FPGA Implementation of MAC Unit for Double Base Ternary Number System (DBTNS) and its Performance Analysis

Aniruddha Ghosh
Calcutta Institute of Technology
Uluberia, Howrah, W.B, India

Amitabha Sinha
Birbhum Institute of Engineering & Technology
Suri, Birbhum, W.B, India

ABSTRACT
New methodologies for efficiently describing and implementing digital systems are investigated as the complexity of binary digital hardware system is relentlessly expanding. From the recent study, it is shown that multi valued logic approach is more advantageous over existing binary digital system. Ternary means a multilevel switching component, which switches between 3 levels. Recent study on ternary number system (TNS), has shown numerous advantages over binary. In recent times, Double Base Number Systems (DBNS) are considered as alternatives to binary number system because of their capabilities of performing partial product free multiplications. On the other hand, Double Base Ternary Number System (DBTNS) multipliers are efficient compared to conventional TNS multiplier. High performance digital signal processing systems which can able to handle all Digital Signal Processing (DSP) algorithms, broadly utilize Multiply-Accumulate (MAC) operation. So, TNS Adder and DBTNS Multipliers can be used to implement fast MAC units. Keeping this in view, a new approach of designing efficient MAC unit using DBTNS multiplier is proposed in this work. The performance of proposed MAC unit is compared with conventional ternary multiplier-based MAC unit and they are mapped on a FPGA chip. Performance analysis clearly indicates that the supremacy of the proposed architecture over conventional ternary multiplier-based MAC unit.

Keywords
Ternary Number System (TNS); Trit; Ternary Gates; Ternary Arithmetic; Double Base Ternary Number Systems (DBTNS); DBTNS Multiplier; Multiply and Accumulate Unit (MAC); FPGA; DSP Algorithms.

1. INTRODUCTION
In recent times, design of high performance Digital Signal Processors is gaining attention to many researchers because of emergence of high end applications. DSP algorithms [1], which are nothing but sum of products and these are normally performed in single cycle. The speed of DSP based applications can be improved by enhancing the speed of multiplier and addition [2] unit. In recent studies, it has been observed that non-weighted and non-binary number system can help to design high performance DSP applications [3]. Keeping these issues in view, multivalued [4] framework like ternary number system (TNS) [5][6] can be considered for designing high performance DSP applications. From the Shannon's information theory, it is clear that one trit (ternary digit) contains more information (about 1.58496) than one bit (binary digit). Ternary is a multivalued logic system [4], mainly base-3 numeral framework. Although ternary frequently refers to a framework in which the three levels 0, 1, and 2 are used to generate all numbers. Now a days, Double Base Number System (DBNS) [7] is very attractive for its ability to perform partial product free multiplication. So, instead of conventional TNS multiplier, Double Base Ternary Number System (DBTNS) multiplier can help to reduce the complexity of multiplication [8] [9]. But, major bottleneck is the extraction of indices ([i, j] pair) [10] while converting ternary number to double base number. For implementing DBTNS conversion, LUT based approach has been adopted [11]. Moreover, when dynamic range increases, LUT based approaches become incapable of breaking the complexity as the LUT size increases exponentially. The operations mainly required in all the Digital Signal Processing applications are multiplication and additions, in other word it can be termed as multiplication and accumulation. Therefore, there is a demand for high speed processors having committed hardware to upgrade the speed with which these multiplications and accumulations are performed. Initially, accumulator is updated with zero. Input data sequences are first multiplied in the multiplier unit. The first product is added with zero, which is earlier stored in the accumulator and then accumulator value is updated. Partial product free multiplication can be performed by DBTNS multiplier so high speed multiply accumulate (MAC) units can be implemented using DBTNS multiplier. To understand the area-efficient and high-speed MAC unit proposed in this work, at first hardware complexities of DBTNS MAC architecture is examined and contrast with the conventional TNS MAC unit is depicted by investigating the execution on a FIR algorithm. The architecture for the proposed MAC unit is executed and validated on Xilinx Virtex FPGA using ISE Simulator.

2. REVIEW OF TNS and DBTNS
2.1 Ternary Number Systems (TNS)
Ternary is a multivalued [4] rationale framework, for the most part base-3 numeral system. Albeit ternary regularly alludes to a system in which the three digits 0, 1, and 2 are used to present all numbers in this domain. Ternary Number Systems have the capabilities of performing basic arithmetic operations like compliment operation, addition, subtraction, multiplication and division [12][13].

2.2 Double Base Ternary Number Systems (DBTNS)
An integer can be represented as a sum of mixed powers of two integers. This technique of representation is called Double Base Number System i.e. DBNS. A thorough discussion on DBNS is presented in the ref. [7] and [11]. In the DBNS, an integer, x, can be represented in the following form:

$$x=\sum_{i,j} d_{i,j} 2^i 3^j, \quad \text{where } d_{i,j}=\{0, 1\}.$$

From the expression, a given binary number can be converted into DBNS as number of (i, j) pair. These are also referred as DBNS indices [14][15][16]. Basis on the discussion of ref. [7]

and [11], if x is a ternary number then x can be expressed as follows:

$$x=\sum_{i,j} d_{i,j} 2^i 3^j ,$$

where $d_{i,j}=\{0, 1, 2\}$. These indices (i, j) are in ternary number system. So, conversion of a ternary number into DBNS as number of (i, j) pair in TNS domain [12] can be termed as Double Base Ternary Number Systems (DBTNS). In this paper, table 1 represents DBTNS table where trit length of indices (i, j) is 1 whose dynamic range is 91 and in table 2, trit length of indices (i, j) is 2 whose dynamic range is 5028751.

3. ARCHITECTURE OF MAC UNIT USING TNS

Main objective of a conventional MAC unit is multiplication and accumulation. The term accumulation signifies its ability of performing addition operation and accumulate the sum of the previous consecutive products [17]. So, multiplier, adder and accumulator are required for implementing MAC unit. Single cycle multiplication and accumulation can be done using this MAC unit. Accumulator unit is implemented based on TNS Register [18]. Here, in the proposed architecture all the modules are implemented using Ternary Number System (TNS). The proposed architecture of MAC unit is shown in figure 1. The N-trit input are fetched from memory location and fed to the multiplier unit for multiplication operation. Then the product is added with content of accumulator. The output of multiplier unit is 2N trit, so the output of the adder is of 2N+1. In this architecture, value of N is considered as 2 trit, 3 trit and 4 trit.

3.1 Multi-trit TNS Adder

The multi-trit TNS adder [6][12] is implemented based on conventional Ripple Carry Adder [2][19]. In this adder, carry is propagated to next one trit adder as shown in figure 2.

3.2 Multi-trit TNS Multiplier

The logic multiplication of two multi-trits ternary numbers can be accomplished in the same way of doing things as in longhand multiplication [12][20].The multiplicand is multiplied by the individual trits of the multiplier to generate the partial products. Initially, multiplicand is multiplied by the first trit of the multiplier for generating first partial product; second partial product is generated when the multiplicand is multiplied by the second trit of the multiplier et cetera. The (i+1)th partial product is one trit moved to one side w.r.t the i-th partial products. The product is the summary of these partial products [20]. The TNS Multiplier for 1 trit is shown in figure 3. This TNS Multiplier is implemented using LUT based approach. The input and output relationship which is depicted in the table 3, is kept in the LUT. After multiplying two 1 trit data, output is of 2 trit. Most significant trit is called carry and least significant trit is product. The TNS Multiplier for 2 trit is shown in figure 4. Initially, partial product is generated using Partial Product Generation (PP Gen) Unit. Two 1-trit TNS Multiplier and two 1-trit TNS Half Adder are required to implement PP Gen Unit. Then the partial products are pass through the adder to produce multiplied data. The TNS Multiplier for 3 trit is shown in figure 5. Here also, PP Gen Unit is one of the important unit for generating partial product. Each PP Gen Unit has length of 4 trit output. These partial products are added. The TNS Multiplier for 4 trit is shown in figure 6. The partial product which is the output of PP Gen unit, is of 5-trit. These partial products are added to produce product of multiplier.

4. ARCHITECTURE OF PROPOSED MAC UNIT USING DBTNS

Multiplication and accumulation operation can be performed in single cycle by MAC unit. In the architecture of the proposed MAC unit [2][12][17], there are two input h(n) and x(n) which is in TNS. Initially, they are converted into DBTNS. The proposed architecture is depicted in the figure 7. So, the following modules are required for implementation of proposed MAC unit.

A. Integer to TNS Conversion
B. DBTNS Conversion
C. DBTNS Multiplier
D. TNS Adder
E. TNS Accumulator

4.1 DBTNS Conversion Unit

The conversion of Ternary Number to Double Base Ternary Number System is carried out by this unit. The approach is totally Look Up Table (LUT) based [11]. In DBTNS, there are two bases, one is 2 and another is 3 and the number is represented in terms of power 2 and 3 i.e $X=2^i.3^j$ where these indices (i, j) are in ternary number system[4]. Here, the values of i and j are stored in different location of LUT as shown in figure 8. These i and j can be used in the consecutive steps.

4.2 DBTNS Multiplier Unit

Suppose, X_1 and X_2 are two ternary numbers. In DBTNS, $X_1=2^{i_1}.3^{j_1}$ and $X_2=2^{i_2}.3^{j_2}$. Now, Z = X_1. X_2 then $Z=2^{(i_1+i_2)}.3^{(j_1+j_2)}$. The operation of DBTNS Multiplier [10] is represented by the flowchart which is depicted in the figure 9. The architecture of DBTNS Multiplier is depicted in the figure 10. The steps for performing multiplication operation using DBTNS Multiplier are stated below:

Step-1: The input sequence, x(n) and h(n) are initialized for N-tap FIR Filter. Go to next step.

Step-2: x(n) and h(n) are converted in ternary number system i.e. x(n) → X and h(n) → H, $\forall X, H \in TNS$, Go to next step.

Step-3: X and H are converted in DBTNS using LUT based approach i.e. X → $2^{i}_2 . 3^{j}_2$ and H → $2^{i}_1 . 3^{j}_1$, $\forall$ $^{i}_1$, $^{i}_2$, $^{j}_1$, $^{j}_2$ $\in TNS$. Go to next step.

Step-4: Both the power of 2 is added to generate 'i' and both the power of 3 is also added to generate 'j'. i ← $i_1 + i_2$ and j ← $j_1 + j_2$, $\forall$ ^{i}, $^{i}_1$, $^{i}_2$, ^{j}, $^{j}_1$, $^{j}_2$ $\in TNS$. Where '+' is the ternary based addition. Go to next step.

Step-5: Contents of LUT memory location 'i' is transferred to 'temp' i.e temp ← LUT[i]. Go to next step.

Step-6: The contents of 'temp' are passed through barrel shifter. Barrel shifter can perform single cycle multi-bit shifting. The contents of 'temp' are shifted by 'j' amount to produce final result as product i.e. Product ← BS[temp, j] Finally output is generated as product.

In the architecture of DBTNS multiplier, 'n' is the trit length of indices, 'N' is the trit length of ternary equivalent of power of 2 i.e. $2^{(i_1 + i_2)}$ and 'M' is the trit length of product. For implementing this architecture [21], TNS Adder, Barrel Shifter

[17] and LUT are required. TNS Adder is used to add the indices. The ternary equivalent of power of 2 i.e. $2^{(i_1 + i_2)}$ is kept in the LUT. This stored data is passed through a barrel shifter as it has ability to perform multi-trit shifting in a single cycle. The amount of shift is defined by the power of 3 i.e. $(j_1 + j_2)$. The multiplied result can be collected from the barrel shifter. The DBTNS Multiplier with indices trit length 1 trit i.e n = 1, N = 3 trit then length of multiplied data, M is 7. Here, 1 trit TNS Adder can produce maximum value '11' (in ternary). So, the range of LUT is '000' to '121'. Trit length of output of DBTNS multiplier is described in the table 4 for different trit length of indices.

5. PRINCIPLE OF OPERATION OF TNS MAC UNIT FOR FIR FILTER

A MAC unit can perform single cycle multiplication & Accumulation. As per the block diagram x(n) and h(n) are the input and y(n) is output. Here LUT is used for storing input data sequence and filter coefficient and the number of location of LUT is used for this purpose, depends on the number of tapping of FIR filter [1][17]. FIR filter can be represented by the following equation: $y(n) = \sum_{k=0}^{N-1} x(n-k).h(k)$

There are two inputs namely x(n) (input data sequence) and h(n) (filter coefficient). The operation of TNS based FIR is represented in the flowchart of fig. 14. The steps involved for implementing TNS based FIR Filter is written below:

Step-1: The input sequence, x(n) and the filter coefficient, h(k) are initialized for N-tap FIR Filter, $\forall\ n,k \in N.$

Step-2: Initially n, k, acc (i.e. accumulator) are updated with 0.

Step-3: 'acc' is updated i.e. acc ← acc+x(n-k)*h(k) where '+' and '*' are the ternary based addition and multiplication respectively.

Step-4: If k≠N-1, then k is incremented by 1 i.e. k=k+1 then go to step 3, otherwise go to next.

Step-5: If n≠N-1, then n is incremented by 1 i.e. n=n+1 then go to step 3, otherwise go to next.

Step-6: Accumulator data is sent to the output and final result is generated.

Depending on the above flowchart, architecture of TNS based MAC unit is implemented which is depicted in the fig. 15. The clocked based analysis of the 4 tap FIR filter operation using proposed MAC unit is as follows:

1st Clock - It means that in first clock two inputs x(0) and h(3) will multiplied, and multiplied result will added with zero that initially kept in accumulator.

2nd Clock - In Second clock again two inputs x(1) and h(2) will multiplied and obtain result will be added with the result that is stored in accumulator i.e. x(0).h(3) + x(1).h(2).

3rd Clock - Again in third clock two inputs x (2) and h (2) will multiplied and added with the result that stored in accumulator i.e. x(0).h(3) + x(1).h(2)+x(2).h(1).

4th Clock - In fourth clock, we obtain y(n)= x(0).h(3) + x(1).h(2)+x(2).h(1)+x(3).h(0).

In the last clock i.e. in 4th clock it will generate the output. The output will depend upon the number of tapping of FIR filter. If the no. of tapping is four, the MAC unit will generate output in the fourth clock. If the tapping is eight the output will be generated in eighth clock. From the above analysis, it can be concluded that the output of the MAC unit is same as the FIR filter output. Hence FIR filter can be implemented using MAC unit.

6. PRINCIPLE OF OPERATION OF DBTNS MAC UNIT FOR FIR FILTER

Multiply-accumulate operation can be performed in single cycle by MAC unit [22]. There are two inputs, h(n) & x(n) in a MAC unit. The inputs are multiplied first and added with zero which is initially stored in an accumulator. In the very next clock, the next two inputs are multiplied and added with previous data and update the accumulator. The architecture of the proposed MAC unit is shown in figure 13. In the proposed architecture, there are 5 (five) LUTs, among these 5 LUTs, LUT-1 & LUT-3 are used for integer to ternary number, LUT-2 & LUT-4 are used for converting TNS to double base ternary number and LUT-5 is used to convert ternary to integer. Initially, **x(n)** and **h(n)** are converted into DBTNS i.e. $x(n) = 2^{i_1}.3^{j_1}$ and $h(n) = 2^{i_2}.3^{j_2}$. The indices of 2 & 3 are passed through DBTNS Multiplier unit and multiplied data is added with zero which is initially stored in an accumulator. The architecture of DBTNS multiplier is shown in figure 10. For indices 'i' and 'j', trit length are 1, 2, 3 then output trit length are 7, 27, 85 respectively (Table 4). The operation of DBTNS based FIR is represented in the flowchart of figure 14. The steps involved for implementing DBTNS based FIR Filter are written below:

Step-1: The input sequence, x(n) and the filter coefficient, h(n) are initialized for N-tap FIR Filter.

⇨ n ← 0, $\forall\ n \in N$ and acc ← 0.

Go to next step.

Step-2: x(n) and h(n) are converted in ternary number system.

⇨ x(n) → X and h(n) → H, $\forall X, H \in TNS$.

Go to next step.

Step-3: X and H are converted to DBTNS in LUT based approach.

⇨ X → $2^{i}_{2} . 3^{j}_{2}$ and H → $2^{i}_{1} . 3^{j}_{1}$, $\forall\ i_1, i_2, j_1, j_2 \in TNS$. Go to next step.

Step-4: X and H are multiplied using DBTNS multiplier and product P_n is generated.

⇨ $P_n = 2^{(i_1 + i_2)} . 3^{(j_1 + j_2)}$, $\forall\ i_1, i_2, j_1, j_2 \in TNS$.

Go to next step.

Step-5: 'acc' is updated i.e. acc ← acc + P_n where '+' is the ternary based addition.

Go to next step.

Step-6: If n≠N-1, then n is incremented by 1 i.e. n=n+1 then go to step 2, otherwise go to next step.

Step-7: Y ← acc where 'Y' is in DBTNS.

Step-8: 'Y' is converted to real number, y(n) and finally output is generated.

7. PERFORMANCE ANALYSIS OF PROPOSED TNS MAC UNIT

To implement FIR Filter [1][21] using TNS MAC unit, the delay and hardware complexity [2][6] of different adder and multiplier circuits have been compared. Total delay of this FIR Filter = (n-trit LUT access delay + time taken by TNS Multiplier + time taken by TNS Adder). Total delay of n-trit TNS Adder = (T_{HA} + (n-1).T_c), where, T_{HA} is time taken by half adder and T_c is the carry propagation delay in later stages. If the no. of trit of input data of FIR Filter is changed then execution time is also varied. Synthesis report of 8 tap FIR filter with change of Trit is shown in table 5. The relation between number of LUTs and number of trit maximum frequency and trit and execution time and trit is shown in the figure 15.

8. PERFORMANCE ANALYSIS OF PROPOSED DBTNS MAC UNIT

DBTNS conversion is performed by LUT based approach [10][17]. So over all time complexity [2][18] depends on the LUT size. To implement FIR filter using DBTNS MAC unit, total delay can be represented as (n - trit LUT access time for integer to TNS conversion + n - trit LUT access time for TNS to DBTNS conversion + time taken by DBTNS multiplier + time taken by TNS Adder + n - trit LUT access time for TNS to integer conversion). If the number of trit of indices of input data of FIR filter [2][21][23] are changed then execution time is also varied. The synthesis report of 8 tap FIR filter with change of trit is shown in the table 6. The relation between number of LUTs and number of trit, maximum frequency and trit and execution time and trit is shown in the figure 16.

9. CONCLUSION

In this paper, a new architecture for MAC unit has been proposed for implementing DSP algorithm like FIR algorithm [1][2] using two different number system like TNS and DBTNS. Partial product free multiplication operations can be performed by DBTNS multiplier efficiently. Since, DBTNS multipliers are efficient compared [17][19] to conventional TNS multiplier, so the novelty of the proposed MAC unit is depicted by the experimental results. The architecture was validated on Xilinx FPGA [15] and the detailed analysis and studies of different modules of the proposed units have been simulated using Xilinx ISE version 12.3. TNS is a multivalued logic approach which offers several advantages over existing binary digital system [4][5]. So, a detailed study can be made in this DBTNS domain on performance improvement for other DSP algorithms [1][24] like speech processing , high quality sound systems, adaptive echo cancellation , solar signal processing , military applications etc where in addition to high speed, high precisions are also required. Beside that exploring the possibilities of VLSI implementation of the multi-valued logic system using double base number system can also be a topic for future work.

10. REFERENCES

[1] Mitra, S.K.: 'Digital Signal Processing' (A Wiley-Inter science Publication,1999).

[2] Hwang, K. (Purdue University), Briggs, F.A. (Rice University): 'COMPUTER ARCHITECURE AND PARALLEL PROCESSING'(International Edition, 1985).

[3] Roy, R. , Datta, D. et al.: 'Comparative Study and Analysis of Performances among RNS, DBNS, TBNS and MNS for DSP Applications', Journal of Signal and Information Processing, 2015, (6), pp. 49-65. doi: 10.4236/ jsip.2015.62005.

[4] Gonzalez F, Mazumder P.: 'Multiple-valued signed digit adder using negative differential resistance devices', IEEE Trans. on Computers, 1998, (47), pp. 947 – 959.

[5] Chung-Yu-Wu.: 'Design& application of pipelined dynamic CMOS ternary logic & simple ternary differential logic', IEEE journal on solid state circuits, 1993, (28), pp. 895-906.

[6] Radanovic M. S.: 'Current-mode CMOS adders using multiple-valued logic': Canadian Conference on Electrical and Computer Engineering, 1996, pp. 190-193.

[7] Dimitrov, V. S., Jullien, G. A., Miller, W. C.: 'Theory and Applications of the Double-Base Number System', IEEE Trans. Computers, 1999, 48, (10), pp.1098-1106.

[8] Chen, J., Chang, C. H., et al.: 'Novel Design Algorithm for Low Complexity Programmable FIR Filters Based on Extended Double Base Number System', IEEE Transactions on Circuits and Systems I: Regular Papers, 2015, 62, (1), pp. 224-233. doi: 10.1109/TCSI.2014.2348072.

[9] Méloni, N., Hasan, M. A.: 'Efficient Double Bases for Scalar Multiplication', IEEE Transactions on Computers, 2015, 64, (8), pp. 2204-2212. doi: 10.1109/TC.2014.2360539.

[10] Singha, S., Ghosh, A., Sinha, A.: 'A New Architecture for FPGA based Implementation of Conversion of Binary to Double Base Number System (DBNS) Using Parallel Search Technique', ACM SIGARCH Computer Architecture News, 2011, 39, (5), pp. 12-18. DOI:10.1145/2093339.2093343.

[11] Dimitrov, V. S., Sadeghi-Emamchaie, S., et al.: 'A Near Canonic Double-Based Number System (DBNS) with Applications in Digital Signal Processing', Proceedings SPIE Conference on Advanced Signal Processing, 1996, pp. 14-25. http://dx.doi.org/10.1117/12.255433

[12] Yoeli, M., Rosenfeld, G.: 'Logical Design of ternary switching circuits', IEEE Trans Computer., 1965, 14, pp. 19-29.

[13Dhande, A.P., Ingole, V.T.: 'Design and Implementation Of 2 Bit Ternary ALU Slice', SETIT, 3rd International Conference: Sciences Of Electronic, Technologies Of Information And Telecommunications, 2005, pp. 7-21.

[14] Muscedere, R., Dimitrov, V. S., et al.: 'On Efficient Techniques for Difficult Operations in One and Two-digit DBNS Index Calculus', Proceedings 34th Asilomar Conference on Signals, Systems and Computers, November, 2000.

[15] Tessier, R., Burleson, W.: 'Reconfigurable computing for digital signal processing: A survey', Journal of VLSI Signal Processing, 2001, 28, pp 7-27.

[16] Singha, S., Sinha, A.: 'Survey of Various Number Systems and Their Applications', International Journal of Computer Science and Communication, 2010, 1, (1), pp. 73-76.

[17] Ghosh, A., Singha, S., Sinha, A.: 'A New Architecture for FPGA Implementation of A MAC Unit for Digital Signal Processors using Mixed Number System', ACM SIGARCH Computer Architecture News, 2012, 40, (2), pp. 33-38. DOI:10.1145/2234336.2234342.

[18] Dhande, A. P., Ingole, V.T.: 'Design of clocked ternary S-R and D flip-flop based on simple ternary gates',

International journal on software engineering and knowledge engineering, 2005, 15, (2), pp. 411-417.

[19] Hayes, J. P. : 'Computer Organization', (McGraw-Hill, 1998, 3rd edition).

[20] HASSAN, F. J., ABDUL-KARIM, M. A. H.: 'N × M trits ternary multiplier', International Journal of Electronics, 1983, 54, (5), pp. 643-650. DOI: 10.1080/00207218308938763.

[21] Sinha, A., Sinha, P., et al.: 'Multi based number systems for performance enhancement of Digital Signal Processors', Filed for U.S. patent. (U.S. Pat. Appl. No. 11/488,138), published in U.S. Patent documents serial no.488138 ,U.S. Class at publication 708/620, Int'l class : G06F 7/52 20060101 G06F007/52, 2006.

[22] Jullien, G. A., Dimitrov, V. S., et al.: 'A Hybrid DBNS Processor for DSP Computation' Proceedings International Symposium on Circuits and Systems, 1999, 1, pp. 5-8. doi: 10.1109/ISCAS.1999.777792.

[23] Eskritt,J., Muscedere, R., et al.: 'A 2-Digit DBNS Filter Architecture', Proceedings SiPS Workshop (Lafayette, L A), October, 2000, pp. 447-456. doi: 10.1109/SIPS.2000.886743

[24] B. G. Lee, A new Algorithm to compute the discrete cosine transforms. IEEE Trans on Acoustics, speech and signal Processing, 1984, 32, pp.1243- 1245.

Table 1. DBTNS Table for i, j → 1 trit

i \ j	0	1	2
0	0001	0010	0100
1	0002	0020	0200
2	0011	0110	1100

Table 2. DBTNS Table for i, j → 2 trit

i, j	00	01	02	10	11	12	20	21	22
00	000000000 00001	000000000 00010	000000000 00100	000000000 01000	000000000 10000	000000001 00000	000000010 00000	000000100 00000	000001000 00000
01	000000000 00002	000000000 00020	000000000 00200	000000000 02000	000000000 20000	000000002 00000	000000020 00000	000000200 00000	000002000 00000
02	000000000 00011	000000000 00110	000000000 01100	000000000 11000	000000001 10000	000000011 00000	000000110 00000	000001100 00000	000011000 00000
10	000000000 00022	000000000 00220	000000000 02200	000000000 22000	000000002 20000	000000022 00000	000000220 00000	000002200 00000	000022000 00000
11	000000000 00121	000000000 01210	000000000 12100	000000001 21000	000000012 10000	000000121 00000	000001210 00000	000012100 00000	000121000 00000
12	000000000 01012	000000000 10120	000000001 01200	000000010 12000	000000101 20000	000001012 00000	000010120 00000	000101200 00000	001012000 00000
20	000000000 02101	000000000 21010	000000002 10100	000000021 01000	000000210 10000	000002101 00000	000021010 00000	000210100 00000	002101000 00000
21	000000000 11202	000000001 12020	000000011 20200	000000112 02000	000001120 20000	000011202 00000	000112020 00000	001120200 00000	011202000 00000
22	000000001 00111	000000010 01110	000000100 11100	000001001 11000	000010011 10000	000100111 00000	001001110 00000	010011100 00000	100111000 00000

Table 3. Ternary Multiplication

MULTIPLICAND	MULTIPLIER	CARRY	PRODUCT
0	0	0	0
0	1	0	0
0	2	0	0
1	1	0	1
1	2	0	2
2	2	1	1

Table 4. Data Table of DBTNS Multiplier

INPUT INDEX TRIT LENGTH (i, j) **(n)**	TNS ADDER		LUT DATA RANGE	BARREL SHIFTER INPUT DATA TRIT LENGTH **(N)**	MAXIMUM SHIFT	PRODUCT **(M)**
	OUTPUT TRIT LENGTH **(n+1)**	OUTPUT DATA RANGE				
1	2	00 to 11	2^0 to 2^4	3	4	7
2	3	000 to 121	2^0 to 2^{16}	11	16	27
3	4	0000 to 1221	2^0 to 2^{52}	33	52	85

Table 5. Synthesis report of 8 tap FIR filter using TNS MAC unit with change of Trit

Sl. No.	No. of Trit	Synthesis Report			
		Number of Slice LUTs	Minimum period	Minimum input arrival time before clock	Maximum output required time after clock
1	4	148 out of 46560	3.571ns (Maximum Frequency: 279.994 MHz)	6.464ns	0.576ns
2	3	84 out of 46560	2.853ns (Maximum Frequency: 350.471 MHz)	4.974ns	0.576ns
3	2	35 out of 46560	2.135ns (Maximum Frequency: 468.362 MHz)	3.066ns	0.576ns

Table 6. Synthesis report of 8 tap FIR filter using DBTNS MAC unit with change of Trit

Sl. No.	INPUT INDEX TRIT LENGTH (i, j)	Synthesis Report			
		Number of Slice LUTs	Minimum period	Minimum input arrival time before clock	Maximum output required time after clock
1	1	90 out of 343680	6.405ns (Maximum Frequency: 156.128 MHz)	8.098ns	0.676ns
2	2	308 out of 343680	11.996ns (Maximum Frequency: 83.364 MHz)	14.235ns	0.676ns
3	3	619 out of 343680	21.887ns (Maximum Frequency: 45.690 MHz)	25.736ns	0.676ns

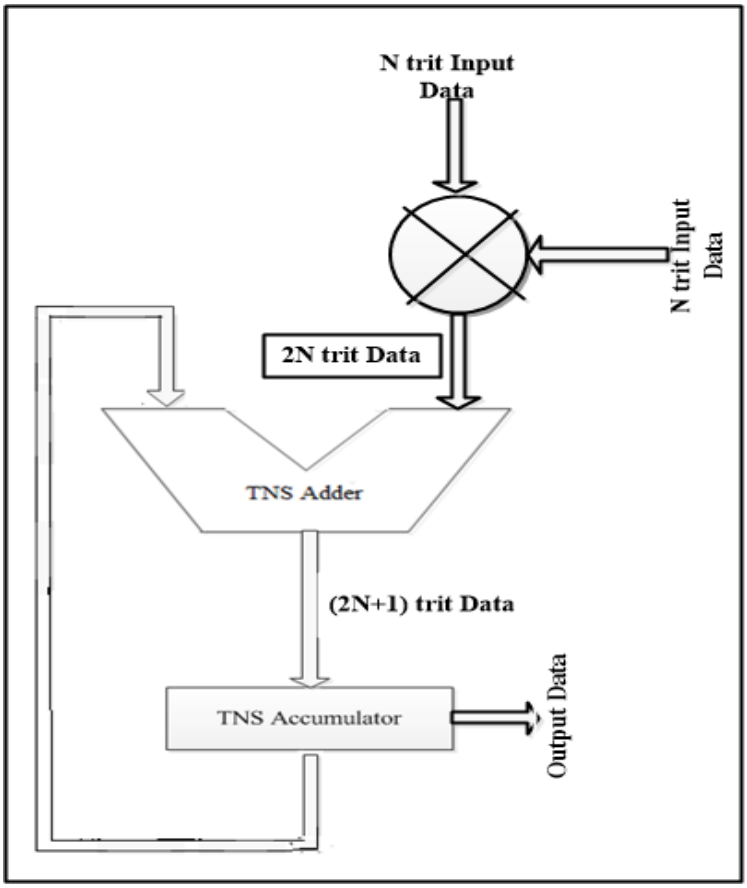

Figure 1. TNS MAC Unit

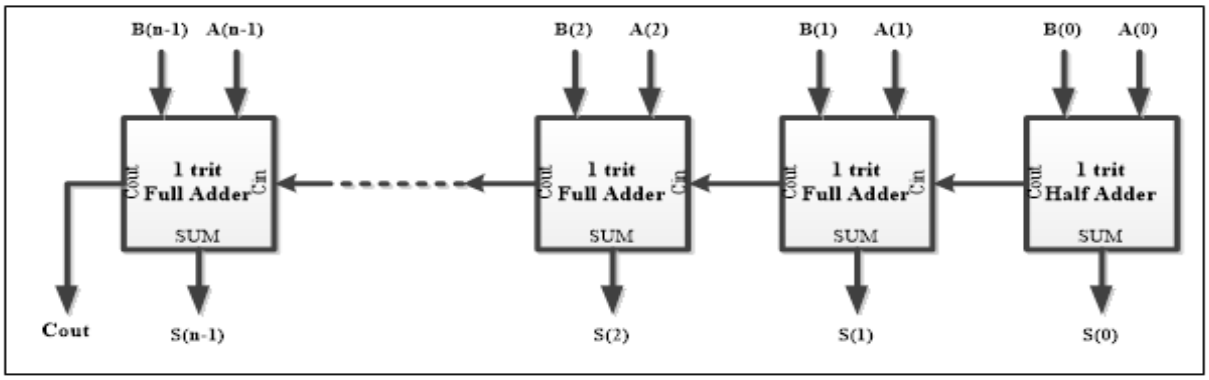

Figure 2. TNS Adder

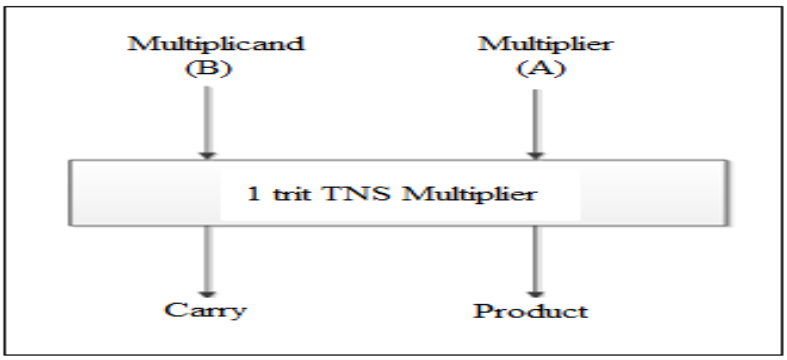

Figure 3. 1 Trit TNS Multiplier

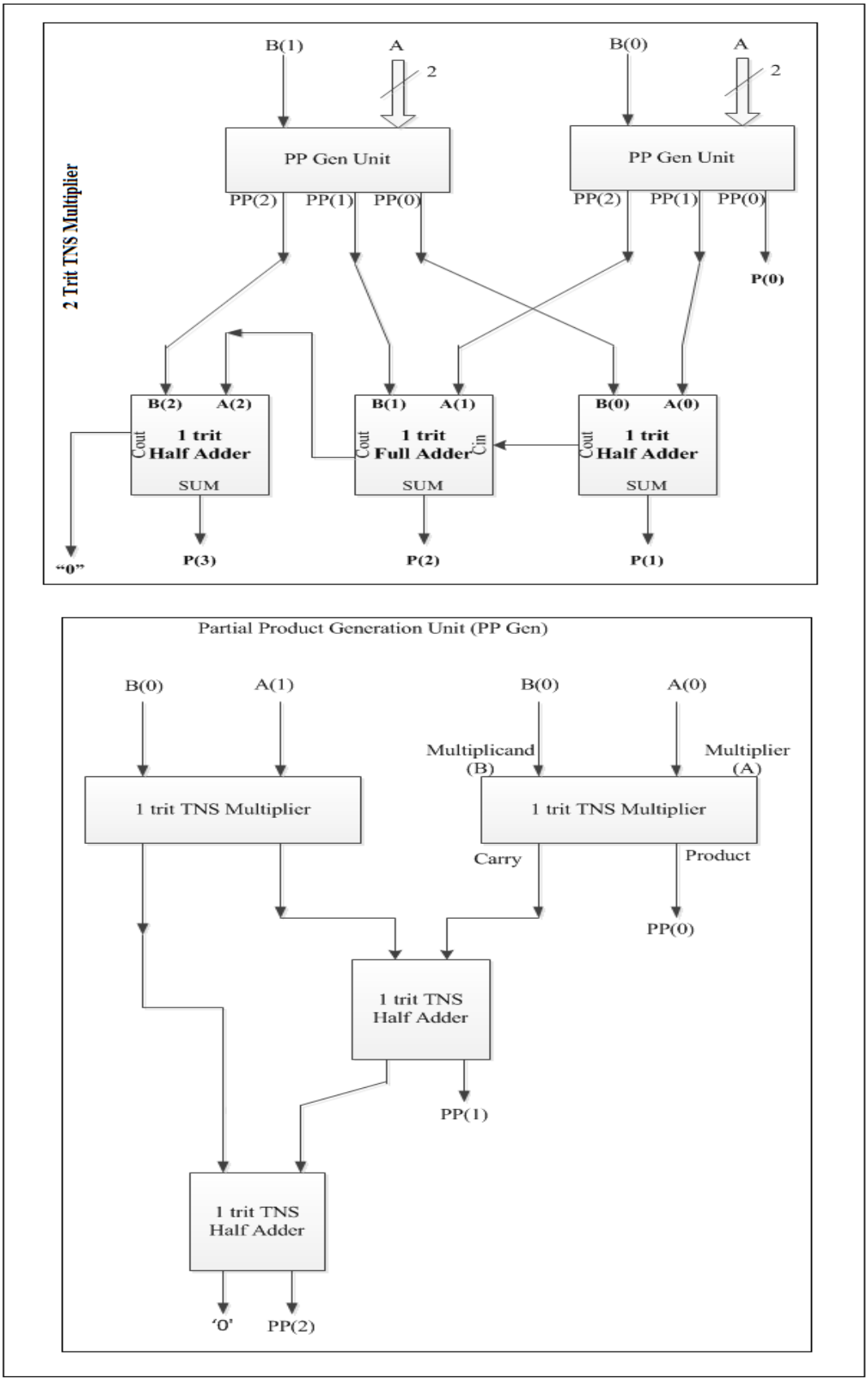

Figure 4. 2 Trit TNS Multiplier

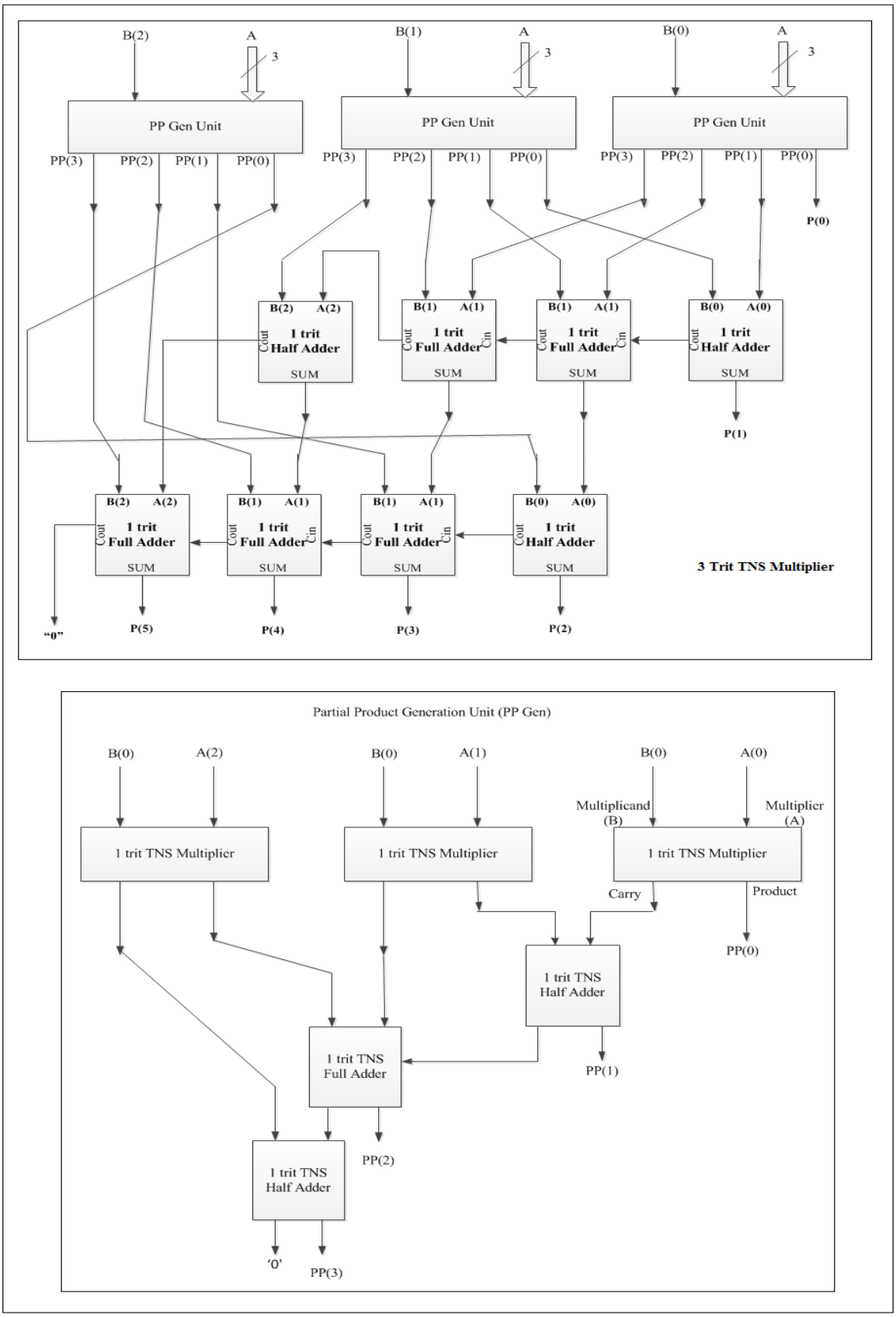

Figure 5. 3 Trit TNS Multiplie

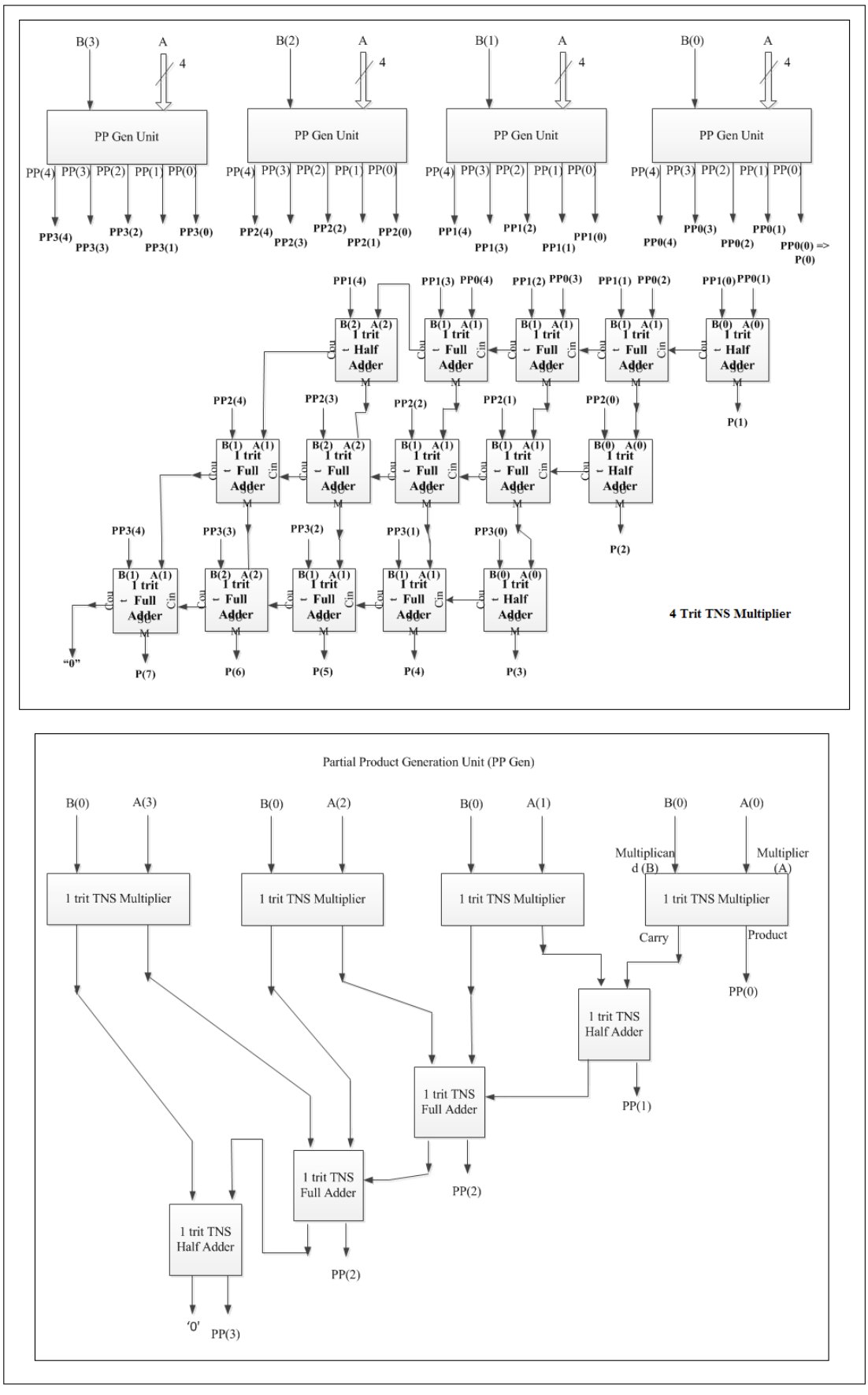

Figure 6. 4 Trit TNS Multiplier

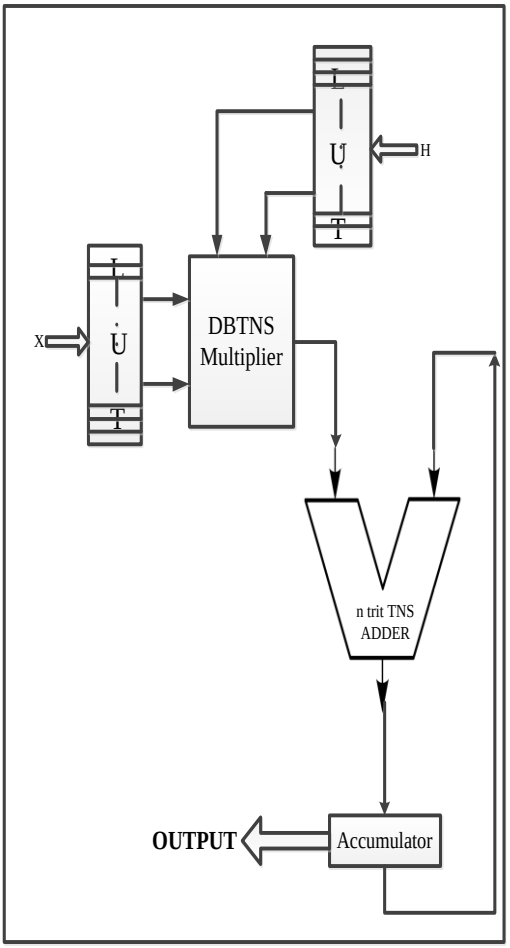

Figure 7. DBTNS Based MAC Unit

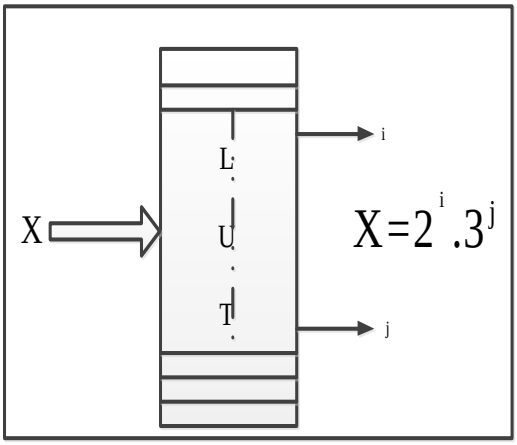

Figure 8. LUT Based DBTNS Conversion Unit

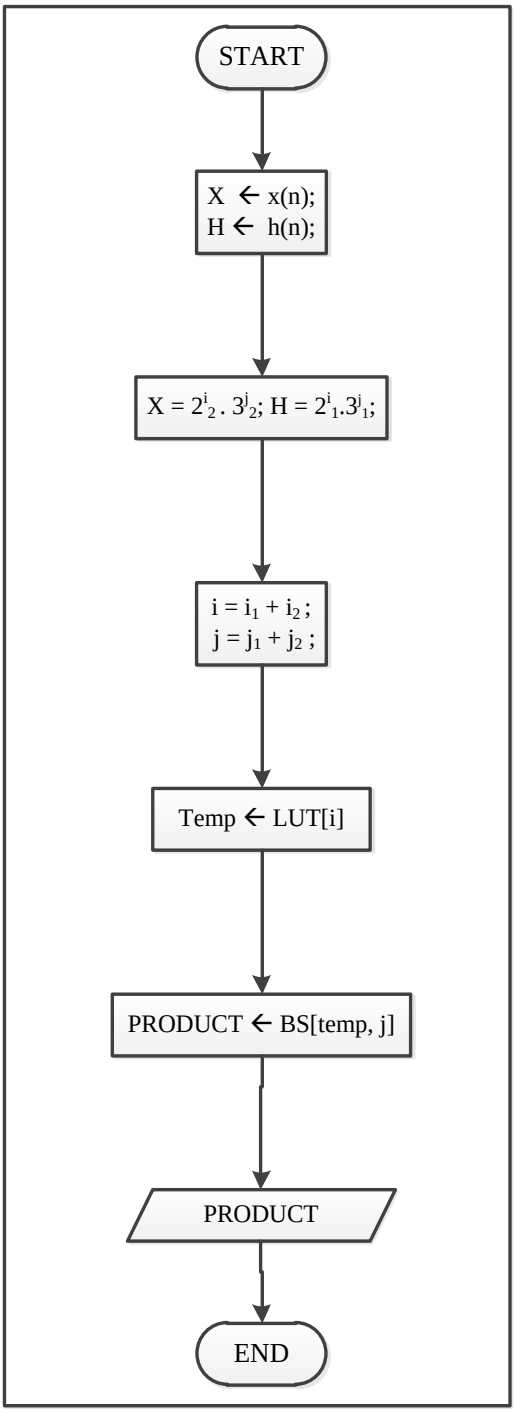

Figure 9. Computational flowchart of DBTNS Multiplier Unit

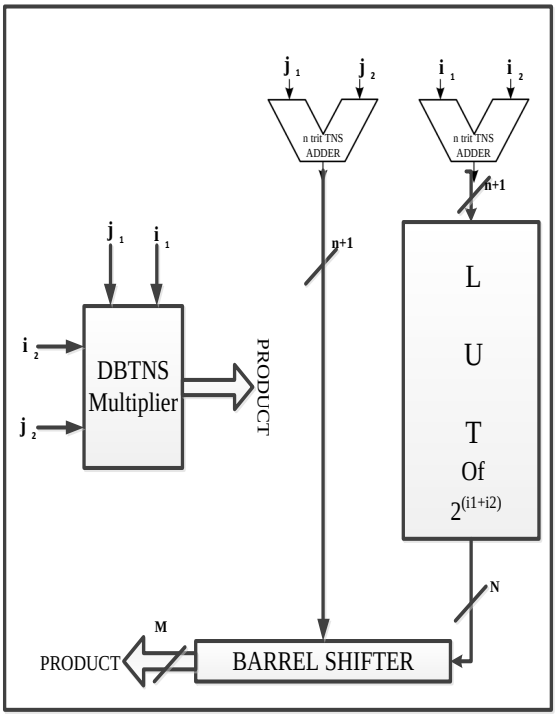

Figure 10. DBTNS Multiplier Unit

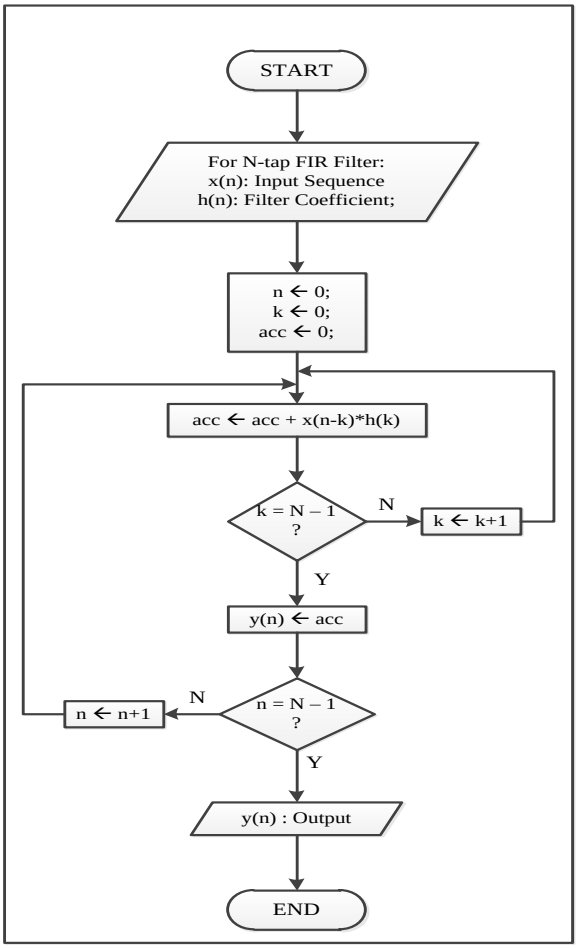

Figure 11. Computational flowchart of TNS based FIR Filter

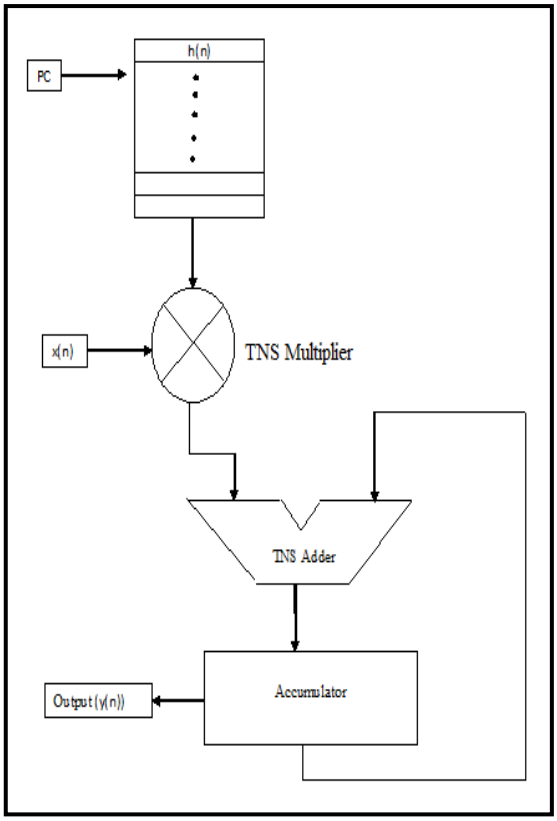

Figure 12. Block diagram of a proposed TNS MAC unit

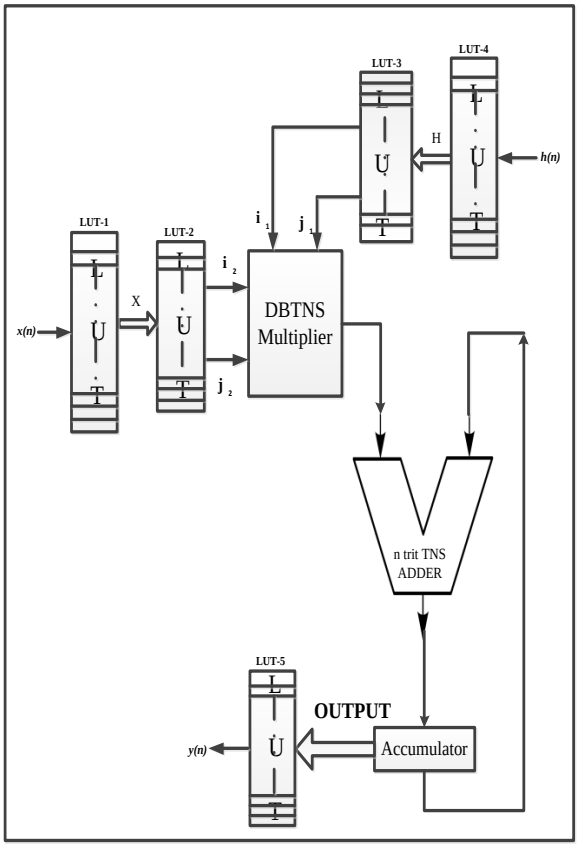

Figure 13. Block diagram of a proposed DBTNS MAC unit

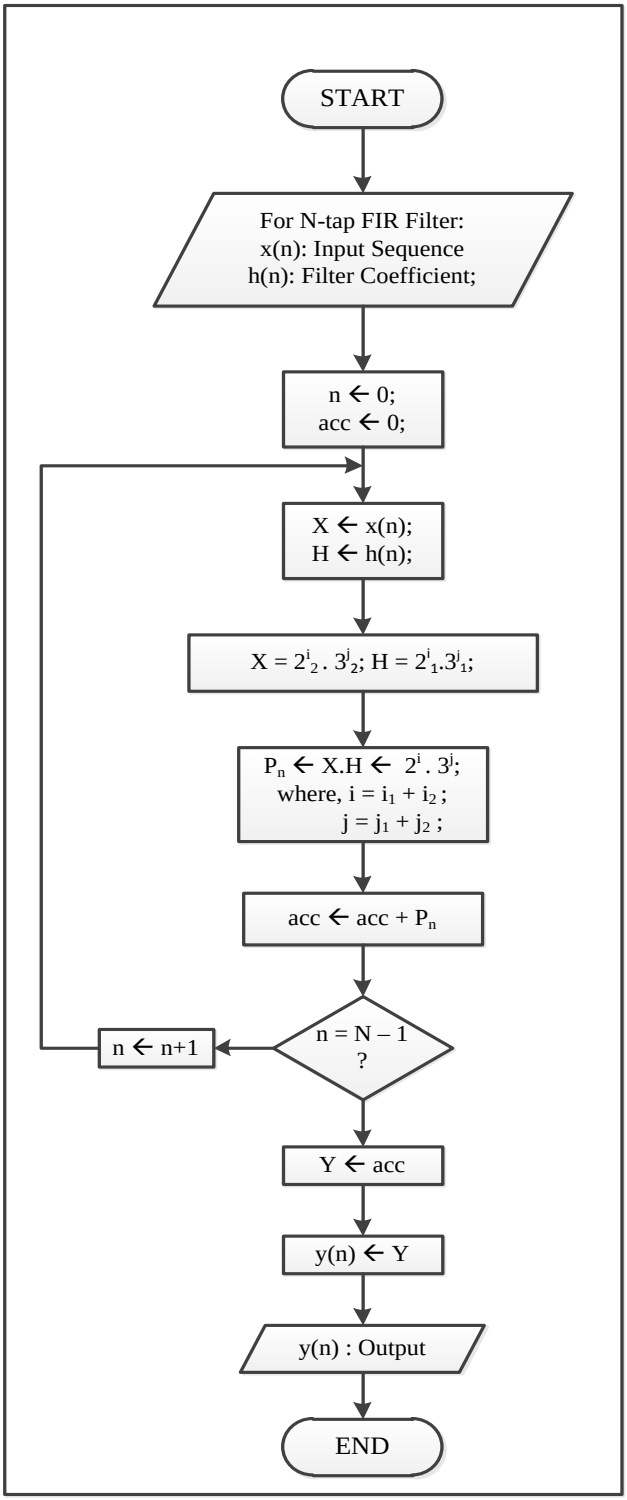

Figure 14. Computational flowchart of DBTNS based FIR Filter

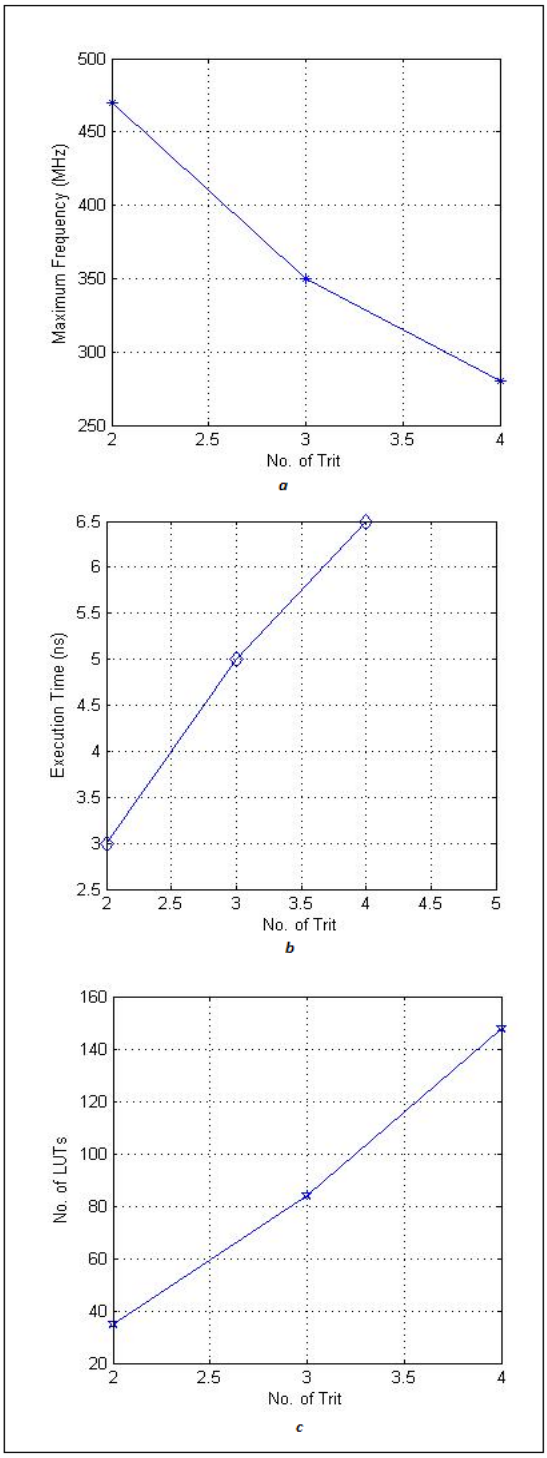

Figure 15. Complexity analysis of 8-tap FIR Filter using TNS MAC unit with the change of trit

(**a** - Maximum Frequency vs. Trit; **b** - Execution Time vs. Trit; **c** - No. of LUTs vs. Trit)

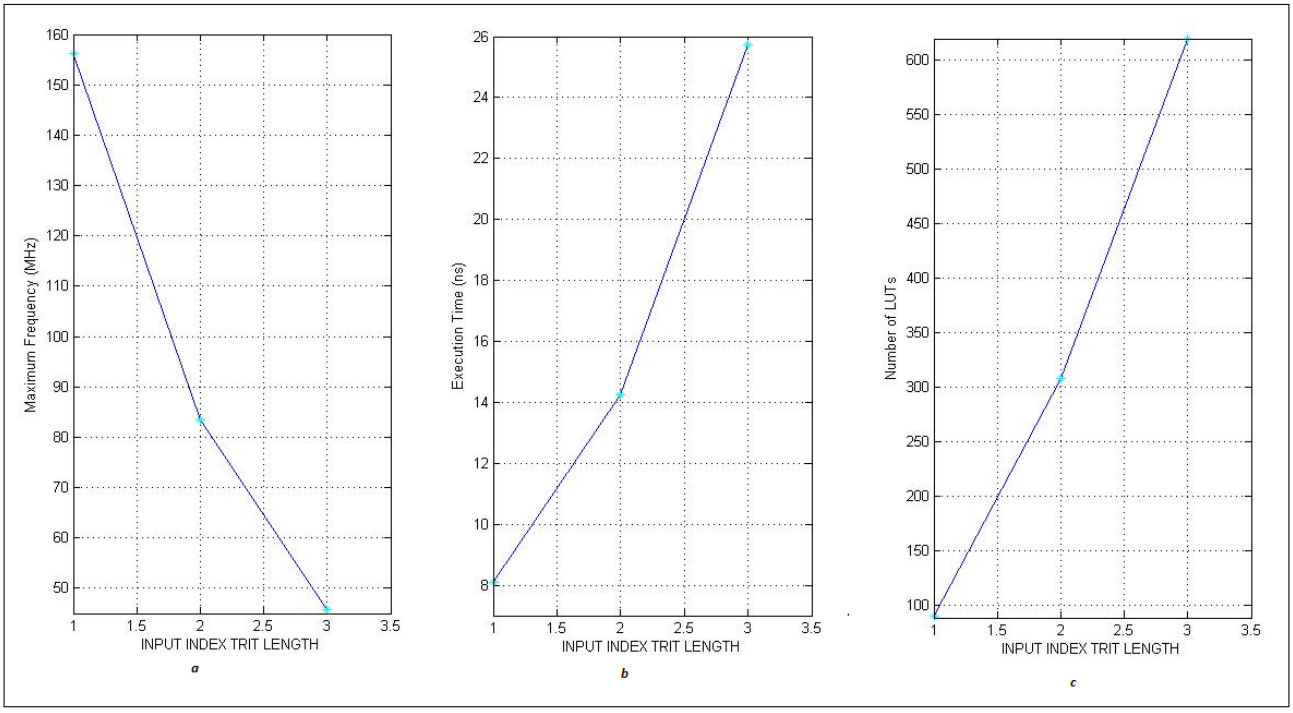

Figure 16. Complexity analysis of 8-tap FIR Filter using DBTNS MAC unit with the change of trit

(**a -** Maximum Frequency vs. Trit; **b -** Execution Time vs. Trit; **c -** No. of LUTs vs. Trit)