Arithmetic Logic Design with Color-Coded Ternary for Ternary Computing

Afolayan A. Obiniyi
Department of Mathematics
Ahmadu Bello University
Zaria

Ezugwu E. Absalom
Department of Mathematics
Ahmadu Bello University
Zaria

Kwanashie Adako
Department of Mathematics
Ahmadu Bello University
Zaria

ABSTRACT
This paper introduces a novel means of representing ternary states using color-codes, suggests a logic design model for a ternary half adder circuit and separate carry circuits. A ternary simulator was also developed to aid in the research and development of ternary systems.

.General Terms
Arithmetic Logic Design, Color-Coded Ternary

Keywords
Ternary computing, color-coded ternary, ternary machine, ternary logic

1. INTRODUCTION
All commercially available Arithmetic Machines are designed and built using BASE 2 logic. Even though High Level Languages such as C++, JAVA, Visual Basic amongst others, all appear to be written in some form of cryptic English, computers find such languages incomprehensible in this format, so they reduce all program statements (in whatever language) to basic machine codes and reduce these even further to **0**'s and **1**'s (Flores, 1960) . Information represented in this dual state format becomes intelligible to the computer hardware, which is designed around different forms of two state devices such as transistors, magnetic cores, and flip-flops.

The exclusive success of Binary Machines has led to an erroneous impression (familiar among computer scientists) that; BASE 2 is the only base system which can be adapted to build Arithmetic Machines. Many Computer Scientists give little or no research-time to the many other forms of realizable Alternate Base Machines, to the sole benefit of Binary Systems. This work is a modest attempt made toward contributing to the body of knowledge relating to the design of Arithmetic Machines.

Ternary systems have three states (Arpasi, 2003), and these systems have been identified to have several features which distinguishes them from the other existing BASES and make them worth an extra look, some of these features include:

a. When representing information using base systems; the higher the base system used the more dense the representation(in other words, fewer spaces are required to represent a given number) but the down side is that there is an explosion of different symbols which must then be recognized (Hayes, 2001). If 'r' represents the radix of a number system and 'w' represents the width, then the minimum value of 'rw' for all bases systems represents the most efficient base in which to represent information. According to Brian Hayes; Base 3 has the second best values for 'rw' after Base e (the base of the natural logarithms)(Hayes, 2001).

b. If 'rw' is an approximate measure of hardware complexity (Hayes, 2001) then any reduction in its value represents a reduction in hardware components, a reduction in cost of building an arithmetic machine, and also a reduction in power consumption of the machine.

c. Ternary systems also involve a reduction in interconnections required to implement logic functions (Dhande et al, 2005) thereby reducing chip area.

d. More information can be transmitted over a given set of lines, and less memory space for a given data (Dhande et al, 2005)

e. According to Haidar et al (2008), "the ternary arithmetic and Logic Unit (ALU) is a breakthrough not only because we can control more functions in the same number of control lines but because it can perform faster than a binary ALU and it can become the first stone in switching from binary to Multi Value Logic (MVL)."

f. Negative numbers can be represented in a balanced ternary notation without the resolve to a sign trit thereby making it more efficient for arithmetic computations (Connelly, 2007) and because ternary uses less digits to represent given numbers than base two fewer arithmetic operations are required to add any two given numbers.

The aforementioned merits makes Base three a prospective field of research in the nearest future for computer scientists, because they can potentially lead to smaller, cheaper, more efficient and less power consuming Arithmetic Machines.

2. DEVELOPMENT OF TERNARY MACHINES
The first recorded practical Ternary Machine ever built, was a mechanical model built by Thomas Fowler, in 1840 (Glusker, 2008). Even though the original model hasn't survived, a working model, based on a description written by De-Morgan, was built recently by Mark Glusker, Pamela Vass and David Hogan (Glusker, 2008). Their model is shown in Figure 1.

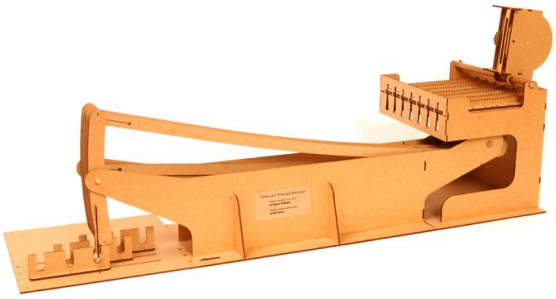

Fig. 1: Ternary calculating machine of Thomas Fowler

In 1958, a much more complicated machine was built in Russia by Nikolai P. Brousentsov and his colleagues at Moscow State University (Brousentsov et al, 1997). It was called *SETUN* and operated by storing ternary coded information on a pair of magnetic cores, wired in such a way that they had three stable states.

In 1973, Gideon Frieder and his colleagues' at the State University of New York designed a Base-3 machine they called *TERNAC*, and went ahead to create a software emulator for it, (Hayes, 2001). This machine like that of Fowlers and SETUN were all based on Balanced Ternary Arithmetic, that is, information is represented and processed using the three alternating states: (-1, 0, +1) (Hayes, 2001)

The first machine failed to evolve in to the modern computer system, because of its highly mechanical nature and resistance to extreme miniaturization. The failure of the second machine was due to its inability to hold on to its higher information-density title. Brian Hayes was right in his assertion that:

*"A pair of cores could have held two binary bits, which amounts to more information than a single trit, and so the ternary advantage was squandered." (*Hayes, 2001*).*

2.1 Benefits of Ternary Technology

Perhaps the most important and immediate use of ternary technology is in the new and emerging field of Quantum Computing, a technology which has been described as 'a promising and flourishing research area' (Khan, 2004)

Ternary computing also serves as a reference point from which a comparison of different base systems can be conducted, while higher bases maybe too cumbersome for practical use and lower bases quickly become too bogus, ternary systems strike a careful balance (Hayes, 2001).

Ternary logic also serves as a stepping stone on the way to Multi-Valued-Logic (MVL) which is currently being probed for its application in artificial intelligence, particularly for students who have not prior been introduced to the topic.

3. MATERIALS AND METHODS

The design features used in this work is one adopted from Binary systems, the machine is designed as an *Address Machine*, which uses a combination of hardware and software features to carry out Ternary Arithmetic.

Several basic processes are required in order for computers to execute instructions (Hellerman, 1967); these basic processes include:

i) Supplying a Memory Element with the address to an instruction.
ii) The Memory Element responds by supplying the instruction that was stored at the given address.
iii) A Control Unit decodes the instruction and executes the instruction by supplying correct control signals to different system components for example, an Arithmetic and Logic Unit.

Ternary logic design theory adapts these principles in order to execute arithmetic instruction codes in an orderly and well synchronized fashion. The ternary simulator backing this research work was developed in Visual Basic 6.0 platform and deployed on an ACER Extensa 4220 Laptop, with 512MB RAM, 80GB, Intel Celeron processor 530 (1.73 Ghz, 533 Mhz FSB, 1 MB L2 cache)

4. RESULT AND DISCUSSION

The diagram shown Figure 2 is a schematic diagram showing functional elements and the direction of flow of information in the proposed ternary system.

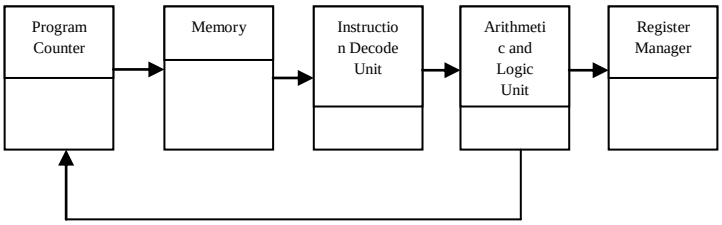

Fig. 2: Functional Diagram of a Proposed Ternary Machine

4.1 Ternary Logic Operators

A Ternary logic element satisfies the condition:

F = A#B

Where A, B and F ∈ [0,1,2]

And # is the logic operator.

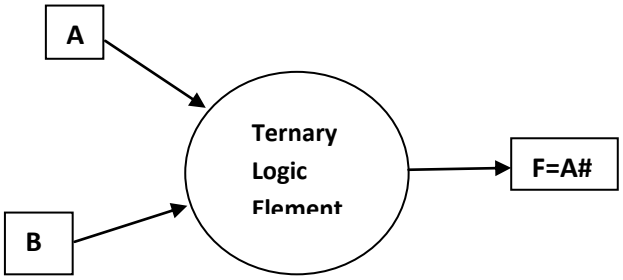

Figure.3: Schematic Diagram of a Ternary Logic Element**.**

The truth table depicted in Table I is for some selected ternary operators. For this table: value 1 is equivalent to a binary TRUE state, 2 represents a binary FALSE state and 0 represents an UNKNOWN or MAYBE state.

Table I: Truth Table for Ternary Functions.

A	B	A OR B	A AND B	NOT A
1	1	1	1	2
1	0	1	0	2
1	2	1	2	2
0	1	1	0	0
0	0	0	0	0
0	2	0	2	0
2	1	1	2	1
2	0	0	2	1
2	2	2	2	1

4.2 Ternary Arithmetic

Table II shows some basic arithmetic operations performed on ternary arithmetic.

Table II: Truth Table for Ternary Arithmetic Operations

A	B	Sum	Carry	Diff	Borrow	Max	Min
1	1	2	0	0	0	1	1
1	0	1	0	1	0	1	0
1	2	0	1	2	1	2	1
0	1	1	1	2	1	1	0
0	0	0	0	0	0	0	0
0	2	2	0	1	1	2	2
2	1	0	1	1	0	2	1
2	0	2	0	2	0	2	0
2	2	1	0	0	0	2	2

Figure 4 displays the diagram of a proposed ternary machine.

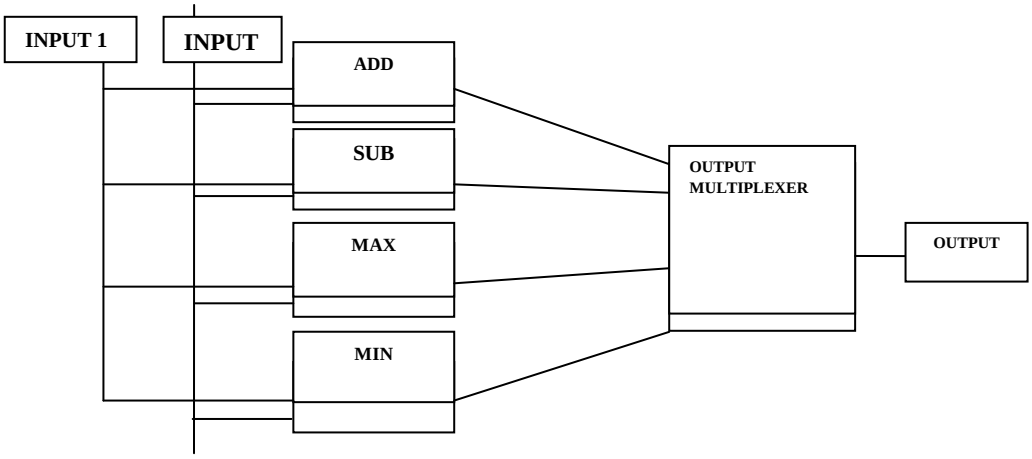

Fig 4: Functional diagram for the proposed Ternary Arithmetic Machine

5. COLOR-CODED GATES AND OTHER DEVICES

5.1 Color Code Representation

By putting logical restrictions on the paint artist impression of how primary colors combine to form secondary colors, Ternary states can be fruitfully represented as the three primary colors.

Color codes can be arbitrarily assigned to ternary values. These assignments can be changed as the purpose fits, but for the rest of this paper, the following format will be employed.

(a) Red= 0 → R≠ 1 and R≠ 2

(b) Green= 1 → G≠ 0 and G≠ 2

(c) Blue= 2 → B≠ 0 and B≠ 1

And the three primary colors can be generated by logical complementation.

R = G'.B'

G = R'.B'

B = R'.G'

5.2 Positive Complement Gate (PCG)

A Positive Complement Gate is a device purposely built to take two color coded inputs and generate a one color coded output based on the rules of positive complementation. It is represented by an equilateral triangle, with two arrows pointing to the triangle and one arrow pointing out as is illustrated in Figure 5. The truth table for this gate is shown in Table III.

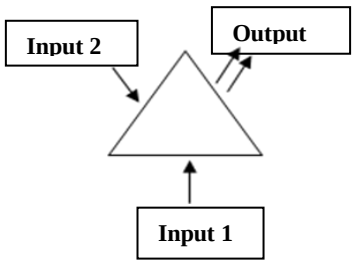

Fig. 5: Functional diagram of a Positive Complement Gate.

Table III: Truth Table for a Positive Complement Gate

Input 1	Input 2	Output
R	G	B
R	B	G
G	B	R
R	R	R*
G	G	G*
B	B	B*

* When two similar inputs appear on the input ports of a Positive Complement Gate (PCG), the same color is produced as output.

5.3 Negative Complement Gate (NCG)

A Negative Complement Gate is a device purpose built to take a one color coded input and generate two color coded outputs based on the rules of negative complementation.

It is represented by an equilateral triangle with one arrow pointing to the triangle and two arrows pointing out, Figure 6 is a functional diagram of a Negative Complement Gate and Table IV is a truth table for this device.

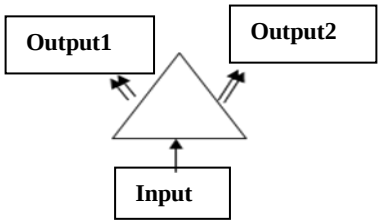

Fig.6: Functional Diagram of Negative Complement Gate.

Table IV: Truth Table for a Negative Complement Gate

Input	Output1	Output2
R	G	B
G	R	B
B	R	G

5.4 Color Coded Screen (CCS)

A Color Coded Screen is a device that will allow only one color code through it, and will screen out others, (or it may screen one color code and allow others through). For example a Blue CCS will selectively screen Red and Green and allow only Blue to pass through. This is illustrated in Figure 7.

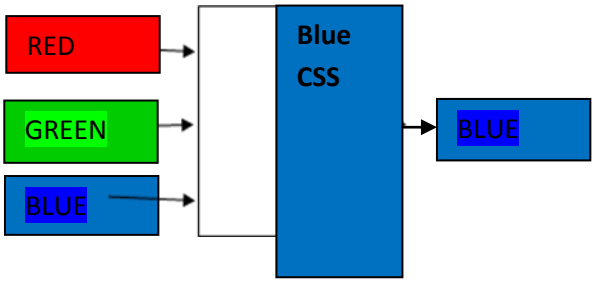

Fig 7a: Functional Diagram of a Color Coded Screen

Color Coded Screens, in this paper, are represented in color-coded diagrams as a box with a color code written in it, as is shown in Figure 7b.

Fig. 7b: Color coded diagram

5.5 Broadcast Box

A Broadcast Box is a device used to multiplex a single color code onto multiple output lines; such a device may contain a power source, which it uses to amplify the light signal to be broadcasted. The operation of this device is as shown in figure 8.

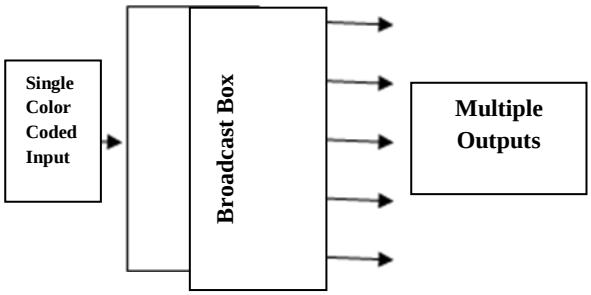

Fig. 8: Functional Diagram of a Broadcast Box

5.6 Optical Amplifiers (OA)

Optical Amplifiers are used to boost optical signals, which may have deteriorated due to interaction with circuit elements or even during transmission through optical conduits. They may be either electrical regenerators or purely optical devices which use laser to amplify optic signals.

5.7 Arithmetic Circuits

One of the advantages of using number bases to perform arithmetic is the fact that, arithmetic operations, which include division, multiplication, subtraction and addition, can all be carried out using only addition (and suitable formatting techniques), this advantage greatly simplifies the task of designing an Arithmetic Machine for any base, all that is needed is to; design circuits for Addition. The addition circuits presented in this paper are certainly not the only way to implement ternary addition. Addition can be carried out by attaching the OUT of a PCG (Positive Complement Gate) to the IN1 or IN2 of another PCG (Positive Complement Gate), this is illustrated in Figure 9.

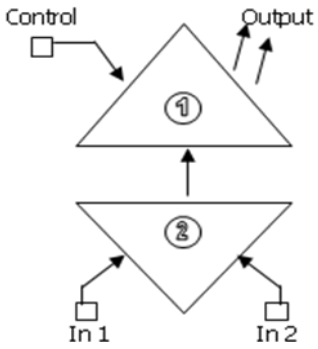

Fig. 9: Arithmetic Circuit Element

Figure 9 is a realization of the logic function given by:

P (a . b) = T; P(x.y) implies the positive complement of x and y (1)

A (a . b) = P (T . Control) ;

A(x.y) implies the arithmetic sum without

carry of x and y (2)

where x, y, P(), A() ∈ [0, 1, 2]

Equations 1 and 2 are a strict interpretation of

the more general

form given by;

P((x.y).c) = 0 (3)

x, y, c, P(), and 0 ∈ [0, 1, 2]

The addition without carry of any two primary codes,

are satisfied as shown in the Table v:

Table V: Truth Table for Addition using Color Circuits

Input 1	Input 2	Positive complement of(In1.In2)	Control	Positive complement of ((In1.In2).Control)
(R) 0	(R) 0	(R) 0	(R) 0	(R) 0
(G) 1	(G) 1	(G) 1	(R) 0	(B) 2
(B) 2	(R) 0	(G) 1	(R) 0	(B) 2
(G) 1	(R) 0	(B) 2	(R) 0	(G) 1
(G) 1	(B) 2	(R) 0	(R) 0	(R) 0*
(B) 2	(B) 2	(B) 2	(R) 0	(G) 1*

By forcing the input1 of PCG1 which is labeled control in Fig. 9 to a permanent RED, the Unbalanced Ternary Addition without Carry, of any two primary colors put on IN1 and IN2 of PCG2 is the result realized in the output of PCG1.

It is pertinent to note that the operations marked with asterix are operations where carry pix are expected, which raises the issue of how carry pix are to be generated and handled. The solution is to either

a. Incorporate a hardware element to the addition circuit to detect when any of the possible combinations that produce carry pix occurs, or

b. handling the carry pix can be taken care of with appropriate software codes, which test the two pix to be added and 'predicts' that a carry pix 'should be' generated, so the carry pix (predicted) is then incorporated in the next addition step.

Subtraction, Multiplication, and Division Algorithms can be carried out using modified Addition Algorithms and suitable formatting techniques, so it is unnecessary to develop detailed circuits for these operations.

5.8 Carry Circuits

The purpose of a carry element is to identify when a combination that should cause a carry has occurred and respond by generating the carry pix, carry circuits can be built by slightly modifying an Arithmetic Circuit Element, to include Color Coded Screens and special controls. Figure 8 shows an Arithmetic Circuit Element attached to a carry element which generates a carry color code whenever the input which should produce a carry, are applied to Input1 and Input2, Table vi is a truth table for the carry element.

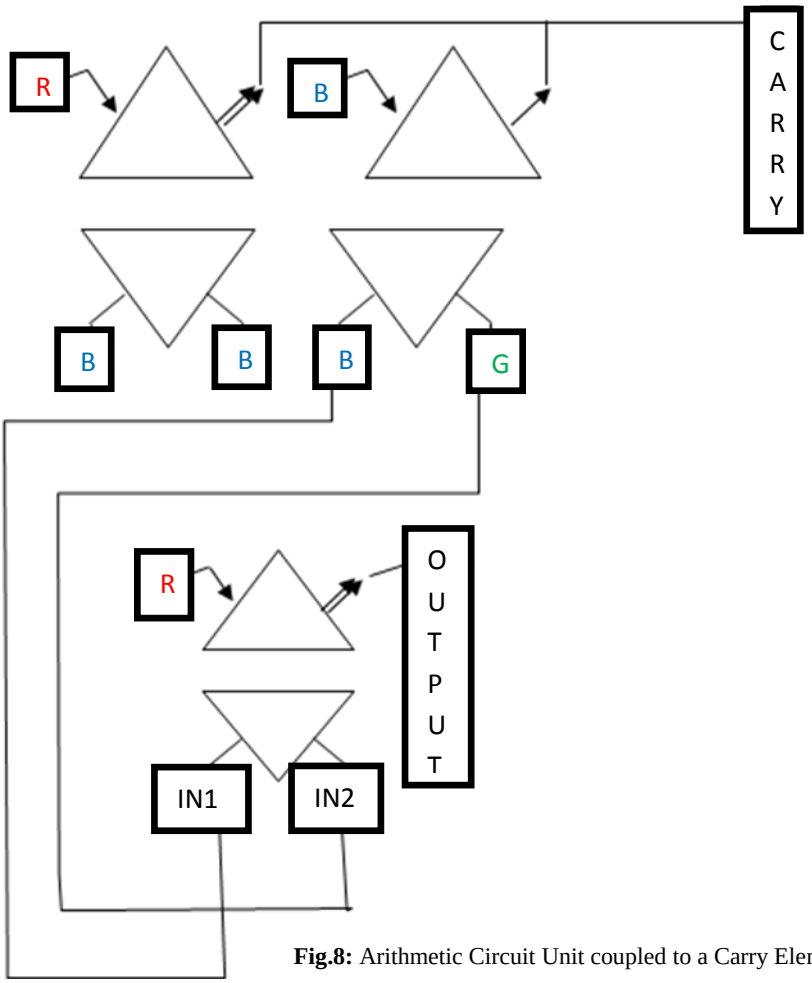

Fig.8: Arithmetic Circuit Unit coupled to a Carry Element.

Table VI: Truth table for the Carry Element

Input 1	Input 2	Positive Complement of (In1.In2)	Control	Positive Complement of ((In1.In2).Control)
(G) 1	(B) 2	(R) 0	(B) 2	(G) 1
(B) 2	(B) 2	(B) 2	(R) 0	(G) 1

5.9 Ternary Simulator

The ternary simulator provided was developed on a Visual Basic 6.0 platform; it is intended as a research tool to aid in the development of the proposed ternary systems. The form contains three functions and one Arithmetic Circuit Element simulator:

a. **Decimal to Ternary Converter:**

The decimal to ternary converter contains a decimal input box labeled **Enter Decimal number:** and an output box labeled **Ternary Equivalent:** and two command buttons labeled **Convert and Clear**. The converter converts any number given in decimal to its ternary value as is shown in Figure 9 and 10.The clear command resets all the input and output entries.

b. **Positive Complement Simulator:**

The Positive complement Simulator imitates the operation of a Positive Complement Gate, which is a core component of Arithmetic circuit Element. It has two inputs **Input1** and **Input2** and an **Output** which gives the Positive complement of the two inputs provided, and a command button labeled **Positive Complement**

c. **Arithmetic Sum:**

The Arithmetic sum displays the arithmetic sum and carry of any two ternary variables inserted in **Input1** and **input2**

d. **Arithmetic Circuit Simulator:**

This simulates the operation of an Arithmetic Circuit element using the color codes from **Input1** and **Input2**.

Ternary Code snapshot

```
parameters: value, base, digits

    int decimal[digits]; // Stores the ordinary decimal number, one digit per array entry.

int ternary[digits]; // Stores the base-N ternary number.

// Put the normal base N number into the baseN array. For base 10, 109

// would be stored as [9,0,1]

for(i = 0; i < digits; i++) {

baseN[i] = value / pow(base,i) % base

}


// Convert the normal baseN number into the ternary equivalent. Note that

// the loop starts at the most significant digit and goes down.

int shift = 0;

for(i = digits; i >= 0; --i) {

// The ternary digit gets shifted down equal to the sum of the higher

// digits.

ternary[i] = (baseN[i] - shift) % base

shift += ternary[i]

}
```

The above code snapshot follows a general principle for ternary system conversion indicated in (*Sakamoto and Ikeda, 2011*):

$(t_{n-1}t_{n-2} \cdots t_1t_0)ST$

$= t_{n-1}3^{n-1} + t_{n-2}3_{n-2} + \cdots + t_13^1 + t_0$

$t_i \in \{ 1 , 0 , 1 \}, i = 0, 1, \ldots ,n - 1$

radix = 3

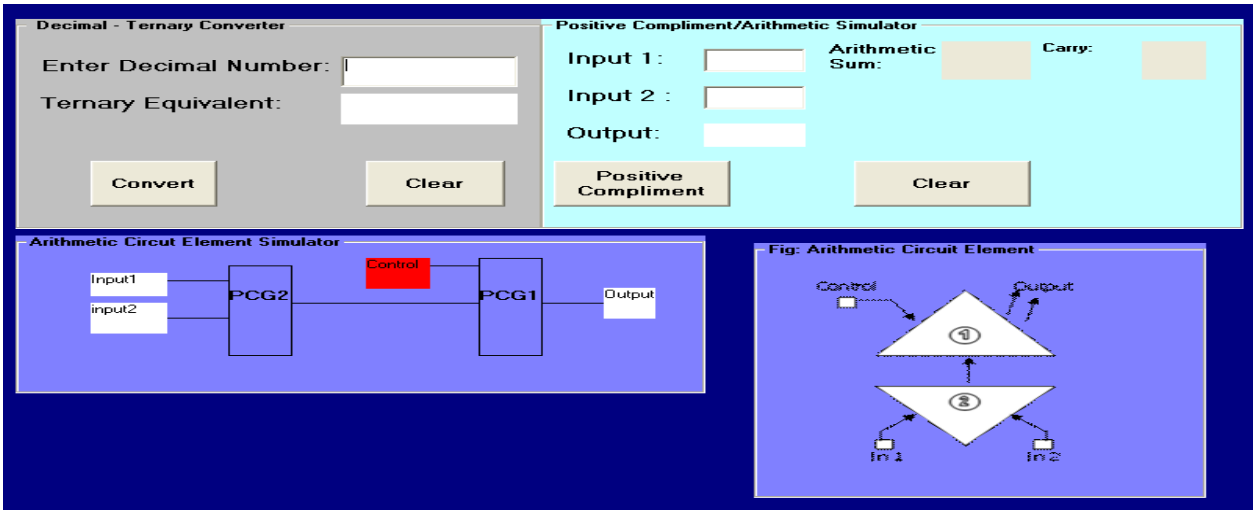

Fig: 9: A Ternary Simulator

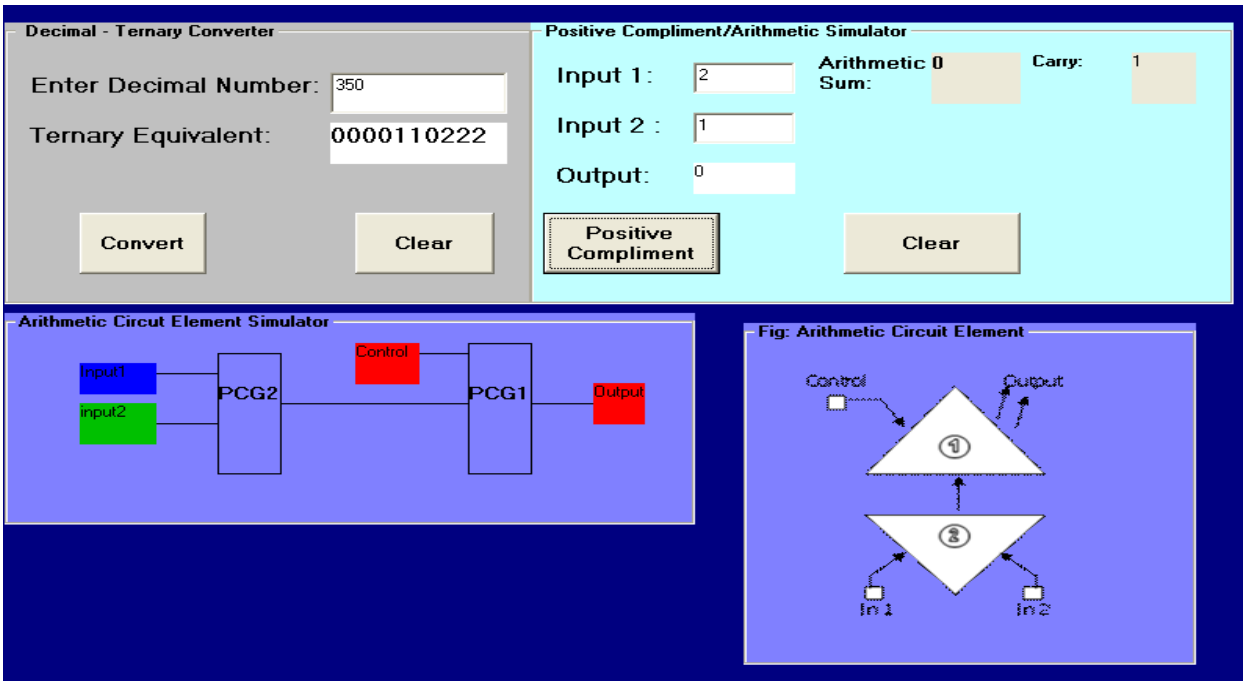

Fig. 10: Ternary simulator in Use

6. CONCLUSIONS

Traditionally Arithmetic and Logic Units (ALU) form the core component of Central Processing Units (CPU), and the Ternary Arithmetic and Logic Unit (TALU) described in this paper is also meant to form the core component of a complete Ternary Central Processing Unit (TCPU). Due to the benefits of ternary representation, which include a more compact number system, fewer operations for arithmetic operations and better efficiency, this paper concludes that ternary arithmetic machines even though they are not yet commercially available, remain a viable field of research, and have a promising future as a replacement for binary machines.

7. REFERENCES

[1] Brian Hayes: *Third Base*, American Scientist Online, 2001, www.americanscientist.org/issues/pub/third-base/, (accessed: Nov. 2010)

[2] Brousentsov N. P., Maslov S. P., Ramil Alvarez J., Zhogolev E.A.: *Development of Ternary Computers at Moscow State University*, 1997-2008, http://computer-museum.ru/english/0.htm (accessed: Oct. 2010)

[3] Flores Ivan: *Computer logic: the functional design of digital computers*, 1960, Prentice-Hall inc. Englewood cliffs, New Jersey.

[4] Mark Glusker, 2008: *The Ternary Calculating Machine of Thomas Fowler* www.mortati.com/glusker/fowler/demorgan.htm (accessed: Nov. 2010)

[5] Haidar A., Hamdan M.J., Rashid M. B., Hassan H., Ahmad I., Abdallah K., A Novel (2008): Neural Network Ternary Arithmetic Logic Unit, the 23rd International Technical Conference on Circuits/systems, computers and Communication (ITC-CSCC 2008)

[6] Connelly Jeff: *Trinary Computer Systems*, 2007, http://jeff.tk/wiki/Trinary (accessed: Nov. 2010)

[7] http://xyzzy.freeshell.org/trinary/ (accessed: Nov. 2010)

[9] Jorge Pedraza Arpasi: *A Brief Introduction to Ternary Logic*, 2003, http://aymara.org/ternary/ternary.pdf (accessed: Nov. 2010)

[10] Khan M.H.A.: *Quantum realization of ternary adder circuits*, 3rd international conference on electrical and computer engineering (ICECE 2004) Dhaka, Bangladesh

[11] Muzio J.C and Miller D.M, 1976: *A Ternary Universal Decision Element.* Notre dame journal of formal logic volume XVII, Number 4

[12] Richard Barnett: *Microprocessor System Design Techniques*, Sigma Publishing Company, Wilmson UK, 1991, PP 15-29

[13] Viktor Lofgren, 2008, Tunguska the ternary computer emulator, www.acu.umu.se/~achtt315/tunguska/docs.html (accessed: Nov. 2010)

[14] *Masahiro SAKAMOTO and Takeshi IKEDA*, A Fast Converter Circuit From Signed-Ternary Number to Radix-2 Signed-Digit Number: https://www.itc-cscc2010.org/main/sites/default/files/paper/219.pdf, (Accessed November, 2010)