International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

Arithmetic of 5th Generation Computer

Sinthia Roy[1], Sanjit Kumar Setua[2]

Assistant Professor, Computer Science & Engineering Department, Guru Nanak Institute of Technology

Associate Professor, Computer Science & Engineering Department, University of Calcutta, West Bengal, India

---

Abstract–*Decimal number system is purportedly the most used number system by humans. However, there are diverse ways of counting numbers, when it comes to computer systems, the number system that crosses our mind is the binary number system. When represented, binary number system takes a considerable number of bits in comparison to ternary number system i.e trits.*
In this paper we propose an algorithm based on ternary number system which enables us to multiply two ternary numbers, which will be more efficient in a ternary system only.

***Keywords*— Binary, Ternary, Trit, Arithmetic, Decimal, Conversion.**

I. INTRODUCTION

During the childhood of organic systems the counting was with the pebbles first, then with fingers, afterwards with Abacus, slide rule, calculator and finally digital computers.[1] Digital computer deals with digits. In fact a computing system is nothing but high speed, cost effective manipulator of different codes. The bottle neck of codes and their technology mapping was overcome by the advent of positional number systems.

In computers, if we can count something then it cannot be infinite. The problem can be solved if we take only binary digits 0 and 1 using positional number system. This is because 0 and 1 can serve our multipurpose activity. Now man machine interface demands use of decimal numbers, but alas 10 symbols required for this cannot be represented in a robust way.

Although we are accustomed to deal with numbers 0 and 1 effectively, according to Hayes, base e=3 is the most optimal base[16]. Moreover it is shown that it is much more efficient in ALU design provided we can forecast an balanced technology mapping . Therefore we will be dealing with base 3 i.e ternary number system in the rest of the paper whose benefits are illustrated in the section II.

In this paper, we demonstrate that addition, subtraction and multiplication can be done inexpensively if time factor can be effectively minimized.

The rest of the paper has been organized as follows: Section II delineates ternary number system , its optimality and its benefits. Section III specifies the algorithm to convert a decimal number to its equivalent balanced ternary number. Section IV describes the kinds of arithmetic operations viz. addition, subtraction and multiplication. Section V concludes the paper.

II. TERNARY NUMBER SYSTEM

A. *What is ternary number system?*

Ternary (sometimes called trinary) is the base 3 numeral system. Analogous to bit, a ternary digit is a trit (trinary digit). One trit contains $\log_2 3$ (about 1.58496) bits of information. Ternary number system is of two kinds namely unbalanced ternary number system and balanced ternary number system. Unbalanced ternary number system is expressed using the symbols 0, 1 and 2[18]. Balanced ternary is a non-standard positional number system (a balanced form), useful for comparison logic. Unlike unbalanced ternary system, the digits have the values −1, 0, and 1[5]. Balanced ternary can represent all integers without resorting to a separate minus sign. Therefore in balanced ternary number system, if we encounter a positive number then its corresponding negative number can be understood by interchanging the sign, for ex: $\bar{1}11$ represents -5 and $1\bar{1}\bar{1}$ represents +5. Balanced ternary is enumerated as follows- the symbol $\bar{1}$ / 1' denotes the digit −1, but alternatively for easier parsing "−" may be used to denote −1 and "+" to denote +1.

B. *Why Base 3 is considered as the most optimal base?*

- Firstly, it is seen that for a particular decimal number, the requisite number of bits are larger than the equivalent number of trits which has been shown in table 1. Thus Ternary number system offers the most economical way of representing numbers.[2]

Value in Decimal Number	Length of the Binary Equivalent	Length of the Ternary Equivalent
4	0100	0 1 1
16	10000	1 -1 -1 1
64	1000000	1 -1 1 0 1
256	100000000	0 1 0 0 1 1 1
1024	10000000000	0 1 1 1 -1 0 -1 1

Table 1 : The decimal numbers and their corresponding binary & ternary numbers.

- Secondly, according to Haye's[16] the cost(i.e cost of computing for a particular base or cost in terms of space or time tradeoff) of a particular base or radix 'r' is proportional to 'r', if r(for radix) denotes the number for primitive symbols or digits used to represent numbers in a computer. This is quiet reasonable because the complexity of the circuits(i.e appropriate physical logic) needs to store and process numbers increases with r. Now according to him the optimal base, which is the product of base and width (number of digits used to represent a number) of the digit is an objective function needs to be minimized[16]. Thus, for optimal number of computations, it may be noted that the product b × w needs to be minimized, where b is the base and w is the width in the digits. Using simple mathematics it is shown that for optimal result, b = e. For practical purposes, the integer 3,taken as the ceiling function of e is considered as the optimal value of base width.

C. Benefits of Ternary Technology

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.

Ternary computing also serves as a reference point from which a comparison of different base systems can be conducted, while higher bases may be too cumbersome for practical use and lower bases become too phony, therefore ternary systems strike a careful balance[16]. Some of the interesting properties of a balanced ternary number system include[10] :

1. The negative number is obtained by interchanging 1 and -1.

2. The sign of a number is given by its most significant nonzero trit.

3. The operation of rounding off to the nearest integer is identical to truncation.

Ternary number system 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.

III. CONVERSION OF DECIMAL TO BALANCED TERNARY NUMBER

Conversion from decimal to balanced ternary number requires converting the decimal number to unbalanced ternary notation (represented with 0, 1, and 2). Then we have to append a zero before the unbalanced number. The unbalanced ternary is then converted to balanced ternary by adding runs of 1's i.e 1111… with considering carry, then again from the resultant unbalanced ternary numbers subtract runs of I's i.e 1111… without considering borrowing (the thing to be noted is that the string of 1s should be of the same length as the ternary numbers, so if the result of the addition has more number of digits, then subtract nothing from these extra digits). In order to add 1s to the unbalanced ternary notation, ternary arithmetic addition operation is performed. The following table 2 shows the basic addition operations performed on ternary arithmetic.

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

Table 2 : Truth Table of Unbalanced Ternary arithmetic operation

The Decimal to balanced ternary conversion can be done using the following algorithm:

```
//Accept the input
decNum ← Decimal number from user
base ← Target base(in this case it is 3)

//convert decNum to unbalanced ternary
while(decNum!=0)
do
{
```

```
 remainder ← decNum % base;
 unbalancedTernary ← (remainder * 10) +
unbalancedTernary;
 decNum ← decNum / base;
}

//Prefix unbalancedTernary with a zero
unbalancedTernary ← "0" + unbalancedTernary;

//Generate a run of one whose length matches
//with the unbalancedTernary
For i=1 step 1 to lengthofUnbalancedTernary
 //Prefix unbalancedTernary with a zero
 secondTernay ← "1" + secondTernay;
end For // loop on i

//Add secondTernary to unbalancedTernary
considering //carry
unbalancedTernary ← secondTernay +
unbalancedTernary;

//Subtract secondaryTernay from unbalancedTernary
//This gives resulting balanced ternary
balancedTernary ← unbalancedTernary –
secondaryTernay;
```

Decimal number = 55		Base = 3	
Number	**Operation1**	**Result**	**Operation 2**
55	55%3=1	1	Number = 55/3 = 18
18	18%3=0	0	Number = 18/3 = 6
6	6%3=0	0	Number = 6/3 = 2
2	2%3=2	2	Number = 2/3 = 0
Unbalanced Ternary Equivalent= 2001			

*'%' -modulo operator

Table 3: Example of Conversion from Decimal to Unbalanced ternary & Unbalanced ternary to Balanced Ternary

Example:

Adding 11111 to 02001: 0 2 0 0 1
1 1 1 1 1
2 0 1 1 2

Subtracting 11111 from 20112: 2 0 1 1 2
1 1 1 1 1
1 -1 0 0 1

So, Resultant Balanced Ternary equivalent for decimal number 55 is 1 -1 0 0 1.

IV. ARITHMETIC OPERATIONS IN BASE 3 NUMBER SYSTEM

A. Addition- The underlying table illustrates some examples of additions for the ternary number system. Each column corresponds to a pair of trits to be added and a carry trit. Thus, the total number of possible columns would be $3^3 = 27$.

A	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
B	0	0	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	0	0	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	0	0	1	1	1	$\bar{1}$	$\bar{1}$	$\bar{1}$	0
C	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	$\bar{1}$
Sum	$\bar{1}$	0	$\bar{1}$	0	1	$\bar{1}1$	$\bar{1}$	0	$\bar{1}$	0	1	0	1	$1\bar{1}$	$\bar{1}$	0	1	0	1	$1\bar{1}$	1	$1\bar{1}$	10	$\bar{1}0$	$\bar{1}1$	$\bar{1}$	$\bar{1}1$

Table 4: Addition of trits.

B. Subtraction- The operation of Subtraction can be viewed simply as negation of a number followed by addition.

- *Multiplication*- For multiplication we store multiplicand in a register BR, (say), & ABR is another register which holds the sign bit of the multiplicand. Multiplier is again stored in a register QR, (say), & AQR is another register holding the sign bit of the multiplier. AC is a register which acts as an accumulator and AAC register correspondingly holds the sign bit of the accumulator. Initially, we assume that product is zero. This is known as the partial product, where a partial product is obtained by multiplying the multiplicand with one trit of the multiplier. In simple multiplication, if the trit of the multiplier is 1( ) then multiplicand is added (subtracted) with the partial product to generate a new partial product. Now the next trit of the multiplier is multiplied with multiplicand and the product is shifted by one trit to the left and added with the partial product to generate a new partial product.[10] But in case of hardware multiplication (using registers), instead of shifting the (multiplicand × c) (where c is a trit of the multiplier, having value 0 or 1 or) to the left we shift the partial product one trit to the right. We use the term ASHR to indicate arithmetic shift right. The multiplication algorithm is shown below.
-

Algorithm:

// Initialize ER to zero
ER ← 0;
Initialize AC(accumulator), BR(multiplicand) and QR(multiplier), and their respective sign representations AAC,ABR and AQR respectively
while SC != 0
do
 If LSB of multiplier = 1
  If LSB of the sign representation == 1
   AC ← AC – multiplier;
   //same operation is performed on the signed representation
  else
   AC ← AC + multiplier;
   //same operation is performed on the signed representation
 Perform Arithmetic right shift on ER, AC, QR, AAC, and AQR;
 SC ← SC -1;
If SC = 0
 ER, AC, QR is displayed along with ER, AAC, and AQR; this represents the ternary equivalent.

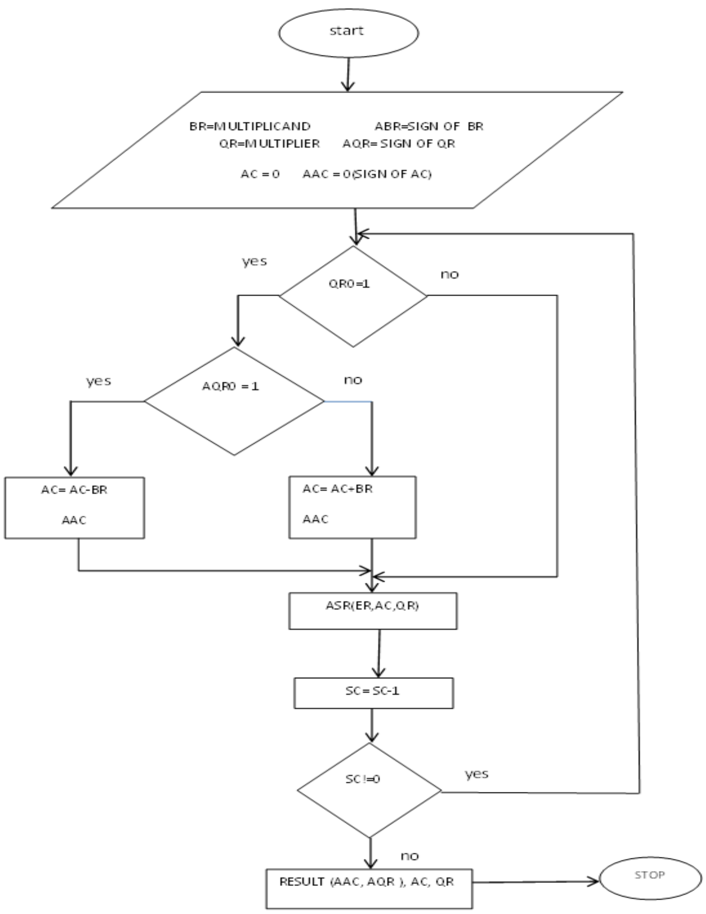

Figure 1: Flowchart of Ternary Multiplication algorithm

Example:

- BR=1 $\bar{1}$ 0 0 1(i.e 55 in decimal)
  BR=1 1 0 0 1
  ABR=0 1 0 0 0
- QR=1 $\bar{1}$ 0 1 $\bar{1}$(i.e. 56 in decimal)
  QR=1 1 0 1 1
  AQR=0 1 0 0 1
- $\overline{BR}$=$\bar{1}$ 1 0 0 $\bar{1}$
  $\overline{BR}$ = 1 1 0 0 1
  $\overline{ABR}$ = 1 0 0 0 1

QR[0]	AQR[0]	Operation	ER	AC AAC	QR AQR
		Initial. Sn=5	0	00000 00000	11011 01001
1	1	AC=AC-BR=AC+$\overline{BR}$	0	11001 10001 11001 10001	11011 01001 11011 01001
		Shift. Sn=4	0	01100 01000	11101 10100
1	0	AC=AC+BR	0	11001 01000 01101 00000	11101 10100 11101 10100
		Shift.Sn=3	0	00110 00000	11110 01010
0	0	Ashr. Sn=2	0	00011 00000	01111 00101
1	1	AC=AC-BR=AC+$\overline{BR}$	0	11001 10001 11010 10000	01111 00101 01111 00101
		Shift. Sn=1	0	01101 01000	00111 00010
1	0	AC=AC+BR	0	11001 01000 01111 00001	00111 00010 00111 00010
		Shift. Sn=0	0	00111 00000	10011 10001

Table 5: Example of Ternary Multiplication

Final Product $= 0\ 0\ 1\ 1\ \bar{1}\ 1\ 0\ 0\ \bar{1}\ 1$

$= (3080)_{10}$

V. CONCLUSION

The model of computation provided by an ordinary computer assumes that the basic arithmetic operations- Addition, subtraction, multiplication and division can be performed in constant time. This abstraction is reasonable because most basic operations on a random access machine have similar costs. But when it comes to designing the circuit's those implement these operations, from here we realise that the performance depends on the magnitudes of numbers been operated on.

References

[1] Behrooz Parhami.2005. Computer Architecture-from microprocessors to supercomputers. United States: Oxford University Press.

[2] Cormen, Thomas, H. et al. (2008). Introduction to Algorithms. Tata McGraw Hill

[3] Rangaraju H G, Venugopal U. (3 september 2010). Low Power reversible Parallel Binary adder subtractor.International Journal of VLSI design & communications System.

[4] TedBorys. (19.04.2004). Computer Arithmetic.

[5] ProfLoh.(February 2,2005). Carry Save Addition.

[6] Digi Key Corporation (7 September,2006). Binary Coded Decimal Division by Shift & Subtract Application Note:DKAN0003A.

[7] SherifGalal, Dung Pham. Division Algorithms and Hardware Implementations.

[8] AbenetGetahun. (2003). Booth Multiplication Algorithm.

[9] Prof. M RajashekharaBabu. Binary Multiplicatin-Booths Algorithm.

[10] Subrata Das et.al. (2011). Algorithms for Ternary Number System. Science Direct

[11] Afolayan A. Obiniyi et.al.(July 2011).Arithmetic Logic Design with Color-Coded Ternary for Ternary Computing. International Journal of Computer Application.

[12] A.Satish Kumar et.al.(2010). Minimization of Ternary Combinational Circuits-A Survey.Internatioanal Journal of Engineering Science & Technology.

[13] A.P.Dhande et.al.(2007). Ternary Logic Simulator Using VHDL.IEEE.

[14] HenningGundersen et.al.(2000). Ternary Operations.University of Oslo.

[15]file:///E:/M%20TECH/SSS%20PROJECT/Ternary%20study/Paper/History-of-Ternary-Computers.htm

[16] Brian Hayes.December(2001).Third Base.A reprint from American scientist.

[17] http://en.wikipedia.org/wiki/Ternary_numeral_system

[18] http://en.wikipedia.org/wiki/Balanced_ternary

[19] http://www.mortati.com/glusker/fowler/ternary.htm

[20] http://www.scribd.com/shellreef3125/d/78370674/34-B-1-Balanced-Arithmetic