INDIAN INSTITUTE OF MANAGEMENT CALCUTTA

WORKING PAPER SERIES

WPS No. 714/ September 2012

Arithmetic Algorithms for Ternary Number System

by

Subrata Das
Assistant Professor, Department of Information Technology,
Academy of Technology, West Bengal

Parthasarathi Dasgupta
Professor, IIM Calcutta, Diamond Harbour Road, Joka, Kolkata 700104, India

&

Samar Sensarma
Professor, Department of Computer Science & Engineering University of Calcutta

Arithmetic Algorithms for Ternary Number System

Subrata Das, Parthasarathi Dasgupta and Samar Sensarma

***Abstract*—The use of multi-valued logic in VLSI circuits can reduce the chip area significantly. Moreover, there are several additional advantages of using multi-valued logic in VLSI over the conventional Binary logic, such as energy efficiency, cost-effectiveness and so on. It has been shown that Base-3 number system is nearly optimal for computation. This paper is composed of two parts. In the first part of this paper we have studied some existing logical operation on ternary number system. We have also discussed some of the existing arithmetic operations using ternary number system. Some new algorithms for arithmetic operations have also been proposed, and shown to be quite efficient in terms of time complexity. In the second part of this paper we have discussed a special class of Boolean function, known as *Rotation Symmetric Boolean Function* in base-3. Algorithms for Rotation symmetric Boolean Function in base-3 is also proposed in this paper.**

***Index Terms*—Ternary number, trit, arithmetic, 3-valued logic,VLSI.**

I. Introduction

Numbers are counted in tens by human beings simply because we use our ten fingers for counting. Digital computations are based on binary logic because of digital devices having two states: ON and OFF. There are countably infinite number of ways to represent numbers. The number of digits used to represent numbers and the size of the base are two parameters to select a number system. For a unary number system only one symbol is required to represent a number but the number of digits required is high. On the other hand if the value of base is high, then less number of digits is required to represent a number, and the number of different symbols required is large. In order to have an optimal number system the product of base(b) and width (w) (i.e. number of symbols used to represent a number) should be minimized [1] where b^w is a constant. It is found that for optimal result the base should be chosen as e. As 3 is the integer nearest to e base 3 i.e. ternary number system is a good choice for representing a number.Ternary logic has some advantages over traditional binary logic. The information content can be increased by using ternary logic instead of using conventional binary logic. It is expected that the use of ternary logic in VLSI implementations should be energy efficient and cost effective[14]. Moreover, arithmetic operations can be performed at higher speed compared to binary logic.

Rest of the paper is organized as follows. Section II reviews some of the related recent works. Section III introduces some of the basic terminologies to be used in subsequent discussion. Section IV discusses the applications of ternary logic in computer science specially in the field of $VLSI$, $Computer\ Architecture$ and $Codding\ theory$. Section V discusses different arithmetic algorithms for ternary number system. Section VI discusses hardware implementation of multiplication and division algorithms . Section VII analyzes the performances of multiplication and division algorithms. Section VIII discusses a special of *Boolean Function* known as *Rotation Symmetric Boolean Function* and different algorithms for generating the same. Finally, Section IX concludes the chapter and briefly states the future scopes of work.

Subrata Das is with the Department of Information Technology , Academy of Technology,Aedconagar, Hooghly,India.
E-mail: dsubrata.mt@gmail.com
Parthasarathi Dasgupta is with the Dept. of MIS group,Indian Institute of Management Calcutta,India.
E-mail: partha@iimcal.ac.in
Samar Sensarma is with the Dept. of Computer Science and Engineering University of Calcutta,India,India .
E-mail:sssarma2010@gmail.com.

This paper is an extended version of the paper [22]

II. Literature review

A detailed review on third base number systems and the justification of the use of third base are reported in [1]. Several advantages of using ternary logic(multi-valued logic) over the traditional binary logic appear in [18]. A survey on the development of the algebras and techniques for the realization of three valued function are reported in [19]. The design and implementation of a low power ternary full adder is described in [7]. In [8], the authors represent a novel method for defining, analyzing and implementing the basic combinational circuit with minimum number of ternary multiplexers along with a survey on ternary switching algebra. A complete architecture, design and implementation of 2-bit ALU slice are discussed in [9]. A new type of transmission functions theory appear in [10]. Ternary mirror symmetrical number system is discussed in [11]. This paper also discusses technical realization of ternary symmetrical structure using binary logical elements along with discussions on ternary Analog to Digital and Digital to Analog converter, and introduces the concepts of *flip-flap-flop*. In [13] a mixed binary-ternary number system and its application in elliptic curve cryptosystem are discussed.

$B-T-N$	$D-N$
00000	0
00001	1
$0001\overline{1}$	2
00010	3
00011	4
$001\overline{11}$	5
$001\overline{1}0$	6
$001\overline{1}1$	7
$0010\overline{1}$	8
00100	9
00101	10
$0011\overline{1}$	11
00110	12
00111	13
$01\overline{1}\ \overline{1}\ \overline{1}$	14
$01\overline{1}\ \overline{1}0$	15
$01\overline{1}\ \overline{1}1$	16
$01\overline{1}001$	17
$01\overline{1}00$	18
$01\overline{1}01$	19
$01\overline{1}1\overline{1}$	20
$01\overline{1}10$	21
$01\overline{1}11$	22
$010\overline{1}\ \overline{1}$	23
$010\overline{1}0$	24
$010\overline{1}1$	25

TABLE I
BALANCED TERNARY NUMBERS AND CORRESPONDING DECIMAL NUMBER

III. PRELIMINARIES

In this section we discuss about some basic concepts of ternary number system, ternary logic and ternary gates.

A. Ternary Number

Three different symbols 0,1,2 are used to represent ternary number system.In ternary system the term trit is used to represent ternary digit One of the major issue is how we can represent a negative number in ternary number system. One easy solution is to use signed $3's$ complement representation to represent negative ternary numbers.
$3's$ complement of any ternary number can be obtained as $3^n - N$ where N is the ternary number and n is the number of trits. If the value of sign digit of a ternary number is 0 or 1 then the number can be taken as a positive number and if sign digit is 2 then the the number is negative.
For example $A = 02101$, $B = 11210$ are two positive numbers and $C = 21020$ is a negative number since it's sign trit is 2.

B. Balanced Ternary Number System

Now instead of using 0, 1 and 2, an alternative representation of the symbols may be as -1, 0 and 1 [2]. For simplicity, the symbol used for -1 is $\overline{1}$. Hereafter, we shall use the notations -1 and $\overline{1}$ to refer to the same value. The ternary number system with this set of symbols is known as a balanced ternary system [2]. Table I represents first few balanced ternary numbers $(B-T-N)$ and corresponding decimal numbers$(D-N)$. Some of the interesting properties of a balanced ternary number system include [2]:

1) The negative number is obtained by interchanging 1 and $\overline{1}$.
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.

C. Ternary Logic Gates

Logic AND,OR,NOT,XOR,$NAND$,NOR operation between two ternary variables and their truth table are shown in Table II and Table V respectively. In general there are three ternary inverter known as negative ternary inverter(NTI),standard ternary inverter(STI), positive ternary inverter(PTI)[14] these are defined in Table III. Similarly ternary NOR and $NAND$ circuits can be defined in three different ways as shown in Table IV and the corresponding truth table is shown in Table VI

$A\ \ AND\ \ B = min(A,B) = A \wedge B$
$A\ \ OR\ \ B = max(A,B) = A \vee B$
$NOT\ \ (A) = \overline{A} = 2 - A$
$A\ \ XOR\ \ B = (A+B)\ \ mod\ 3 = (A \bigoplus B)$
$A\ \ NAND\ \ B = not(min(A,B)) = \overline{A \wedge B}$
$A\ \ NOR\ \ B = not(max(A,B)) = \overline{A \vee B}$

TABLE II
TERNARY GATES

$InverterType$	$Output$
NTI	$Y_0 = 2\ \ \ if\ \ x = 0\ \ and\ \ Y_0 = 0\ \ otherwise$
STI	$Y_1 = \overline{x} = 2 - x$
PTI	$Y_2 = 0\ \ \ if\ \ x = 2\ \ and\ \ Y_0 = 2\ \ otherwise$

TABLE III
TERNARY INVERTERS

$Type$	$Output$
$NTNOR$	$Y_0 = NTI(max(A,B)) = NTI(A \vee B)$
$STNOR$	$Y_1 = STI(max(A,B)) = \overline{A \vee B}$
$PTNOR$	$Y_2 = PTI(max(A,B)) = PTI(A \vee B)$
$NTNNAND$	$Y_0 = NTI(max(A,B)) = NTI(A \wedge B)$
$STNAND$	$Y_1 = STI(max(A,B)) = \overline{A \wedge B}$
$PTNAND$	$Y_2 = PTI(max(A,B)) = PTI(A \wedge B)$

TABLE IV
TERNARY NOR AND $NAND$

a	b	$Y_{A \wedge B}$	$Y_{A \vee B}$	$\overline{(A)}$	$Y_{A \bigoplus B}$	$Y_{\overline{A \wedge B}}$	$Y_{\overline{A \vee B}}$
0	0	0	0	2	0	2	2
0	1	0	1	2	1	2	1
0	2	0	2	2	2	2	0
1	0	0	1	1	1	2	2
1	1	1	1	1	2	1	1
1	2	1	2	1	0	1	0
2	0	0	2	0	2	2	0
2	1	1	2	0	0	1	0
2	2	2	2	0	1	0	0

TABLE V
TRUTH TABLE FOR TERNARY GATES

a	b	Y_0^{NOR}	Y_1^{NOR}	Y_2^{NOR}	Y_0^{NAND}	Y_1^{NAND}	Y_2^{NAND}
0	0	2	2	2	2	2	2
0	1	0	1	2	2	2	2
0	2	0	0	0	2	2	2
1	0	0	1	2	2	2	2
1	1	0	1	2	0	1	2
1	2	1	2	1	0	1	0
2	0	0	0	0	2	2	2
2	1	0	0	0	0	1	2
2	2	0	0	0	0	0	0

TABLE VI
TRUTH TABLE FOR TERNARY NOR AND $NAND$ GATES

D. Ternary Full Adder

The following Table VII is the truth table of full adder. The expression for SUM is $A \bigoplus B \bigoplus C$.

a	b	cin	$cout$	sum
0	0	0	0	0
0	0	1	0	1
0	0	2	0	2
0	1	0	0	1
0	1	1	0	2
0	1	2	1	0
0	2	0	0	2
0	2	1	1	0
0	2	2	1	1
1	0	0	0	1
1	0	1	0	2
1	0	2	1	0
1	1	0	0	2
1	1	1	1	0
1	1	2	1	1
1	2	0	1	0
1	2	1	1	1
1	2	2	1	2
2	0	0	0	2
2	0	1	1	0
2	0	2	1	1
2	1	0	1	0
2	1	1	1	1
2	1	2	1	2
2	2	0	1	1
2	2	1	1	2
2	2	2	2	0

TABLE VII
TERNARY FULL ADDER

The expression for carry is $\overline{s} \wedge ((A \wedge B \wedge C) \vee (A \wedge B \wedge \overline{C}) \vee (A \wedge \overline{B} \wedge C)$ $\vee(\overline{A} \wedge B \wedge)) \vee (A \wedge B \wedge C)$

IV. APPLICATION OF TERNARY LOGIC IN COMPUTER SCIENCE

In the section we discuss application of ternary logic in different fields of computer science.

A. Application of ternary system in VLSI

There are several advantages of using ternary logic in VLSI circuits over the conventional Binary logic. For a fixed number of lines for transmitting information it is obvious that more information can be transmitted using ternary logic [14]. With the improvement of fabrication process the devices are scaled down but the scaling rate of interconnect is not same as that of the devices. As a result almost 60% of path delay is due to interconnects [3]. In a VLSI chip almost 60% to 70% area is covered with active devices and rest of the area has interconnects. This area leads to performance degradation [3]. Using ternary logic the number of interconnections can be reduced to $\frac{1}{log_2^3}$[20]. An energy efficient digital system can be designed using ternary logic as the complexity of interconnects and chip area can be reduced using this logic [14][15]. In electrical circuits power dissipation is mainly due to dynamic switching and current, and sub-threshold leakage current. About 80% of the total power is dissipated due to switching activity [16]. Average Power dissipation due to switching activity is given by $P_{avg} = \frac{1}{2}V_{dd}^2 Cf$,where V_{dd} is the supply voltage, C is the load capacitance and f is the frequency of operation. For aperiodic signals the frequency of operation can be estimated by the average number of signal transitions per unit time[16]. Using asynchronous ternary logic signal system the dynamic power is reduced to $P_{dyn} = (\frac{V_{dd}}{2})^2 Cf$ [17]. In this system, for the communication line at voltage level $\frac{V_{dd}}{2}$, it is in idle state. Thus in order to transmit one trit of information the voltage level is either high at V_{dd} or low at 0[17].

B. Application of Ternary logic in Computer Architecture

CPU is basically an instruction set processor.An instruction set defines the architecture of a processor.It is a interface between program and resources.A program is a sequence of instructions that performs a task.The quality of a processor is judged by the quality of instruction set and the quality of an instruction set is judged by the space it required and obviously the time required to interpret the instructions.Now suppose we are considering a 64 bit computer.
Now $3^{40} < 2^{64} < 3^{41}$. $\therefore$ $\frac{64-41}{64} \times 100\%$=35.93% space is reduced in base 3 computer in compare to base 2 computer.

C. Application of Ternary logic in Codding Theory

When we transmit information(i.e. a set of symbols$s_1, s_2, ..., s_q$) from here to there or from now to then the the problem of representing the source alphabet symbols s_i in terms of another system of symbols then the main problem of representation is the following

1) How to represent the source symbols so that their representation if far apart in some suitable sense. As a result in spite of small changes(noise),the altered symbols can be discovered to be wrong and even possibly corrected.
2) How to represent the source symbols in a minimal form for purposes of efficiency. The average code length, $L=\sum_{i=1}^{q} p_i l_i$ is minimized where l_i is the length of the representation of the i_{th} symbol s_i.

In some early days one variable length ternary code was popularly used for communication known as Morse code.Three different symbols of this code are dash(-),dot(.) and space( ).The length of the high frequency alphabet such as "E" is small and that of low frequency alphabet such as "J" is long. As a result the average length of the code is reduced.[6]

V. Arithmetic Operation on Ternary number

In this section we discuss arithmetic operations of ternary number systems such as shift operation, addition, subtraction, multiplication and division.

A. Shift Operation using balanced ternary number system

1) Arithmetic right shift operation.: In a given number A having n trits, let R_{n-1} and R_0 respectively denote the most significant (MST) and least significant trits (LST). Since in case of *balanced ternary system* the most significant non-zero $trit$ represents the sign, after arithmetic right shift the new MST becomes zero and the original LST is lost. If the LST of A is 1 then the arithmetic right shift of A yields $\lfloor \frac{A}{3} \rfloor$.

For example if A = 1011 (i.e. 31 in decimal), then arithmetic right shift we yield 0101 (i.e. 10 in decimal). If the LST of A is $\bar{1}$, then the arithmetic right shift of A yields $\lceil \frac{A}{3} \rceil$. Consider A = $101\bar{1}$, (i.e. 29 in decimal) then arithmetic right shift we get 0101(i.e. 10 in decimal). If the LST of A is 0 then the arithmetic right shift yields $\frac{A}{3}$. If A = 1010 (i.e. 30 in decimal) then arithmetic right shift we get 0101(i.e. 10 in decimal).

2) Arithmetic left shift operation.: In arithmetic left-shift operation of the number A an overflow *flip-flap-flop* [2] can be used to store the MST, and the new LST is 0. Arithmetic left shift operation yields $A \times 3$. For example if A = 0101 (i.e. 10 in decimal), then arithmetic left-shift operation yields 01010 (i.e. 30 in decimal). If the LST of A is $\bar{1}$, then the arithmetic right shift of A yields $\lceil \frac{A}{3} \rceil$. On the other hand, if A = $0\bar{1}0\bar{1}$ (i.e. -10 in decimal), then arithmetic left-shift operation yields $0\bar{1}0\bar{1}0$ (i.e. -30 in decimal).

B. Shift Operation using conventional ternary number system

1) Arithmetic right shift operation.: With this system after arithmetic right shift operation$(Ashr)$ LST is lost. If the MST is 2 before shift operation then after $Ashr$ it will be 2 otherwise after $Ashr$ MST will be 0. If the LST is 1 or 2 then the $Ashr$ of a number A yields $\lfloor \frac{A}{3} \rfloor$.If the LST is 0 then $Ashr$ yields $\frac{A}{3}$.
If A=2120(i.e. $(-12)_{10}$) then $Ashr$ yields 2212 (i.e. $(-4)_{10}$).If A=2121(i.e. $(-11)_{10}$) then $Ashr$ yields 2212 (i.e. $(-4)_{10}$).
If A=1120(i.e. $(42)_{10}$) then $Ashr$ yields 0112 (i.e. $(14)_{10}$).If A=1121(i.e. $(43)_{10}$) then $Ashr$ yields 0112 (i.e. $(14)_{10}$).

2) Arithmetic Left shift operation.: With this system after arithmetic left shift operation$(Ashl)$ the LST become zero. If $R_{n-1}R_{n-2}$ are 00, 01 or 22 then after $Ashl$ MST is lost otherwise a one $trit\ flip-flap-flop$ is needed to store the initial MST.Arithmetic left shift operation yields $A \times 3$.
If A=121 (i.e. $(16)_{10}$) then $Ashl$ yields 1210 ($(48)_{10}$).If A=2121 (i.e. $(-11)_{10}$) then $Ashl$ yields 221210 ($(-33)_{10}$).
If A=2221 (i.e. $(-2)_{10}$) then $Ashl$ yields 2210 ($(-6)_{10}$).

C. Addition and subtraction of balanced ternary numbers

1) Addition of two balanced ternary numbers: The following Table VIII 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.

1	1	1	1	1	1	1	1	1	0	0
1	1	1	0	0	0	$\bar{1}$	$\bar{1}$	$\bar{1}$	1	1
1	0	$\bar{1}$	1	0	$\bar{1}$	1	0	$\bar{1}$	1	0
10	$1\bar{1}$	01	$1\bar{1}$	01	00	01	00	$0\bar{1}$	$1\bar{1}$	0 1

TABLE VIII
Examples of Addition of $trits$

A	$111\bar{1}(38)$	$11\bar{1}\bar{1}(32)$	$1111(40)$	$1010(30)$
B	$1111(40)$	$\bar{1}\bar{1}0\bar{1}(-37)$	$1111(40)$	$\bar{1}\bar{1}\bar{1}\bar{1}(-40)$
$A+B$	$100\bar{1}0(78)$	$0\bar{1}11(-5)$	$1000\bar{1}(80)$	$0\bar{1}0\bar{1}(-10)$

TABLE IX
Examples of Addition of two 4 trits numbers

Now using this table one can easily add two n trits numbers. The following Table IX shows few example of addition of two 4 trits numbers.

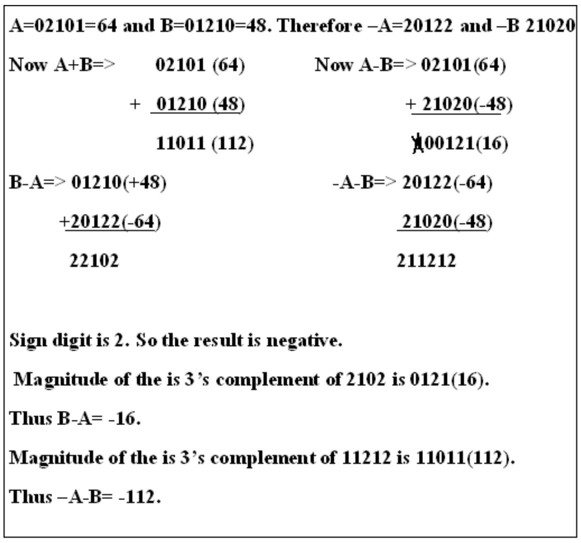
2) Subtraction of two balanced ternary numbers: The operation of Subtraction can be viewed simply as negation of a number followed by addition.Few examples of subtraction between two 4 trits numbers are shown in Table X.

A	$111\bar{1}(38)$	$11\bar{1}\bar{1}(32)$	$1111(40)$	$1010(30)$
$-B$	$\bar{1}\bar{1}\bar{1}\bar{1}(-40)$	$1101(37)$	$\bar{1}\bar{1}\bar{1}\bar{1}(-40)$	$1111(40)$
$A-B$	$001\bar{1}(-2)$	$10\bar{1}\bar{1}0(69)$	$0000(0)$	$10\bar{1}\bar{1}1(70)$

TABLE X
Examples of Subtraction between two 4 trits numbers

D. Addition and subtraction of two conventional ternary numbers

For addition of conventional ternary numbers we have to use the truth table for full adder as shown in Table VII.For substation(A-B) we have to take 3's complement of B and add it to A.3's complement of a number can be easily obtained by interchanging 0 and 2 followed by add 1 to it.
The Figure 1 shows few examples of addition and subtraction of two conventional ternary numbers.



A=02101=64 and B=01210=48. Therefore –A=20122 and –B 21020

Now A+B=> 02101 (64)
+ 01210 (48)
11011 (112)

Now A-B=> 02101(64)
+ 21020(-48)
$\cancel{1}$00121(16)

B-A=> 01210(+48)
+20122(-64)
22102

-A-B=> 20122(-64)
21020(-48)
211212

Sign digit is 2. So the result is negative.

Magnitude of the is 3's complement of 2102 is 0121(16).

Thus B-A= -16.

Magnitude of the is 3's complement of 11212 is 11011(112).

Thus –A-B= -112.

Fig. 1. Addition and subtraction of conventional ternary numbers

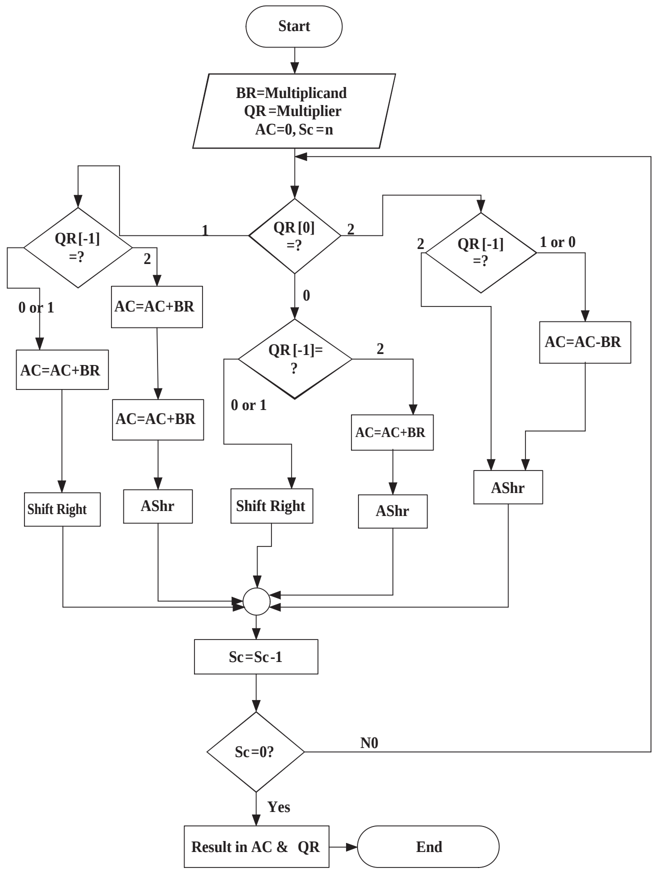


Fig. 2. Flowcharts for multiplication Algorithm for two conventional ternary numbers

VI. Hardware algorithm for multiplication and division of two base-3 numbers

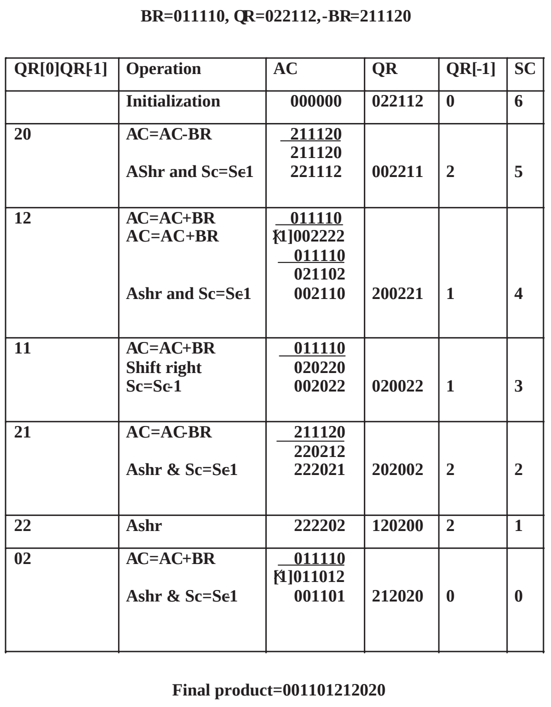
In this Section we describe multiplication and division of two ternary numbers.

A. Multiplication Algorithm using conventional ternary numbers

A ternary number with a stream of 2 can be expressed as $3^{m+1} - 3^k$ where k is the starting position of 2 and m is the ending position of 2 and counting is start from number 0.As for example $2220 = 3^4 - 3^1$.and $2221 = 3^4 - 3^1 + 1$.
$2022 = 3^4 - 3^3 + 3^2 - 3^0$.$2122 = 3^4 - 3^3 + 3^2 \times 1 + 3^2 - 3^0$.
If the number contain only 0 and 1 as symbols then we take the usual expression $\sum_{i=0}^{n-1} a_i 3^i$ where$a_i = 0, 1$;
Arithmetic left shift of a number is three times the number.So at the time of multiplication if the multiplier contains stream of $2s$ we simply Arithmetically left shift the multiplicand.
For multiplication we store multiplicand in a register BR, say, and Multiplier in register QR, say. 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. Now, if the trit of the multiplier is 1 then multiplicand is added 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. In case of hardware multiplication (using registers), instead of shifting the $multiplicand \times c$ (where c is a trit of the multiplier to the left we shift the partial product one trit to the right [21] [22].
The multiplication algorithm is based on the following rules

1) if $QR[0]QR[-1] = 22$ or 00 then partial product is arithmetically right shifted.
2) $QR[0]QR[-1] = 20$ or 21 then multiplicand is subtracted from from the partial product followed by $AShr$.
3) $QR[0]QR[-1] = 00$ or 01 then the partial product is simply right shifted.
4) $QR[0]QR[-1] = 02$ then multiplicand is added with the partial product followed by $AShr$.
5) $QR[0]QR[-1] = 10$ or 11 then multiplicand is added with the partial product followed by simple right shift.
6) $QR[0]QR[-1] = 12$ then multiplicand is added twice to the partial product arithmetic right shift.

The entire multiplication operation is shown in Figure 2 and the example for multiplication shown in Figure 3.



BR=011110, QR=022112,-BR=211120

QR[0]QR[-1]	Operation	AC	QR	QR[-1]	SC
	Initialization	000000	022112	0	6
20	AC=AC-BR AShr and Sc=Sc-1	211120 211120 221112	002211	2	5
12	AC=AC+BR AC=AC+BR Ashr and Sc=Sc-1	011110 [1]002222 011110 021102 002110	200221	1	4
11	AC=AC+BR Shift right Sc=Sc-1	011110 020220 002022	020022	1	3
21	AC=AC-BR Ashr & Sc=Sc-1	211120 220212 222021	202002	2	2
22	Ashr	222202	120200	2	1
02	AC=AC+BR Ashr & Sc=Sc-1	011110 [1]011012 001101	212020	0	0

Final product=001101212020

Fig. 3. Example of multiplication of two conventional ternary numbers

B. Multiplication Algorithm using balanced ternary numbers

For multiplication we store multiplicand in a register BR, say, and Multiplier in register QR, say. 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 bit of the multiplier is 1 then multiplicand is added with the partial product to generate a new partial product. Now the next bit 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. But in case of hardware multiplication (using registers), instead of shifting the $multiplicand \times c$ (where c is a trit of the multiplier, having value 0 or 1 or $\overline{1}$) to the left we shift the partial product one trit to the right. This operation has been defined for trits in [2].The entire operation is shown in Figure 4.

Lemma 1. *If a and b are two ternary numbers such that a is minimum and b is maximum then $b < 3 \times a$.*

Proof: Let a=10...0=3^{n-1} and b=222...2=$2\times\sum_{i=0}^{n-1} 3^i$. Now $\frac{a}{b}$=$\frac{2\times\sum_{i=0}^{n-1} 3^i}{3^{n-1}}$=$2+\frac{2\times\sum_{i=0}^{n-2} 3^i}{3^{n-1}}$=$2+\frac{(\sum_{i=1}^{n-1} 3^i-1)}{3^{n-1}}$. Now $1 < \frac{\sum_{i=1}^{n-1} 3^i}{3^{n-1}} < 2$. $\therefore \frac{a}{b} < 3$. $\therefore b < 3 \times a$. ■

C. Division Algorithm using conventional ternary number system

In order to divide a number by another, we store the dividend in register Q and divisor in register M. During division we take a set of trits of dividend and if it has a value less than that of the divisor, then we have to take another trit of dividend and insert 0(zero) in the quotient. On the other hand, if the value of a set of trits of dividend is greater than or equal to the value of the divisor, then either 1 or 2(using lemma 1) is inserted in the quotient. For this we have to subtract the divisor from the trits of the dividend; if the result is negative we put 1 in the quotient and add divisor to the result to restore those trits of dividend. This is known as *restoration of the dividend*. If the result of subtraction is positive then quotient is 2.The entire division operation is illustrated in the flow chart of Figure 5 .The example for division using conventional ternary number system is shown in Figure 6.

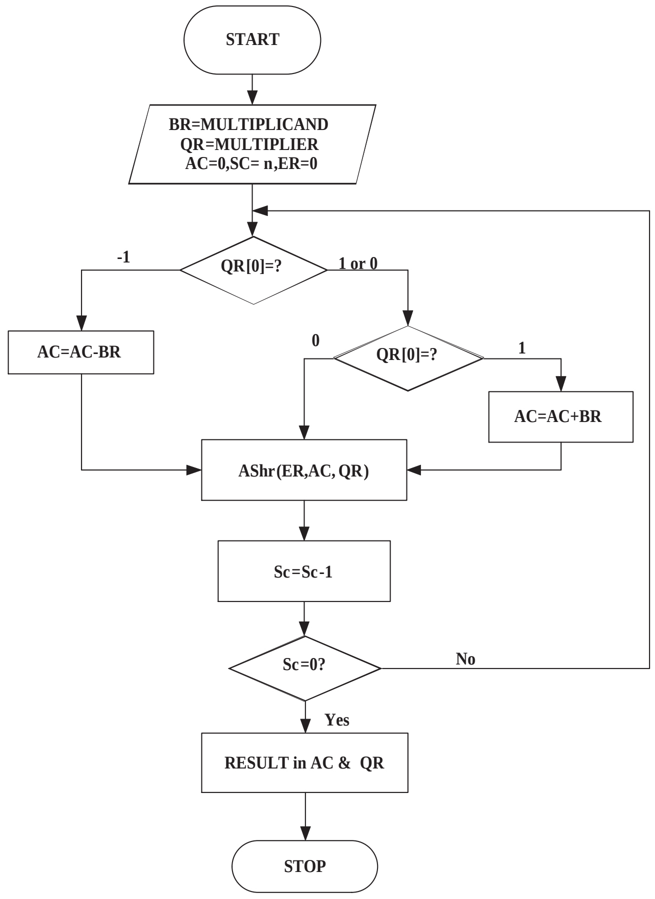


Fig. 4. Flow chart for multiplication of two balanced ternary number

Lemma 2. *If a and b are two balanced ternary numbers such that a is minimum and b is maximum then $b < 3 \times a$.*

Proof: Let a=$1\overline{1}...\overline{1}$=$3^{n-1}$-$\sum_{i=0}^{n-2} 3^i$ and b=111...1=$\sum_{i=0}^{n-1} 3^i$. Now $\frac{a}{b}$=$\frac{\sum_{i=0}^{n-1} 3^i}{3^{n-1}-\sum_{i=0}^{n-2} 3^i}$. Now $\sum_{i=0}^{n-2} 3^i < 3^{n-1}$-$\sum_{i=0}^{n-2} 3^i$. $\therefore \frac{a}{b} < 3$. $\therefore b < 3 \times a$. ■

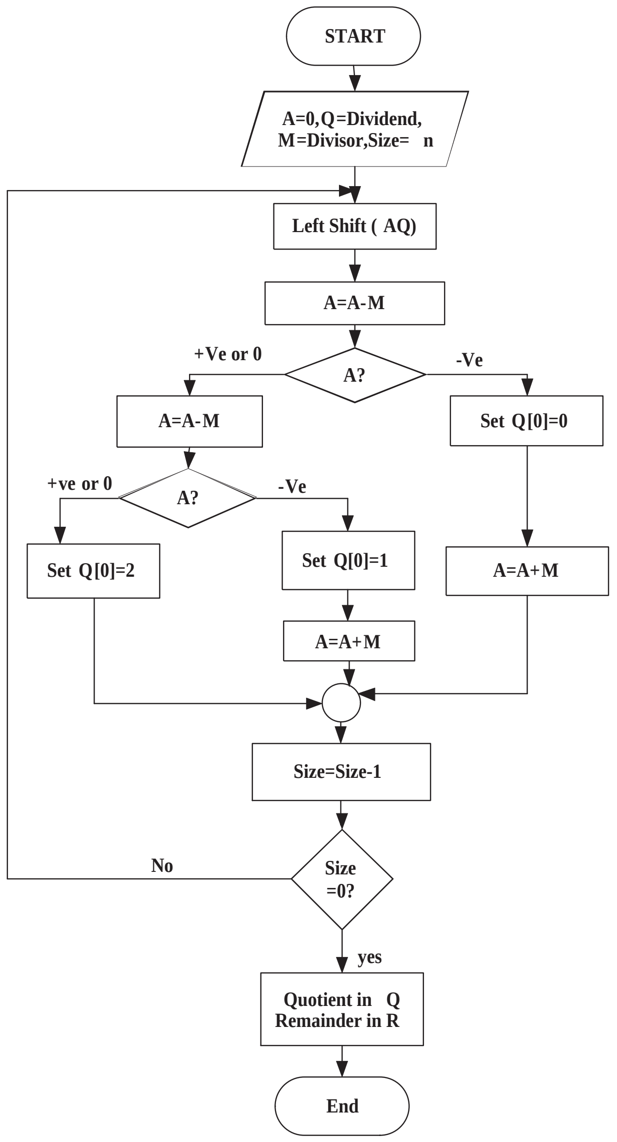


Fig. 5. Flowcharts for Division Algorithm for two non negative numbers using conventional ternary number system

Here we divide $(101211)_3$ by $(201)_3$ i.e. 292 by 192 in decimal. –M=222022

Operation	A	Q	Sc
Initialization	000000	101211	6
Left Shift AQ	000001	01211[]	
A=A-M	222022		
A is -Ve Set Q[0]=0	222100	011211[0]	
A=A+M	000201		
Size=Size-1	[~~1~~]000001		5
Left Shift AQ	000010	1211[0][]	
A=A-M	222022		
A is -Ve Set Q[0]=0	222102	1211[0][0]	
A=A+M	000201		
Size=Size-1	[~~1~~] 000010		4
Left Shift AQ	000101	211[0][0][]	
A=A-M	222022		
A is -Ve Set Q[0]=0	222200	211[0][0][0]	
A=AM	000201		
Size=Size-1	[~~1~~]000101		3
Left Shift AQ	001012	11[0][0][0][]	
A=A-M	222022		
A is +Ve	[~~1~~]000111		
A=A-M	222022	11[0][0][0][1]	
A is –ve Set Q[0]=1	222210		
A=A+M	000201		2
Size=Size-1	[~~1~~]000111		
Left Shift AQ	001111	1[0][0][0][1][]	
A=A-M	222022		
A is +Ve	[~~1~~]000210		
A=A-M	222022		
A is +ve Set Q[0]=2	[~~1~~]000002	1[0][0][0][1][2]	
Size=Size-1			1
Left Shift AQ	000021	[0][0][0][1][2][]	
A=A-M	222022		
A is -Ve Set Q[0]=0	222120		
A=A+M	000201	[0][0][0][1][2][0]	0
Size=Size-1	[~~1~~]00002 1		

Remainder A=00021 and Quotient Q=000120.

Fig. 6. Example for Division Algorithm for two non negative numbers using conventional ternary number system

D. Division Algorithm using balanced ternary number system

The division of two nonnegative ternary numbers is discussed in[21] and the flow chart for that algorithm is shown in Figure 7.Here we describe the algorithm when the dividend is negative.In this case instead of subtracting the divisor from the set of trits of dividend is added with the set of trits of dividend.If the result is positive 0 is inserted in the quotient and divisor is subtracted from the result to get back the previous value. This is known as restoration of dividend. If the result of addition between divisor and set of trits of dividend is negative the either $\overline{1}$ $(i.e. \ \ -1 \ \ in \ \ decimal)$ or $\overline{1}1$ $(i.e.is-2 \ \ in \ \ decimal)$ (using lemma 2)is inserted in the quotient.For this if the result of addition is negative then first $\overline{1}$ is inserted in the quotient then again the divisor is added with the partial result and if the result of this addition is negative or 0 then 1 is inserted in the quotient other wise divisor is subtracted from the last result to restore back the previous result.When all the trits are encountered then if the value of register A is negative then divisor is added with A and the result store in register Q is decremented by 1.The division operation is illustrated in the flow chart of Figure 8 and the corresponding example is shown in Figure 9 . Now using these two algorithms division operation can be easily performed for both negative and nonnegative dividend.

VII. Performance Analysis

A. Multiplication Algorithm

In case of multiplication algorithm using conventional ternary number system the complexity of the algorithm will be as follows

1) If the multiplier is $222...2$ then number of addition/subtraction operation is only one and number of shit operation is n.So Complexity in this case is $O(n)$.
2) If the multiplier is $000...0$ then number of addition/subtraction operation is zero and number of shit operation is n.So Complexity in this case is $O(n)$.
3) For any other multiplier the number of addition/subtraction operation is $O(n)$ and number of shit operation is $O(n)$.So Complexity in this case is $O(n^2)$.

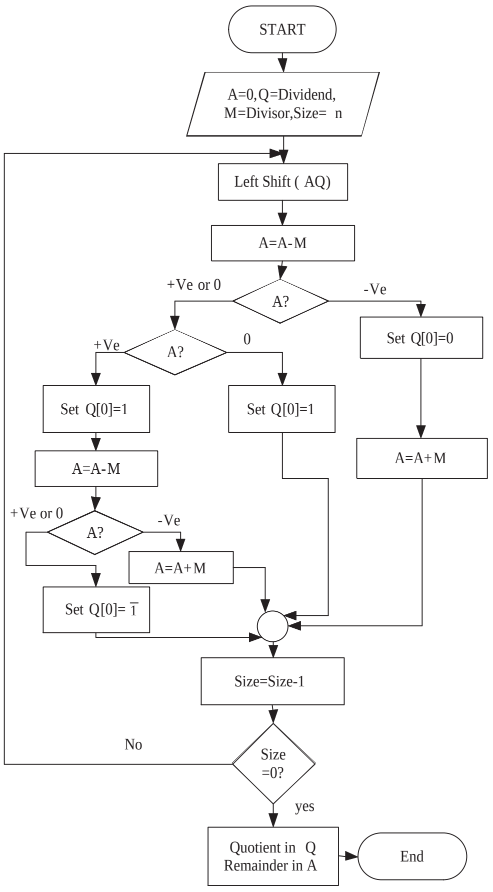


Fig. 7. Flowcharts for Division Algorithm for two non negative numbers

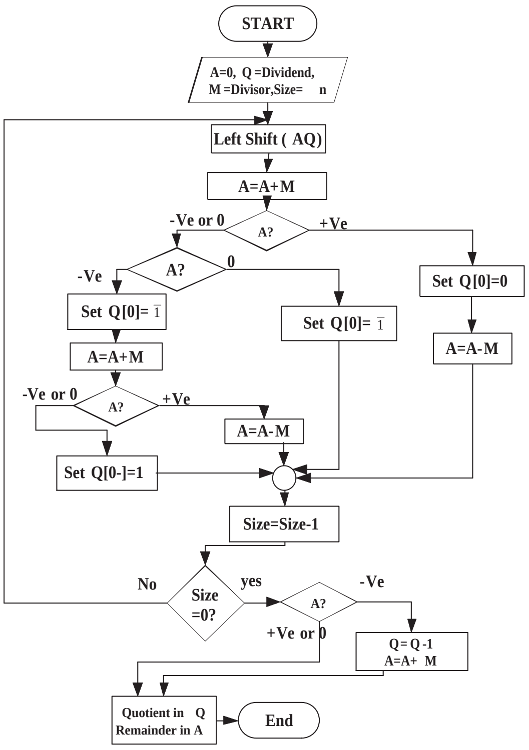
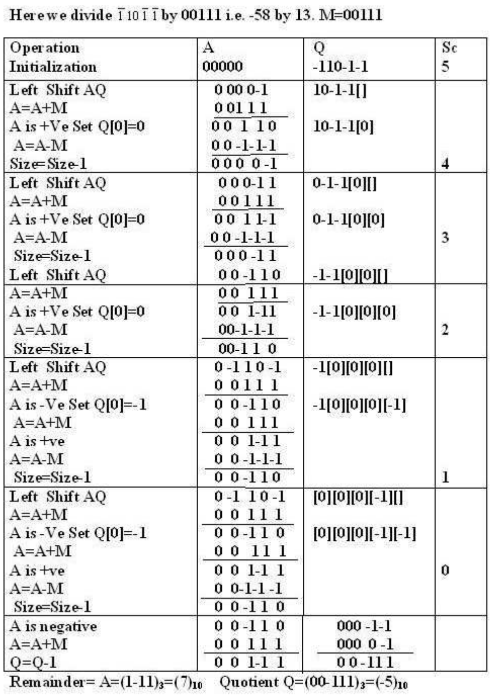


Fig. 8. Flowcharts for Division Algorithm when quotient is negative


Here we divide $\bar{1}10\bar{1}\bar{1}$ by 00111 i.e. -58 by 13. M=00111

Operation	A	Q	Sc
Initialization	00000	-110-1-1	5
Left Shift AQ	0 00 0-1	10-1-1[]	
A=A+M	0 01 1 1		
A is +Ve Set Q[0]=0	0 0 1 1 0	10-1-1[0]	
A=A-M	0 0 -1-1-1		
Size=Size-1	0 0 0 0 -1		4
Left Shift AQ	0 0 0-1 1	0-1-1[0][]	
A=A+M	0 0 1 1 1		
A is +Ve Set Q[0]=0	0 0 1 1-1	0-1-1[0][0]	
A=A-M	0 0 -1-1-1		3
Size=Size-1	0 0 0 -1 1		
Left Shift AQ	0 0 -1 1 0	-1-1[0][0][]	
A=A+M	0 0 1 1 1		
A is +Ve Set Q[0]=0	0 0 1-11	-1-1[0][0][0]	
A=A-M	00-1-1-1		2
Size=Size-1	00-1 1 0		
Left Shift AQ	0 -1 1 0 -1	-1[0][0][0][]	
A=A+M	0 0 1 1 1		
A is -Ve Set Q[0]=-1	0 0 -1 1 0	-1[0][0][0][-1]	
A=A+M	0 0 1 1 1		
A is +ve	0 0 1-1 1		
A=A-M	0 0 -1-1-1		
Size=Size-1	0 0 -1 1 0		1
Left Shift AQ	0 -1 1 0 -1	[0][0][0][-1][]	
A=A+M	0 0 1 1 1		
A is -Ve Set Q[0]=-1	0 0 -1 1 0	[0][0][0][-1][-1]	
A=A+M	0 0 1 1 1		
A is +ve	0 0 1-1 1		0
A=A-M	0 0-1-1 -1		
Size=Size-1	0 0 -1 1 0		
A is negative	0 0 -1 1 0	000 -1-1	
A=A+M	0 0 1 1 1	000 0 -1	
Q=Q-1	0 0 1-1 1	0 0 -1 1 1	

Remainder= A=$(1\text{-}11)_3=(7)_{10}$ Quotient Q=$(00\text{-}111)_3=(-5)_{10}$

Fig. 9. Example for Division Algorithm when quotient is negative

However, for multiplication using balanced ternary number system the complexity is $O(n^2)$. Moreover, using conventional number system maximum range of n-trit numbers is $3^n - 1$ but in case of balanced ternary number system maximum is $3^{n-1} - 1$.The advantage of using balanced ternary number system subtraction operation can be easily performed.

B. Division Algorithm

The complexity of the division algorithm for both conventional and balanced ternary number system is $O(n^2)$. However, in case of balanced ternary number system some extra trits in QR (flip-flap-flop) may be required when $2(1\bar{1})$ is inserted into quotient.

VIII. Rotation Symmetric Boolean Function in Base-3

Rotation symmetric Boolean functions ($RSBF$) have huge application in cryptosystem. A Boolean function is symmetric if it is invariant under any permutation of its variables [4]. A Boolean function $f(.)$ of n variables is rotation symmetric if and only if $f(x_{n-1}, x_{n-2}, \ldots, x_0) = f(x_{n-2}, x_{n-3}, \ldots, x_0, x_{n-1}) = f(x_0, x_{n-1}, \ldots, x_1)$. Now rotation symmetric Boolean functions are a class of Boolean functions which have good combination of non-linearity, co-relation immunity, balancedness and algebraic degree [12]. A Boolean function is applicable to cryptography if it has the above mentioned properties. Now since base three is optimal, it is quite expected that the Boolean function in GF(3) [2] is better. In this paper, we propose an algorithm to generate the partitions of rotation symmetric Boolean functions where a *partition* is a set of a trit string and the rotations of this string, such that the output of each of these strings as input provides the same output.Generation of these functions are known to be combinatorially explosive. It is known that, for n-variable *RSBF* functions, the associated set of input bit strings can be divided into a number of subsets (called *partitions*), where every element of a subset can be obtained by simply rotating the string of bits of some other element of the same set.Formula for generating the partitions for Rotation Symmetric Boolean Function in any base $g_{n,p} = \frac{1}{n}\sum_{t|n}\phi(t)p^{\frac{n}{t}}$[5]. Figure 10 shows the partitions generated for n = 4.

Definition 1. If a Boolean function $f(x_{n-1}, x_{n-2}, \ldots, x_0)$ exhibits rotation symmetry, then the period over which its exhibits this property is defined to be the cycle length for the function.

{(0000)}	partition 0
{(0001) (0010) (0100) (1000)}	partition 1
{(0002) (0020) (0200) (2000)}	partition 2
{(0011) (0110) (1101) (1011)}	partition 3
{(0012) (0120) (1200) (2001)}	partition 4
{(0021) (0210) (2100) (1002)}	partition 5
{(0022) (0220) (2200) 2002}	partition 6
{(0101) (1010)}	partition 7
{(0102) (2010) (0201) (1020)}	partition 8
{(0111) (1110) (1101) (1011)}	partition 9
{(0112) (1120) (1201) (2011)}	partition 10
{(0121) (1210) (2101) (1012)}	partition 11
{(0122) (1220) (2201) (2012)}	partition 12
{(0202) (2020)}	partition 13
{(0211) (2110) (1102) (1021)}	partition 14
{(0212) (2120) (1202) (2021)}	partition 15
{(0221) (2210) (2102) (1022)}	partition 16
{(0222) (2220) (2202) (2022)}	partition 17
{(111111)}	partition 18
{(1112) (1121) (1211) (2111)}	partition 19
{(1122) (1221) (2211) (2112)}	partition 20
{(1212) (2121)}	partition 21
{(1222) (2221) (2212) (2122)}	partition 22
{(2222)}	partition 23

Fig. 10. Partitions of $RSBF$ for n = 4

The proposed algorithm for the generation of partitions of $RSBF$s in GF(3) is given below. This algorithm generates the starting string of each partition and and total numbers of partitions. We use the symbols 0, 1, 2 to represents the numbers instead of using 0,1,$\overline{(1)}$. Table 2 shows the number of partitions for different values of n (number of $trits$). A formal description of the proposed algorithm is given in Figure 11.

1) Correctness of the proposed algorithm genpartA: As the successive numbers are being considered, thus if the number of rotations of any number is less than the number of trits in that number, then the number must have appeared before in some element of some partition.

A. An improved Algorithm

In this paper we propose two algorithms for generating the partition of $RSBF$ in $GF(3)$.Algorithm genpartB and genpartC as described below is an improved version of algorithm genpartA and the formal description of the algorithm is shown in Figure 12.

Lemma 3. *If a is a ternary number then right rotation of a and $a+3$ yield successive numbers.*

Proof: Let, a=$(a_{n-1}a_{n-2}...a_1a_0)_3$
b=$a+3$=$(b_{n-1}b_{n-2}...b_1b_0)_3$
$ROR\ of\ a\ by\ 1\ trit$=$(a_0a_{n-1}a_{n-2}...a_2a_1)_3$ and
$ROR\ of\ b\ by\ 1\ trit$=$(b_0b_{n-1}b_{n-2}...b_2b_1)_3$
=$3^{n-1}\times b_0 + 3^{n-2}\times b_{n-1} + 3^{n-3}\times b_{n-2} + ... + 3^1b_2 + 3^0b_1$
$= \frac{3^nb_0+(3^{n-1}b_{n-1}+3^{n-2}b_{n-2}+...+3^2b_2+3^1b_1)}{3}$
$= \frac{3^nb_0+(3^{n-1}b_{n-1}+3^{n-2}b_{n-2}+...+3^2b_2+3^1b_1+b_0-b_0)}{3}$
$= \frac{3^nb_0+(3^{n-1}b_{n-1}+3^{n-2}b_{n-2}+...+3^2b_2+3^1b_1+3^0b_0)-b_0}{3}$
$= \frac{3^na_0+((3^{n-1}a_{n-1}+3^{n-2}a_{n-2}+...+3^2a_2+3^1a_1+3^0a_0)+3)-a_0}{3}$ (since b=a+3 and $b_0 = a_0$)
$= \frac{3^na_0+3^{n-1}a_{n-1=}+3^{n-2}a_{n-2}+...+3^2a_2+3^1a_1+3^0a_0+3-a_0}{3}$
=$3^{n-1}a_0 + 3^{n-2}a_{n-1} + 3^{n-3}a_{n-2} + ... + 3^1a_2 + 3^0a_1 + 1$
=$ROR\ of\ a\ by\ 1\ trit$ ■

Lemma 4. *Two $trits\ ROR$ of $\{(0^{n-2}2^2)_3 + (p\times 9)_{10}\}$ and $\{(0^{n-2}2^2)_3 + ((p+1)\times 9)_{10}\}$ yields successive numbers.where p is is any integer including 0.*

Proof: Two $trits\ ROR$ of $\{(0^{n-2}2^2)_3 + ((p+1)\times 9)_{10}\}$-two $trits\ ROR$ of $\{(0^{n-2}2^2)_3 + (p\times 9)_{10}\}$ =two $trits\ ROR$ of $\{(0^{n-2}2^2)_3 + (p\times 9)_{10}\}$ +two $trits\ ROR$ of $\{9_10\}$-two $trits\ ROR$ of $\{(0^{n-2}2^2)_3 + (p\times 9)_{10}\}$
=two $trits\ ROR$ of $\{(9)_{10}\}$
=two $trits\ ROR$ of $\{(100)_3\}$
=$\{(001)_3\}$
Hence the proof. ■

1) Correctness of the proposed algorithm genpartB:

1) From lemma 3 it is obvious that if a and $a+3$ are two ternary numbers then ROR of a and $a+3$ yields successive numbers.Now 1 $trit$ ROR of $0^{n-1}1^1$ is 1^10^{n-1} and 1trit ROR of $02^{n-2}1^1$ is 102^{n-2}.
Again $02^{n-2}1^1$=$2\times 3^{n-2} + 2\times 3^{n-3} + ... + 2\times 3^1 + 1\times 3^0$
=$1 + (m-1)\times 3$ where m=$3^{n-2} - 1$.
So as soon as the starting string of a partition become $02^{n-2}1^1$ using lemma 3 it is obvious that the algorithm generates all the number between 1^10^{n-1} and $1^10^12^{n-2}$.So the number between 1^10^{n-1} $1^10^12^{n-2}$ can't be the starting string of any partition and can be skipped.
2) 1^12^{n-1} and 2^01^{n-1} are two successive numbers.Now lemma 3 indicates that we have all the orbits having elements from 2^01^{n-1} and $2^11^12^{n-2}$.So when the starting string of a orbit becomes 1^12^{n-1} next

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
g_n	3	6	11	24	51	130	315	834	2195	5934	16107	44368	122643	341802	956635	2690844

TABLE XI
NO. OF ORBITS FOR 1^{st} 16 VARIABLES

Algorithm genpartA

Data structures: Counter = Number of partitions, Answer[]=Starting string of partition
Input: Number of trits
Output: Starting string of every orbit and total number of orbits

```
Step 1: Initialization: Counter=0;
Step 2: $Answer[0]$ = $0^n$;
Step 3: Counter=counter +1;
Step 4: Consider the trit-string $s$ corresponding to the next number store its decimal form in Answer[].
Step 5: While trit-string corresponding to $s$ = $\{2^n\}$ do
Step 6:     While a number $p$ is less than of any element $pc$ of that particular orbit then goto step 8.
Step 7:         Take the trit-string corresponding to $s$ as the starting string of the next orbit
Step 8:         Take the trit-string corresponding to next number
Step 9:     End while
Step 10:End While
Step 11: End
```

Fig. 11. Generation of partitions of Rotation symmetric Boolean function in GF(3)

Algorithm genpartB()

Data structures: Counter = Number of partitions, Answer[]=Starting string of partition
Input: Number of trits
Output: Starting string of every orbit and total number of orbits

```
Step 1: Initialization: Counter=0;
Step 2: $Answer[0]$ = $0^n$;(*$a^n$ means a string of $n$ trits*)
Step 3: Counter=counter +1;
Step 4: Consider the trit-string $s$ corresponding to the next number store its decimal form in Answer[].
Step 5:     While trit-string corresponding to $s$ = $\{1^1 2^{n-1}\}$ do
Step 6:         While a number $p$ is less than of any element $pc$ of that particular orbit then goto step 9.
Step 7:             If the next number becomes $1^1 0^{n-1}$ then next trit string must be greater than $1^1 0^1 2^{n-2}$
Step 8:             Take the trit-string corresponding to $s$ as the starting string of the next orbit
Step 9:             Take the trit-string corresponding to next number
Step 10:            Counter=counter+1
Step 11:        End while
Step 12: End While
Step 13:Counter=Counter+1
Step 14:Answer=$2^n$
Step 15: End
```

Fig. 12. Generation of partitions of Rotation symmetric Boolean function in GF(3)

starting string must be greater than $2^1 1^1 2^{n-2}$.Now $2^1 1^1 2^{n-2}$ and 2 trit ROR of $0^{n-2}2^2$ are two successive numbers.Now Decimal of $(1^1 2^{n-1})_3$=decimal of $(0^{n-2}2^2)_3$+$(m-1) \times 9$,where m is an integer.Now two trits ROR of $1^1 2^{n-4}$ is $2^2 1^1 2^{n-2}$. $\therefore$ using lemma 4 it is easy to understand that all the numbers between $2^2 0^{n-3}$ and $2^2 1^1 2^{n-2}$ appears in some partitions.Using similar types of observation it can be easily shown that if the starting string of any partition become $1^1 2^{n-1}$ then all the numbers before $2^{n-1}1$ appears in some partition.So the next starting string must be 2^n.

B. Another Improved Version of the Algorithm genpartB

A still improved version of algorithm is shown in Figure 13.

1) Correctness of the algorithm genpartC(): The correctness of the algorithm genpartC can be easily understood from correctness of algorithm genpartA and genpartB.

C. Performance analysis of the algorithms

The complexity of the above algorithm is $\bigcirc(3^n)$. Since the formula for number of partitions is exponential, the time complexity of the above algorithm is inherently exponential. But actual number of comparisons are different in different algorithms.
For the algorithm genpartA total number of comparison is $3^n - 1$. For the algorithm genpartB total number of comparison is $3^n - 1$-A-B.
A=$\{3^n$-
$(3^{n-1} + 2 \times 3^{n-2} + 2 \times 3^{n-3} + ... + 2 \times 3^1 + 2 \times 3^0)\}$
B=$\{(3^{n-1} + 2 \times 3^{n-3} + 2 \times 3^{n-4} + ... + 2 \times 3^1 + 2 \times 3^0)$-$3^{n-1}\}$. For the third algorithm total number of comparison is much less than that of the previous two algorithms.Here total number of comparison just few more than the number of partition of $RSBF$.

```
Algorithm genpartC()
Data structures: Counter = Number of partitions, Answer[]=Starting string of partition
Input: Number of trits
Output: Starting string of every orbit and total number of orbits
1. Initialization: Counter=0,number1=$0^{n-1}1^1$ and number2=$0^{n-1}2^1$;
2. $Answer[0]$ = $0^n$; (*$a^n$ means a string of $n$ trits*)
3. Counter=counter +1;
4. while trit-string corresponding to $s$ = $\{1^1 2^{n-1}\}$ do
5. number1=number1+3 and number2=number2+3
6. While the starting of any partition(i.e. number1 and number2) is less than any element of the particular orbit then goto step 8
7.     If the next number becomes $\{1^1 0^{n-1}\}$ then number1=$\{1^1 0^1 2^{n-2}+1\}$ and number2=$\{1^1 0^1 2^{n-2}+2\}$
8.     Take the trit-string corresponding to number1=number1+3 and number2=number2+3
9.     Answer[counter++]=number1 and Answer[counter++]=number2
10.Endwhile
11. counter=counter+1
12.Answer[counter]=$\{2^n\}$
13.End
```

Fig. 13. Generation of partitions of Rotation symmetric Boolean function in GF(3)

IX. CONCLUSION

In this paper we discuss few algorithms for arithmetic (addition, multiplication, division and arithmetic shift) operations in the conventional and balanced ternary number systems. Some new algorithms have been proposed and their time-complexity analysis have been discussed. Algorithms for *Rotation Symmetric Boolean Function* is also proposed in this paper. There are ample scopes of further work in this number system such as using them in cryptographic systems, exploring their opportunities in efficient power management in VLSI circuits, and so on.

REFERENCES

[1] Brian Hayes, Third Base, *American scientist*, Vol. 89, No.6, 2001, pp. 490-494.

[2] D.E.Knuth, *The Art of Computer programming, Vol. 2*, Third edition, Pearson Education.

[3] Naveed A. Sherwani, Algorithms for VLSI Physical Design Automation .

[4] Zvi Kohavi, *Switching and Finite Automata Theory*, 2^{nd} Edition, Tata-McGraw Hill, 2008.

[5] Yuan Li, Results on Rotation Symmetric Polnomials Over GF(3),Elsevier Science Direct,Information Science178(2008)280-286.

[6] R.W.Hamming,Codding and Information Theory,Prentice-Hall,Inc.

[7] A, Srivastava, K Venkatapathy, "Design and implementation of a low power ternary full adder", *VLSI Design*, 1996, Vol.4, No.1, pp 75-81.

[8] A.Sathish Kumar, A. Swetha Priya, The Minimization of Ternary Combinational Circuits-A Survey,A. Sathish Kumar et.al., *International Journal of Engineering and Technology*, Vol. 2, No. 8, 2010, pp. 35376-3589.

[9] A.P.Dhande, V.T. Ingole, Design and Implementation of 2-Bit ternary ALU Slice, 3^{rd} *International Conference SETIT*, 2005, Tunisia.

[10] X.W.Wu, CMOS Ternary Logic Circuits, IEE Proceedings, Vol 137, Pt.G, No.1, Feb 1990.

[11] A Stakhov, Brousentsov's Ternary Principle, Bergman's Number System and Ternary Mirror-Symmetrical Arithmetic, *The Computer Journal*, Vol 45, No. 2, 2002.

[12] P. Sarkar,S. Mitra Constructions of nonlinear Boolean Functions with important cryptographic properties,Advances in cryptology,EUROCRYPT-2000,Lecture notes in Computer Science,Vol-1807,Berlin 2000,PP 485-506.

[13] J. Adikari, V. S. Dimitrov, L. Imbert, Hybrid Binary-Ternary Number System for Elliptic Curve Crypto System, *IEEE Transactions on Computers*, Vol. 60, No. 2, Feb 2011.

[14] P.C.Balla and A.Antoniou, Low Power dissipation of MOS Ternary logic Family, *IEEE Journal of Solid State Circuits*, Vol. Sc-19, No.5, 1994.

[15] S. Lin, Y-B Kim,F.Lombardi, CNTFET Based Design of Ternary Logic Gates and Arithmetic Circuits, *IEEE Transactions of Nanotechnology*, Vol.2, No.11, 2011.

[16] K. Roy, S.C. Prasad, *Low Power CMOS VLSI Circuit Design*, Wiley India, 2011.

[17] T.Felicijan,S.B.Furber, An Asynchronous Ternary Logic System, *IEEE Transaction on Very Large Scale Intregation System* , Vol.11, No.6, 2003.

[18] K.C.Smith,The prospects for multivalued logic: A technology and application View,*IEEE transactions on Computer*, Vol C-30, No.9, 1981.

[19] D.I.Porat, Three Valued Digital Systems, *Proc IEE*, Vol.116, No.6, 1969.

[20] Vasundara Patel, K S,K S Gurumurthy, Multi-valued Logic Addition and Multiplication in Galois Field, *IEEE International Conference on Advances in Computing,Control,and Telecommunication Technologies*, 2009.

[21] Subrata Das, Joy Prakash Sain, Parthasarathi Dasgupta and Samar Sensarma, Algorithms for Ternary Number System, 2^{nd} International Conference on Computer, Communication, Control and Information Technology, Elsevier Science Direct,Procedia technology,Vol-4,pp-278-285.

[22] Subrata Das,Parthasarathi Dasgupta and Samar Sensarma,Arithmetic Algorithms for Ternary Number System,VDAT-2012,Springer,LNCS,Vol-7373,pp111-120,2012.