (1)$$

This directly impacts the memory usage and storage costs in a computing system.

The computational overhead involves the complexity of performing basic arithmetic operations. Larger bases require the managing of more states, requiring the processing of a larger set of possible combinations. The number of possible combinations is given by:

$$C = R^i \quad (2)$$

Where i is the number of inputs for the operation and R is the base of the number system as defined before. For example, the number of possible combinations for addition is $C = R^2$ which in binary is $C = 2^2 = 4$ creating the set $\{0 + 0, 0 + 1, 1 + 0, 1 + 1\}$. In ternary the set would be $C = 3^2 = 9$ which requires implementing more logic circuits to handle the additional combinations.

Implementation requirements consider the the practical aspects of building a system that works with a given base. This includes error detection and correction using approaches such as Hamming Code's in binary logic As this paper aims to explore the performance of ternary logic systems in comparison to binary systems within an emulator, we will not be considering the practical challenges of implementing ternary logic in hardware.

TODO: understand how this logic below impacts the performance of the ternary logic system.

The cost or complexity of the system C is directly proportional to the capacity of the digits $(R \cdot d)$. Therefore for some constants k and $\log N$

$$C = k(R \cdot d) = k \left(R \cdot \frac{\log N}{\log R} \right) = k \log N \left(\frac{R}{\log R} \right) \quad (3)$$

Minimising this cost C can be done by differentiating with respect to R and setting the result to 0:

$$\frac{\partial C}{\partial R} = k \log N \cdot \frac{d}{dR} \left[\frac{R}{\log R} \right] = k \log N \cdot \frac{\log R - 1}{(\log R)^2} = 0 \quad (4)$$

After removing constants $\log R = 1$ and solving for $R = e = 2.718$. Since as stated above, the radix R must be an integer then we are left with $R = 3$. Therefore, the ternary logic system is the most efficient.

In binary logic set of possible values is denoted as $\mathbb{B} = \{0, 1\}$. Ternary numeral systems can either unbalanced ternary values $\mathbb{T} = \{0, 1, 2\}$ or balanced ternary values $\mathbb{T} = \{-1, 0, 1\}$ with a variety of glyphs used to represent the values where $\bar{1}$ or T is sometimes used to represent -1 .

Balanced ternary has the elegant property that of representing negative numbers intrinsically without the need for a separate signed bit. This leads to more symmetric carry rules due to a more even distribution around zero as shown in Figure 1.

+	T	0	1
T	$T1$	T	0
0	T	0	1
1	0	1	$1T$

−	T	0	1
T	0	T	$T1$
0	1	0	T
1	$1T$	1	0

×	T	0	1
T	1	0	T
0	0	0	0
1	T	0	1

÷	T	1
T	1	T
0	0	0
1	T	1

Figure 1: Balanced Ternary Single-Trit Addition, Subtraction, Multiplication and Division Tables

For subtraction and division which are not commutative, the first operand is given to the left of the table and the second at the top.

Balanced ternary has a range of advantages when it comes to computation over binary logic such as the reduction in the carry rate for addition/subtraction cuts down the carry rate in multi-digit multiplication as demonstrated in Figure 1 with the values being $2/9$ and $1/4$ respectively. Also notice that it is possible to perform subtraction by negating the second operand and adding the two values together which will be useful for the implementation of the ternary logic gates.

Therefore, for this paper and emulator we will be using balanced ternary logic and any mention of ‘ternary logic’ is referring to ‘balanced ternary logic’.

To calculate the value of a ternary number T with a radix point in decimal we can use the following formula:

TODO: this is not right and needs to be fixed.

$$v = (a_n a_{n-1} \cdots a_0 . c_1 c_2 c_3 \cdots)_3 = \sum_{i=0}^n a_i 3^i + \sum_{i=1}^{\infty} c_i 3^{-i} \quad (5)$$

Where n and m are the number of digits to the left and right of the radix point respectively and v is the value in decimal given a vector of values containing a radix point.

To directly compare between the binary and ternary systems we need to consider how we can encode binary values into ternary values and vice versa. Using continued fractions, the best approximations to rationals are given by convergents to the continued fraction, alternating between overshooting and undershooting the target value.

Since we are looking for $2^a \approx 3^b$ where $a, b \in \mathbb{N}$. Taking the logarithm of both sides we get:

$$\frac{a}{b} \approx \frac{\log 3}{\log 2} \quad (6)$$

Where $2^a < 3^b$ to ensure that the binary value can be encoded into a ternary value.

2.2 Ternary Logic Gates

Gates can be represented in matrix form where the following Binary AND truth table has the equivalent matrix representation:

$A \times B$	0	1
0	0	0
1	0	1

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 2: Binary AND Gate Truth Table and Matrix Representation

2.2.1 Single-Input Gates

Single-input gates can be represented as a vector of length 2 for binary logic and length 3 for ternary logic. Combination theory shows that the number of possible combinations for a single input gate is n^n where n is the number of inputs. Therefore, the number of possible combinations for a single input gate is $2^2 = 4$ and $3^3 = 9$ for binary and ternary logic gates. Most of these are redundant so only the useful gates have been listed in Figure 4.

Matrix	Description	Name	Symbol
$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	Identity, Buffer, Pass	Buffer	A
$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	Inverter	NOT	\overline{A}

Figure 3: Useful Single Value Binary Gates

Matrix	Description	Name	Symbol
$\begin{bmatrix} T \\ 0 \\ 1 \end{bmatrix}$	Identity, Buffer, Pass	BUF	A
$\begin{bmatrix} 1 \\ 0 \\ T \end{bmatrix}$	Inverter	NOT	\bar{A}
$\begin{bmatrix} 1 \\ 1 \\ T \end{bmatrix}$	Positively Biased Inverter	PNOT	\hat{A}
$\begin{bmatrix} 1 \\ T \\ T \end{bmatrix}$	Negatively Biased Inverter	NNOT	\check{A}

Figure 4: Useful Single Value Binary and Ternary Gates

2.2.2 Two-Input Gates

Matrix	Description	Name	Symbol
$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	And	AND	$A \times B$
$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$	Or	OR	$A + B$
$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Exclusive Or	XOR	$A \oplus B$

Figure 5: Useful Two-Input Binary Gates

All of the gates in Figure 5 and 6 have inverted equivalents that can be created by negating the output will be denoted with an N proceeding the gate name. e.g. NAND, NOR, NANY.

2.2.3 Assumptions

Researchers have attempted to investigate this field using two distinct approaches. One from a more theoretical perspective or looking at the theoretical gains of ternary logic while others taking a more practical approach. Ternary test benches have been created in order to attempt to validate the theoretical performance gains.

However, the underlying implementations of the basic ternary logic is underpinned by Complementary Metal Oxide Semiconductor (CMOS) along with it's PMOS and MNOS. Thus the fundamental reliance on the binary nature of the underlying hardware limits the performance gains that can be achieved. For this project we will assume that comparable binary and ternary logic gates have the same propagation delay. This will allow us to directly compare the performance of the two systems. Implications of this assumption will

Matrix	Description	Name	Symbol
$\begin{bmatrix} T & T & T \\ T & 0 & 0 \\ T & 0 & 1 \end{bmatrix}$	And/Minimum	AND	$A \times B$
$\begin{bmatrix} T & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	Or/Maximum	OR	$A + B$
$\begin{bmatrix} T & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Consensus	CONS	$A \boxtimes B$
$\begin{bmatrix} T & T & 0 \\ T & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	Any	ANY	$A \boxplus B$
$\begin{bmatrix} 1 & 0 & T \\ 0 & 0 & 0 \\ T & 0 & 1 \end{bmatrix}$	Multiplication	MUL	$A \otimes B$
$\begin{bmatrix} 1 & T & 0 \\ T & 0 & 1 \\ 0 & 1 & T \end{bmatrix}$	Addition	SUM	$A \oplus B$

Figure 6: Useful Two-Input Ternary Gates

be discussed in the conclusion.

$$\begin{aligned}
AND &\equiv \mathbb{T}AND \\
OR &\equiv \mathbb{T}OR \\
NOT &\equiv \mathbb{T}NOT
\end{aligned} \tag{7}$$

This will allow us to create a basic binary and ternary emulator whose performance will be directly comparable.

2.3 Previous Ternary Implementations

2.4 Emulation of Circuits