TBNS Representation Of Any Real Number

Arnab Chowdhury[1], Moyukh Laha[2], Ayan Santra[3], Balmiki Roy[4], Biplab Bhadra[5], Subhasis Maitra[6]
Department of Electronics and Communication Engineering
Kalyani Government Engineering College
Kalyani, Nadia, West Bengal, India

***Abstract* – The basic aim of Multi Base Number System is to reduce the algebraic complexity and to enhance the speed of the existing algorithms. Both DBNS and TBNS have significant roles in achieving this. In this paper, we have introduced a few algorithms to find Triple Base Number System and Triple Base Chain. This in turn leads to the representation of any real number in TBNS. Introduction of Single Digit Triple Base Number System is another aspect of study in this paper. Analysis has been done in details in terms of time complexity, which prove the novelty of the algorithms.The variation of error with increase in the value of the integer is calculated. Efficient implementation of these algorithms will result in the design of an efficient architecture of a multiplier unit.The proposed algorithms are supposed to be efficient and can be widely used in the field of Digital Signal Processing (DSP) & Elliptic Curve Cryptography (ECC).**

Keywords – DBNS, TBNS, SDDBNS, SDTBNS, Real number representation, Error, DSP and ECC

I. INTRODUCTION

Low performance ALUs never look upon the problem of time complexity. High performance, flexibility, space complexity and low power consumptions are the most important issues in the current signal processing architectures. The signal processing algorithms face many challenges in real-time applications because of their high computational complexity. Therefore, the major issues have been the enhancement of speed of the arithmetic units.In general we can conclude that if the computational complexity of operations performed by an ALU, multiplications and additions in particular, can be reduced, it results in the design of a high performance ALU.

Representation of an integer in TBNS improved the performance of the ALU by reducing the complex computations. Now if we can express any real number in TBNS with a minimal error, it will be a great boon for the industry that deals with DSP & ECC. In this paper, we introduce a few algorithmsto convert any real numberinto Triple Base Number System. The basic principle of the first algorithm is that every integer is divided by 30 (expressed as $2^1 3^1 5^1$). We have designed another algorithm to convert any real number into Single Digit Triple Base Number System (SDTBNS). This is not the only algorithm to represent a floating point number in TBNS. We have approached with a more general algorithm employing the concept of place value.

II. EXISTING ALGORITHMS

Algebraic complexity of different Algorithms in Signal Processing and Cryptography leads to a major problem and Researchers are trying to develop new Algorithms to solve these problems. To enhance the speed of the existing Algorithms, different number systems have been found for point multiplication in elliptic curve cryptography (ECC) &coefficient multiplication in digital signal processing (DSP) mainly for digital filter design. Among the different number system, DBNS, DBC, HBTJSF, w-NAF are efficient. Recently, to increase the speed again, TBNS, SDTBNS have been developed. There are different method to convert any integer or fraction into TBNS and hence SDTBNS.

It all started with a number system which employed bases as 2 and 3, allowing as digits only 0, 1, and requiring O[log N] [1] nonzero digits, known as the 'Double Base Number System (DBNS)'. Any integer can be represented in DBNS.

The general form of representation is:

$$x = \sum_{i,j} d_{ij} p^i q^j \quad , p < q; 0 \le d_{ij} < p$$

Double base chains have been obtained with a greedy approach [8], relying on the search of the closest {2, 3} – integer to a given number. This approach tends to increase the length of the chain. Then tree based approach was introduced which is very well known to us now. The tree based approach [2] has been generalized in order to obtain other kinds of double base chains. Even though DBNS

schemes exhibit reasonably good performance for 8 bit multiplication, they are not efficient for higher bits and is highly redundant.

For further enhancement of the performance of arithmetic operations and to reduce the hardware complexities, a new concept called Triple Based Number Systems (TBNS) [11]. It was introduced for performance enhancement of the multiplier of the digital signal processors. We use 2, 3 and 5 to represent any integer in TBNS. Efficiency of this number system has been dealt already with in details and a comparison between TBNS and DBNS clearly indicate the advantages of the former in terms of speed, hardware complexity and power dissipation. Different algorithms are proposed. Algorithms like Joint Binary-Ternary Representation System (JBTRS), Triple Base Hybrid Joint Sparse Form (JSF) and Joint Double Base Chain (JDBC) [9] are important in terms of complexity, efficiency. From different analysis it is clear that JTBNS is only comparable with HBTNS, but HBTNS requires a pre computation look-up-table. Also the number of doublers required for HBTNS is more than that required for JTBNS at the cost of adders. But the complexity to design a doubler is more than to design an adder. Hence JTBNS is advantageous from all respect.

III. PROPOSED ALGORITHMS

Let us discus about the algorithms to represent any integer in TBNS.

Algorithm I

The algorithm is shown in the following steps -

1. Let us consider any integer ‘n’.
2. We then divide the number by 30.
3. Dividing any number by 30 means that it is either divisible by 30 or after dividing the number by 30 gives a quotient and a remainder that lies in the range of 1-29.
4. Out of the remainders in the range, we can easily represent 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25 and 27 using 2, 3, and 5.

   The remaining remainders that cannot be represented by 2, 3, and 5 are 7, 11, 13, 14, 17, 19, 21, 22, 23, 26, 28, and 29.

5. If 7, 11, 13 can be represented in TBNS then we can easily represent 14, 21, 22, 26, and 28 in TBNS since they can be obtained by multiplying 2^1 or 2^2 or 3^1.The remaining numbers (17, 19, 23, and 29) has to be represented in the same way.
6. To represent the remaining numbers in TBNS, we first check whether the remainder is divisible by 15, 10, 6, 5, 3, and 2 in descending order. That means that the divisibility of the remainder is first checked with 15, then 10 and this particular order is followed. Divisibility by these numbers signifies that the remainder can easily be represented using 2, 3, and 5. The reason behind checking the divisibility of remainders by 15, 10, 6, 5, 3, and 2 is shown in the table 1.

Table 1
Remainders represented using bases 2, 3 and 5

Integers	Indices		
	2 (i)	3 (j)	5 (k)
1	0	0	0
2	1	0	0
3	0	1	0
4	2	0	0
5	0	0	1
6	1	1	0
8	3	0	0
9	0	2	0
10	1	0	1
12	2	1	0
15	0	1	1
16	4	0	0
18	1	2	0
20	2	0	1
24	3	1	0
27	0	3	0

If the remainder is not divisible by any of the following, it is checked whether the remainder is greater than 15 or not. If yes it is divided by 15 & again the remainder is checked using the same procedure i.e. from step 4.

If the remainder >=10 &<15, it is divided by 10 & the divisibility of the remainder is checked and same procedure is repeated.

If the remainder >=6 &<10, it is divided by 6 & the divisibility of the remainder is checked and same procedure is repeated.

If the remainder is not divisible by any of the following, it is checked whether the remainder is greater than 15 or not. If yes it is divided by 15 & again the remainder is checked using the same procedure i.e. from step 4.

The same method is applied by following the specific order to represent the remainder in TBNS.

7. Now let us consider the quotient. The quotient may be greater than 30 or less than 30. If the quotient is greater than 30, repeat from step 2. Else repeat from step 4. The quotient is treated in the same manner as the remainder.
8. The process is terminated when either the remainder or the quotient is zero and thus we get the desired result in TBNS.Thus using this algorithm, we can represent any integer by the bases 2, 3, and 5 in an efficient manner.

The algorithm is illustrated with an example. Let us take a number as 89. Then we follow these steps –

1. The number (89) is divided by 30.
2. Since the number (89) is not divisible by 30, after division, the quotient is 2 & the remainder is 29.
3. Since the remainder is not divisible by 10, 6, 5, 3, 2; we divide 29 by 15. The quotient is 1 & the remainder is 14.
4. Again 14 is divisible by 2. The quotient is 7. Finally 7 is again divided by 3, resulting the value of the quotient as 2 & remainder as 1.
5. Thus we can represent 89 in the following manner -
   $89 = (2^1 3^0 5^0 * 2^1 3^1 5^1) + (2^0 3^1 5^1 + 2^1 3^0 5^0 * (2^1 3^0 5^0 * 2^0 3^1 5^0 + 2^0 3^0 5^0))$
   $= 2^2 3^1 5^1 + (2^0 3^1 5^1 + 2^1 3^0 5^0 * (2^1 3^1 5^0 + 2^0 3^0 5^0))$

Algorithm II

Algorithm II is nothing but a modified version of Algorithm I. To reduce the complexity of the algorithm, we need to modify the algorithm using an approximation. The approximation is used while representing the remainders. Instead of dividing by 15, 10, 6, 5, 3, and 2; we represent the remainders that cannot be represented using 2, 3, and 5 using Single Digit TBNS. Here the algorithm is discussed in details.

The steps from 1-5 are repeated as in Algorithm I. The modifications are:-

Instead of checking the divisibility of the remainders, we introduce & apply the concept of SDTBNS. The remainders that cannot be expressed in terms of 2, 3, and 5; i.e. 7, 11, 13, 17 etc. are expressed in the form of $2^i 3^j 5^k$.

For example, 7 can be expressed using SDTBNS in the following way –

$7 = 2^{-35} * 3^{-45} * 5^{47}$, where the mod error is 0.000228.

Similarly the other remainders can also be represented in SDTBNS as shown in the Table 2 below. The table gives an idea about the errors in representation of an integer in SDTBNS. The values of i, j, and k are assumed to be optimal both in accordance with accuracy and data bus width.

Again the steps 7 and 8 of Algorithm III are repeated for the termination procedure. Thus any integer is expressed using the TBNS chain and the SDTBNS technique, which is assumed to be effective in the context of Signal Processing and Elliptic Curve Cryptography.

Table 2

SDTBNS representation of the prime numbers within the range 30 and the corresponding errors

Remainder	Power of 2 (i)	Power of 3 (j)	Power of 5 (k)	Error (Mod)
7	-35	-45	47	0.000228
11	49	87	-79	0.0001
13	-80	3	34	0.000027
17	52	2	-22	0.000519
19	-58	-12	35	0.00007
23	-53	-56	63	0.000127
29	58	28	-42	0.00022

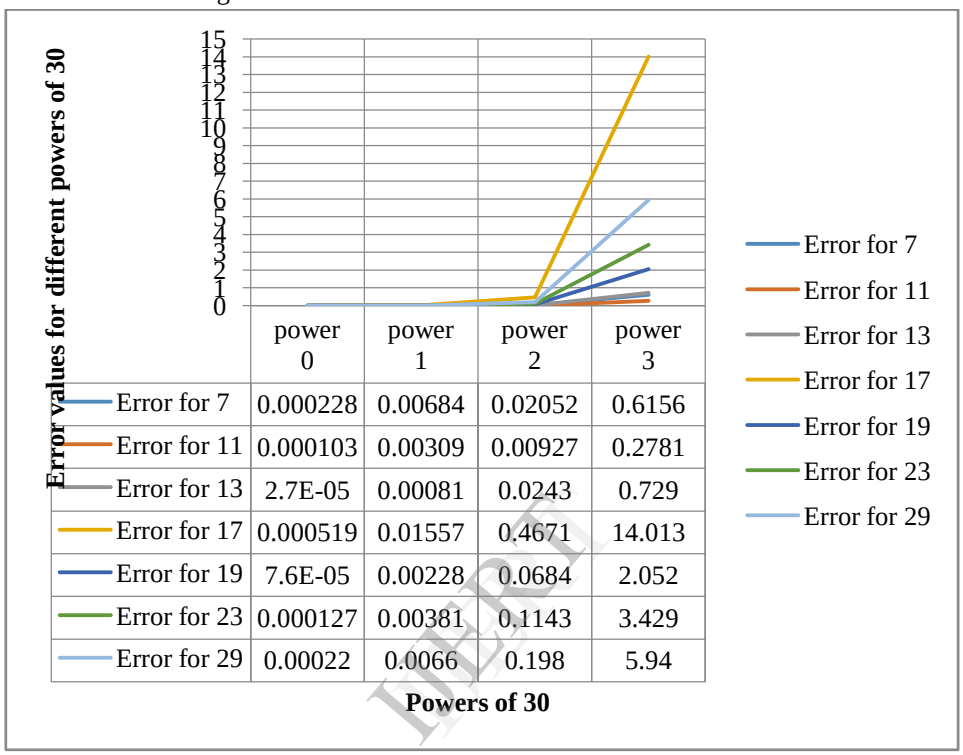
Let us compare between the third and the fourth algorithm. We will work out with the same example i.e. 89, which was represented using Algorithm I.

1. The number (89) is divided by 30
2. Since the number (89) is not divisible by 30, after division, the quotient is 2 & the remainder is 29.
3. 29 can be expressed by SDTBNS in the form of :- $29 = 2^{58} * 3^{28} * 5^{-42}$
   The error in this case is 0.00022.

4. Thus we can represent 89 in the following manner -
   $89 = (2^1 3^0 5^0 * 2^1 3^1 5^1) + 2^{58} 3^{28} 5^{-42}$
   $= 2^2 3^1 5^1 + 2^{58} 3^{28} 5^{-42}$

We can easily differentiate between Algorithm I & Algorithm II. In case of the first algorithm, we can represent any integer in TBNS without any error but the complexity increases as the integer value increases. It is much more generalized. On the contrary, in the case of the second algorithm we are representing an integer with a reduced complexity but by introducing some errors in the representation. The number of terms in the expression is also reduced.

To give an idea about the variation of error with the increase in the value of the integer & the power of 30, we have plot a graph to illustrate the change.



	power 0	power 1	power 2	power 3
Error for 7	0.000228	0.00684	0.02052	0.6156
Error for 11	0.000103	0.00309	0.00927	0.2781
Error for 13	2.7E-05	0.00081	0.0243	0.729
Error for 17	0.000519	0.01557	0.4671	14.013
Error for 19	7.6E-05	0.00228	0.0684	2.052
Error for 23	0.000127	0.00381	0.1143	3.429
Error for 29	0.00022	0.0066	0.198	5.94

Figure 1. Plot of powers of 30 against the calculated error

Let us now discuss some important points regarding the algorithms that are used to represent any integer in TBNS.

1. Maximum number that can be represented using this algorithm up to 3rd power of 30 is
   $(29*30^3 + 29*30^2 + 29*30 + 29 = 80999)$

   Percentage error = 0.00075

2. For any integer, that is to be represented using this algorithm, the maximum percentage error is 0.00305.

From table 2, we find that the maximum error is occurring while representing 17 in SDTBNS. And as we know that error always gets cumulated in this algorithm, so maximum error occurs when the number is 474827, which can be written as –

$(17*30^3 + 17*30^2 + 17*30 + 17)$

Error obtained in this case is 14.496, which is obtained from $[(0.000519*30^3)+(0.000519*30^2) + (0.000519 * 30) + 0.000519]$

To understand the above points, we illustrate them by taking the help of two examples –

i. $14854 = (16*30+15) * 30 + 4$
   $= 16*30^2 + 15*30 + 4$
   $= 2^4.(2.3.5)^2 + (3.5) * (2.3.5) + 2^2$
   $= 2^6.3^2.5^2 + 2.3^2.5^2 + 2^2.3^0.5^0$

   Percentage Error = 0

ii. $15719 = ((17*30+13) * 30) + 29$
   $= (2^{52}3^2 5^{-22})\ (2^1 3^1 5^1)^2 + (2^{-80}3^3 5^{34})\ (2^1 3^1 5^1)$
   $+ (2^{58}.3^{28}.5^{-42})$
   $= 2^{54}.3^4.5^{-20} + 2^{-79}.3^4.5^{35} + 2^{58}.3^{28}.5^{-42}$

   Percentage Error = 0.00297%

3. These algorithms can be used to represent any real numbers. It is discussed in details in the next portion.
4. In case of Algorithm III, as the value of the integer increases, the complexity also increases, which may lead to increase in redundancy.

For example,

a. $2357 = [((2^1 3^0 5^0 * 2^1 3^1 5^1) + 2^1 3^2 5^0) * 2^1 3^1 5^1] + (2^0 3^1 5^1 + 2^1 3^0 5^0) = (2^3 3^2 5^2 + 2^2 3^3 5^1) + (2^0 3^1 5^1 + 2^1 3^0 5^0)$

b. $1573 = 52*30 + 13 = (30+22)*30 + 13$

$= (2^1 3^1 5^1 + 2^0 3^1 5^1 + 2^1 3^1 5^0 + 2^0 3^0 5^0) * (2^1 3^1 5^1) + (2^1 3^0 5^1 + 2^0 3^1 5^0)$

$= 2^2 3^2 5^2 + 2^1 3^2 5^2 + 2^2 3^2 5^1 + 2^1 3^1 5^1 + 2^1 3^0 5^1 + 2^0 3^1 5^0$

Now the algorithms shown above in the paper has applications only while representing an integer. But the real deal is to represent any real number in DBNS or TBNS. In this paper we can use the above four algorithms to represent any real number in DBNS or TBNS. Again we can use the same approach of SDDBNS or SDTBNS to represent the same real number in DBNS or TBNS. Let us discuss the algorithms in detail and then analyse them.

We have discussed the algorithms that can be used to represent any integer to DBNS and then TBNS. Now let us explain the algorithms to represent any real number in DBNS or TBNS, which is the most interesting topic of this paper.

Algorithm III

The algorithm is explained in the following steps -

1. Let us consider any real numberas ‘n’.
2. We then separate the number into integer portion and decimal portion.
3. The integer part is expressed in TBNS using any algorithm that are described in details or any other algorithm like tree based approach in case of DBNS.
4. Now to represent the decimal part in TBNS, we first convert the decimal part in binary form. We need to take some approximation if the representation of the decimal part in binary reaches a recursive loop. In that case we take up to 8 digits.
5. Since we know the place values of each ‘0’ and ‘1’ we can use them to represent the decimal part in TBNS.
6. We add the place values of each ‘1’ and find the result. For example, the ‘1’ at the 8^{th} digit will have a place value as $2^{(8-1)}$.
7. The sum obtained is represented using any of the four algorithms that are shown above or any other algorithm to express it in TBNS.
8. 2^{-8} is then multiplied with the obtained TBNS form.

Thus we get our desired result and can easily represent any real number in TBNS form without introducing large amount of error.

Let us take an example to illustrate the above stated algorithm.

1. Let us take a real number as 6.675.
2. Now the integer part is 6 and the decimal part is 0.675.
3. 6 can be easily represented using any algorithm between I and II or by using any other algorithm that are already being proposed and used.
4. Now 0.675 is represented in binary form as $(10101110)_2$.
5. We represent the above binary form as
   $2^{-8} * (2^1+2^2+2^3+2^5+2^7) = 2^{-8} * 174$.
6. 174 can be expressed in TBNS. The obtained result is multiplied with 2^{-8}.

Error will vary according to the algorithm that is used to represent the number in TBNS. Instead of using the above algorithm to represent any real number in TBNS, we can approach with the concept of Single Digit TBNS. Let us now discuss them in details.

Algorithm IV

Input: Any real number ‘n’

Output: SDTBNS representation of the number

1. int i,j,k,s1=0,s2=0,s3=0;
2. float e,e1=0.005,n1;
3. for k=-100 to k<=100 do /* outer loop */
4. for j=-100 to j<=100 do /* inner loop 2 */
5. fori=-100 to i<=100 do /* inner loop 1 */
6. $n1=2^i * 3^j * 5^k$;
7. e = (n-n1)/n;
8. if e<0 then
   e=-(e);
9. if e<e1 then
   e1=e;s1=i;s2=j;s3=k;
10. end inner loop 1

11. end inner loop 2
12. end outer loop
13. print the powers of 2, 3 and 5 and the error.

The above algorithm can only be implemented using any programming language like C or Java.

Let us take the same example to compare between SDDBNS and SDTBNS –

1. Let the number be 5.67.
2. After computing the program we get the result as $5.67 = 2^{-37} * 3^{-41} * 5^{45}$
3. The error is 0.000033.

After discussing about all the algorithms in details, we present two tables that can help us a lot in distinguishing the different algorithms and a designer can choose the best algorithm required for his purpose. The tables are shown below. We have used the same integer (1077) & real number (1077.36523)for representation in either DBNS or TBNS so that we can compare by using some parameters.

Table 3

Comparison between the different algorithms with respect to various parameters

Algorithms	Number of Elements in the Chain	Maximum/ MinimumPower of 2	Maximum/ MinimumPower of 3	Maximum/ MinimumPower of 5	Highest (or Lowest) Power	Percentage Error
1	5	2	2	2	2	0
2	3	2	3	2	3	0
3	1	32	14	-19	32	0.00268
4	1	69	39	-52	69	0.001046
5	5	-8	3	2	-8	0.00018
6	7	-8	2	2	-8	0.00018

For both the tables drawn, algorithm 1 denotes the algorithm to represent any integer in TBNS as described in 'algorithm I', algorithm 2 resembles 'algorithm II', algorithms 3 approach SDTBNS to represent any integer, algorithms 4 approach SDTBNS to represent any real number, algorithms 5 and algorithm 6 use 'algorithm III' to represent the real number TBNS, where the integer is represented by using'algorithm II', 'algorithm I' respectively.

Table 4

Comparative study of the proposed Algorithms

Algorithms	Applicable to	Computational complexity	Hardware requirement		Error	Speed of computation
			Requirement of adders & multipliers	Capacity of the adders & multipliers		
1	Integer	Moderate	High	Low	Errorless	Moderate
2	Integer	Moderate	Low	High	Moderate	Fast
3	Integer	High	Least	High	Moderate	Slow
4	Any Real number	High	Least	High	Moderate	Slow
5	Any Real number	High	Low	High	Moderate	Slow
6	Any Real number	High	Moderate	Moderate	Low	Moderate

IV. APPLICATIONS

Digital signal processing (DSP) is the technology that is omnipresent in almost every engineering discipline. A typical processor devotes a considerable amount of processing time in performing arithmetic operations, particularly multiplication operations. Multiplication is one of the basic arithmetic operations and it requires substantially more hardware resources and processing time than addition and subtraction. In fact, 8.72% of all the instruction in typical processing units is multiplication. The core computing process is always a multiplication routine; therefore, DSP engineers are constantly looking for new algorithms and hardware to implement them. Finite impulse response (FIR) filters, discrete cosine transform (DCT), and discrete Fourier transform (DFT), for instance, are central operations in high-throughput systems and they use a huge amount of such operations. Their optimization widely impacts the performance of the global system that uses them.Application of this number systems in digital signal processing (DSP) has been explored. Multiplier based on TBNS is one of the fast and low power multiplier.

For multiplication algorithms performed in DSP applications latency and throughput are the two major concerns from delay perspective. In TBNS, the calculations of the exponential power of the different prime bases are necessary. Fewer gates will be required. Less energy, less hardware will be needed for implementation. In this paper, TBNS is applied to the binary number system and is used to develop digital multiplier architecture. TBNS is much more efficient in the multiplication of large numbers as it reduces the multiplication of two large numbers to that of two smaller ones. The algorithms can be implemented in the design of a low power high speed algorithms for arithmetic logic units using TBNS. Employing this technique in the computation algorithms of the coprocessor will reduce the complexity, execution time, area, power.

The TBNS technique has its maximum application in Fast Fourier Transform, where complex and large multiplications are required. In the case of the radix-2 DIT-FFT algorithm, the butterfly takes two inputs (X_0, X_1) and gives two outputs (Y_0, Y_1) by the formula -

$Y_0=X_0+X_1\omega^K$

$Y_1=X_0-X_1\omega^K$where ω= twiddle factor.

As this is basically a complex multiplication and addition so, by this proposed algorithm, FFT calculation can be done faster. Same is applicable in the case of DIF-FFT algorithm. And by decimation more point DFT can be calculated efficiently.

We have shown a block diagram to explain it by using Triple Base Number System (TBNS), employing the bases as 2, 3, and 5. Similar architecture can be obtained in case of DBNS.

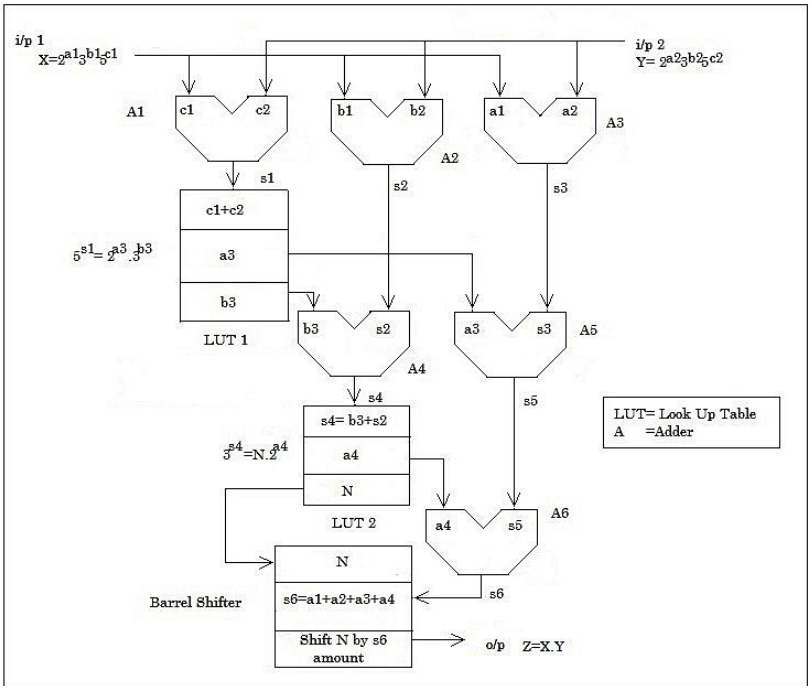


Figure 2. Block Diagram of a Multiplier Unit

One of the most important application of multi base number system lies in the field of cryptography.

Cryptography is study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Triple Base Number System (TBNS) has found application in cryptography especially in Elliptic Curve Cryptography (ECC).

Scalar multiplication is the bottleneck operationof the elliptic curve cryptography. It is to computethe result - Q = rSwhen S and Q are points on the elliptic curve and ris a positive integer. The computation time of theoperation strongly depends on the representationof 'r'.'r' can be represented using the various algorithms in DBNS and TBNS.

In ECC, signature verification is one of the most important step and this mainly requires a double-scalar multiplication of the form [n]P+ [m]Q. It can be done by calculating separately the product values and then by adding them. But if this is done by a combined operation then this is called multi-scalar multiplication. These algorithms can also be used for such operations.

V. CONCLUSIONS

In the past, a thorough examination of the algorithms with the respect to particular technology has only been partially done. The merit of the new technology is to be evaluated by its ability to efficiently implement the computational algorithms. Therefore, it is important to develop computational structures that fit well into the execution model of the processor and are optimized for the current technology. In such a case, optimization of the algorithms is performed globally across the critical path of its implementation.

In this work, we have presented & discussed in details about few new algorithms in DBNS and TBNS, as a technique for representing numbers that allows potentially low complexity arithmetic operations using a variety of implementation media. The design of the architecture that converts any integer into TBNS & that performs the multiplication of two integers is simple & efficient. These approaches greatly improves the practicality of both DBNS and TBNS. We have represented any real number in both DBNS and TBNS which will improve the performance of a multiplier.

As future works, we want to design the complete architecture of the multiplier based on DBNS and TBNS and implement it in a processor. Also we want to improve the proposed algorithms and find and try to reduce the computational complexities and errors.

VI. REFERENCES

[1] Christophe Doche, David R. Kohel, and Francesco Sica, "Double-Base Number System for Multi-scalar Multiplications", Draft, September, 9, 2008.

[2]Doche, C.,Habsieger, L. , "A Tree-BasedApproach for Computing Double-BaseChains", in: Y. Mu, W. Susilo and J. Seberry (Eds.), ACISP 2008, LNCS 5107, PP. 433-446, 2008, Springer-Verlag Berlin Heidelberg 2008.

[3] Avanzi, R. M., Cohen, H., Doche, C., Frey,G., Nguyen, K., Lange, T., Vercauteren, F.:Handbook of Elliptic and HyperellipticCurve Cryptography, in: Discrete Mathematics and its Application, Chapman and Hall/CRC, Boca Raton (2005).

[4]Avanzi, R. M., Dimitrov, V. S., Doche, C., Sica, and F.: Extending ScalarMultiplication using Double Bases, in: Lai, X., Chen, K. (Eds.), ASIACRYPT 2006, LNCS, vol. 4284, pp. 130 – 144, Springer, Heidelberg (2006).

[5]J. Adikari, V. Dimitrov, and L. Imbert. Hybrid Binary-Ternary Joint Sparse Form and its Application in Elliptic Curve Cryptography. Preprint, Available at: http://eprint.iacr.org/2008.

[6] V. S. Dimitrov, G. A. Jullien and W. C. Miller, "Theory and Application of Double Base Number System", IEEE Transaction on Computers, vol. 48, No. 10, pp-1098-1106, October, 1999.

[7] S. Maitra, A. Sinha, "Triple-Base Hybrid Joint Sparse Form and its Applications", International Journal of Computer Applications (0975 – 8887), vol. 43, No. 3, April, 2012.

[8] Pavel Sinha, Amitabha Sinha, KrishanuMukherjee and Kenneth Alan Newton, "Triple Base Number Digital and Numerical Processing System", Patent filed under E. S. P. Microdesign Inc. , Pennsylvania, U. S. A. , U. S. Pat. App. No. 11/488, 138.

[9] S. Maitra, A. Sinha, "A Single Digit Triple Base Number System – A New Concept for Implementing High Performance Multiplier Unit for DSP Applications", Proceedings of the sixth International Conference on Information, Communication and Signal Processing (ICICS2007), December, 10-13,2007.

[10] S. Maitra, A. Sinha, "Architecture of Mixed Radix Number System – A New Approach of Designing Digital Filter", proceedings of the 10th IASTED International Conference on Signal and

Image Processing(SIP2008), August, 18-20,2008, Kailua-Kona, HI, U.S.A