International Journal of Engineering Research and Technology.
ISSN 0974-3154 Volume 11, Number 1 (2018), pp. 11-27

http://www.irphouse.com

FPGA based Efficient N-Point FFT Architecture using CORDIC for Advanced OFDM

Shalini J[1],
[1](Department of ECE, K.S. School of Engineering and Management,Bangalore, Karnataka, India)

Dr. Y R Manjunatha[2]
[2](Department of EEE, University of Visvesvaraya College of Engineering(UVCE), Bangalore, Karnataka, India)

ABSTRACT

Now a day's wireless communication technique replaces most of the wired communication techniques. This is mainly due to the mobility issues of the end to end communication devices, which can be avoided in the case of wireless communications. For proper communications with less noise and covering large distance, all communications techniques are operated at very high speed in terms of GHz range. Most of the existing hardware architectures cannot be able to process data at such high speeds due to the presence of high delay in architecture. To overcome those issues, there is a need to address the efficiency of the architecture. In this paper, we propose FPGA based Efficient CORDIC based N-Point FFT Architecture for Advanced OFDM. The architecture of N point FFT has been improved in terms of speed and area utilization. The Vedic multiplication based Butterfly architecture improves the speed and modified CORDIC processor for twiddle factor calculation optimizes the area utilization. The proposed design is implemented using Spartan-6 FPGA Board and it is observed that the performance of the proposed design is better than existing in terms of various parameters.

Keywords - A-OFDM, Fast Fourier Transform, FPGA Implementation, Modified CORDIC Architecture, Vedic Multiplication etc.

I. INTRODUCTION

Digital Signal processing techniques are commonly used in almost all applications of electronic systems.. For proper operation of any digital signal processing applications we have to properly analyze the specified characteristics of the input signal. All natural signals are in time domain after digitization by sample and hold circuits. It is difficult to extract most of the characteristics from the time domain signals due to various algorithmic issues. To overcome this problem various different domain techniques are invented such as frequency domain, z-domain and s-domain etc. Most of the domain available for processing frequency domain processing gives easier detection and processing. This makes the frequency domain as most popular in the DSP applications.

There are various techniques existing to convert time domain signal to frequency domain signal. The most common techniques used are DFT, FFT and DCT etc. The FFT is commonly used transform domain technique which has various advantages over other domain transform techniques in terms of speed. The main disadvantages of FFT are the requirements of large amount of multipliers and also memory storage to store the corresponding twiddle factor value which depends on data points N. To overcome these problems we propose modified FFT which is the combination of both Vedic multiplications based FFT and CORDIC processor.

II. LITERATURE SURVEY

Prasant and Venkat [1] proposed design of Vedic multiplier compatible for implementation on FPGA. The adder blocks were used as compression elements to compress partial products which is further used to design Vedic multiplier of NxN with reduced hardware. The operating frequency of the design can be increased using “Urdha Tirayakbhyam” sutra. The proposed architecture is tested using Spartan-3 FPGA board. Sreelekshmi et al., [2] designed high speed MAC unit using vedic multiplier. The speed of the MAC unit is increased by high speed Vedic 32x32 bit multiplier using “Urdhva Tiryakbhyam” sutra and CSA (carry save adders) were used to replace normal adders. The proposed design is implemented using Xilinx ISE 10.1 and the coding is done by using HDL language with 24.1 ns delay. Gaurav Sharma et al., [3] compared the delays introduced by different types of adders present in Vedic multiplier architecture. The conventional adder present in the architecture is replaced by CSA (carry save adder), CLS (Carry Lookahead Adder) and RCA (ripple carry adder). These architectures are implemented using VHDL language and the delay introduced is measured on Xilinx Spartan-3 FPGA board. Pusphalata Verma [5] proposed efficient way to implement Vedic multiplier (4x4) using Electronic Design Automation (EDA) tool. The coding was done using VHDL language and simulated using Xilinx 12.1i EDA tool. The comparison of the proposed Vedic multiplier with conventional multiplier was done by synthesizing the proposed model in Spartan-3 FPGA board. The Vedic multiplier operated much faster than conventional multiplier using less hardware resources. Poornima et al., [6] proposed hardware implementation of multiplier using Vedic mathematics efficiently. The proposed method used ripple carry adders (RCA) to generate partial products. The implementation of 8x8 multiplier using performed by two 4x4 multipliers. The proposed design is synthesized using Spartan-3 FPGA board with VHDL language resulting in delay of 28.27 ns delay. Irine and Suchitra [7] proposed floating point arithmetic calculation based vedic multiplier. The accuracy of the proposed multiplier increase in-terms of truncation errors due to floating point calculations. The intermediate products were generated using Ripple Carry Adders. The procedure involves two phases having calculation unit and control unit. The mantissa and exponent calculation required for floating operations is performed in calculation unit and sign control operations is performed in control unit. The proposed architecture is synthesized and implemented on Virtex-2 FPGA board.

Aniket and Mayuresh [8] presented comparisons between various techniques used to implement FFT in hardware. The 256 and 1024 point FFT's were considered to implement for radix-2, radix-4 using rader-brenner algorithm. The result shows small decrease in hardware requirements. Abhishek Gupta et al., [9] proposed FFT processor with Vedic multiplier to increase the operating frequency with reduced area requirements. In this work, the modified structure achieved high frequency and the synthesis report proves that the operating frequency is more than existing. Subha Sri et al., [10] discussed advantages and disadvantages of corresponding architectures with respect to different type of CORDIC architectures. The folded structure, unfolded structure and parallel structures were used for these comparisons. The parallel structure shows high speed operation at the cost of area requirements. Suresh Kumar et al, [11] proposed hardware implementation of FFT architecture higher order FFT architecture. In the paper, the comparison of the performance of both radix-2 and radix-4 architecture were performed. The proposed architecture is implemented on Spartan-3 board with reduction in hardware resources in the case of radix-4 structure. Abhishek Gupta et al., [12] proposed FFT processor where normal multiplier is replaced with Vedic multiplier to increase the operating frequency and reduce area requirements. In the paper they modified the structure to achieve high frequency. The synthesis report proves that the operating frequency is more than existing. In the paper presented by Narayanam and Guravaiah [13], authors check the performance between Grigoryan FFT and Cooley-Tukey FFT algorithm. For this purpose the design of both the algorithms were performed in such way that it can be able implement in FPGA. For proper comparison the implementation of 8, 64, 128 and 256 point FFT using both algorithms were performed on both Xilinx Virex-II pro and Virtex-5 FPGA. The result shows the less area utilizations by of Grigoryan FFT than Cooley-Tukey FFT.

Sneha Kherde [14] proposed a new algorithm for implementing fast FFT. In this paper, the 8-point FFT using radix-2 algorithm was presented. The proposed architecture uses the complex multiplications for proper operation. In this paper, the author implemented 8-point DIT-FFT to convert time domain signal into frequency domain signal. The proposed architecture is coded using VHDL language and implemented using Xilinx ISE tool. The CORDIC (CO ordinate Rotation Digital Computer) algorithm is required to compute various complex mathematical trigonometric functions are discussed in literature [15]. The CORDIC implementation is area efficient; since the equations are reduced to add and shift operations in hardware, but have with trade off with respect to speed.

III. RELATED WORKS

All of the trigonometric functions are often calculated by the usage of vector rotations, which can be mentioned within the sequent sections. For polar two dimensional Cartesian and two dimensional Cartesian to polar conversions, for vector magnitude, and as a basic building block in bound transforms like the DCT and DFT vector rotation may be used.. With the use of only add and shift operations, the CORDIC algorithm [15] render an iterative method in achieving vector rotations with arbitrary angles.

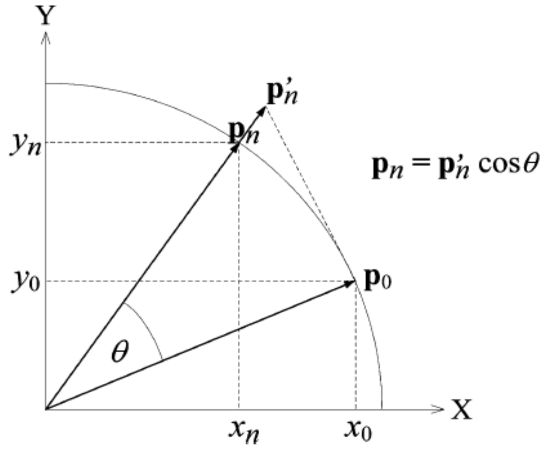

Fig.1. Rotation of vector on a two-dimensional plane

The angle calculation using CORDIC method is shown in Fig 1. This method is based on trial and error method for calculating the sine and cosine value of corresponding input angles. The equation for the rotation matrix R is

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

$$R = (1 + tan^2\theta)^{-\frac{1}{2}} \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \tag{2}$$

We can rewrite the above equation as

$$R_c = \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \tag{3}$$

$$K = (1 + tan^2\theta)^{-\frac{1}{2}} \tag{4}$$

So, now we can rewrite the equation (3) as

$$R_c(i) = K_i \begin{bmatrix} 1 & -\delta_i 2^{-i} \\ \delta_i 2^{-i} & 1 \end{bmatrix} \tag{5}$$

Where $k_i = \frac{1}{\sqrt{1+2^{-2i}}}$

By taking account all those equation we are able to write equation for CORDIC rotation mode [15]

$$X_{i+1} = X_i + \delta_i 2^{-i} Y_i \tag{6}$$
$$Y_{i+1} = Y_i - \delta_i 2^{-i} X_i \tag{7}$$
$$\omega_{i+1} = \omega_i - \delta_i \alpha_i \tag{8}$$

The basic block diagram of CORDIC rotation mode is shown in Fig. 2.

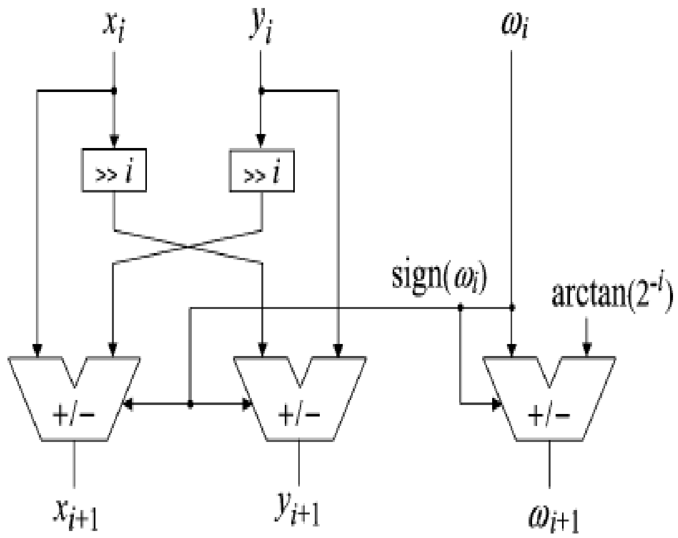

Fig.2. Basic Block Diagram of CORDIC in Rotation Mode.

III.I. Manual Calculation using CORDIC Algorithm

For this we consider θ=40° and the calculation is shown in Table-1.

Table 1: Manual Calculation using CORDIC Method

i	di	θ i	zi	Yi(Sinθi)	Xi(Cosθi)
0	+1	45	+40	0	0.6073
1	-1	26.6	-5	0.6073	0.6073
2	+1	14	+21.6	0.3036	0.9109
3	+1	7.1	+7.1	0.8313	0.8350
4	+1	3.6	+0.5	0.6356	0.7685
5	-1	1.8	-3.1	0.6836	0.7287
6	-1	0.9	-1.3	0.6608	0.7500
7	-1	0.4	-0.4	0.6490	0.7603
8	+1	0.2	0	**0.6430**	**0.7650**

From general method

Cos(40)= 0.76604444311897803520239265055542
Sin (40)= 0.64278760968653932632264340990726

From CORDIC method

Cos(40)=0.7650
Sin (40)=0.6430

Error introduced in CORDIC method

For Cos(40) error is 0.1044%
For Sin(40) error is 0.0212%
For θ error is 40.09 for cosine & 40.01 for sine.

III.II. Fast Fourier Transform (FFT)

The FFT is a form of Discrete Fourier Transform (DFT) [16]. The FFT reduce the time and computational complexity arises in DFT. This is a transform domain technique which converts time domain information to frequency domain information as shown in butterfly diagram given by Fig 3. The equation of forward and inverse FFT is given below

$$X(K) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi Kn}{N}} \quad (9)$$

$$x(n) = \sum_{K=0}^{N-1} X(K) e^{j\frac{2\pi Kn}{N}} \quad (10)$$

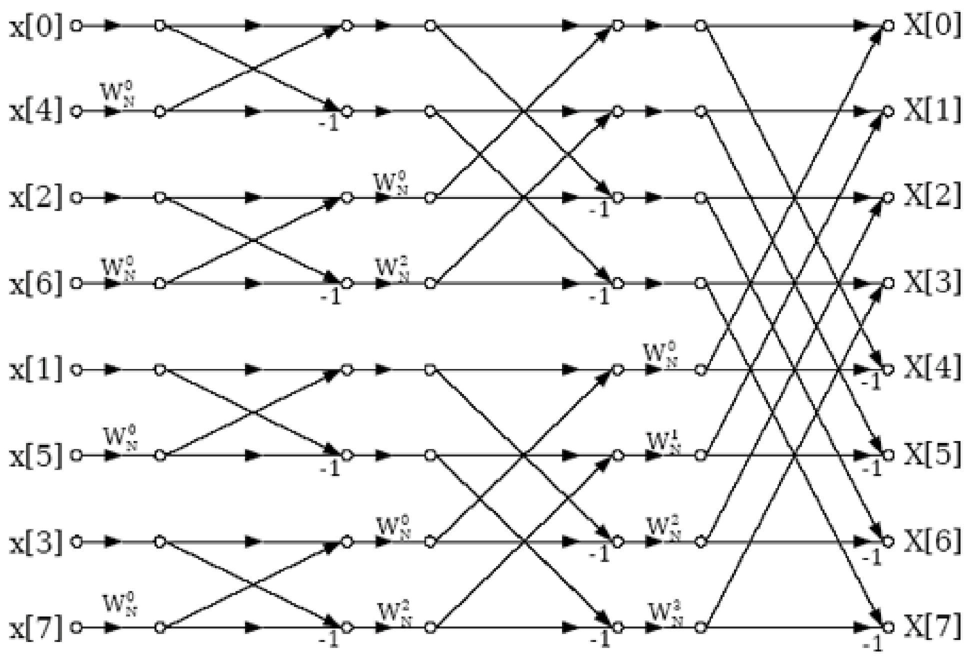

Fig. 3. Eight point DIT-FFT Butterfly Diagram

IV. PROPOSED WORK

The proposed FFT architecture based on CORDIC algorithm to compute the twiddle factor and Vedic multiplier is as shown in Fig. 4. The angles required for computation are fed in parallel based on N value to the proposed CORDIC. Further the twiddle factor values obtained are fed to the N point FFT structure to compute the output samples. The computation speed is increased due to vedic multiplication.

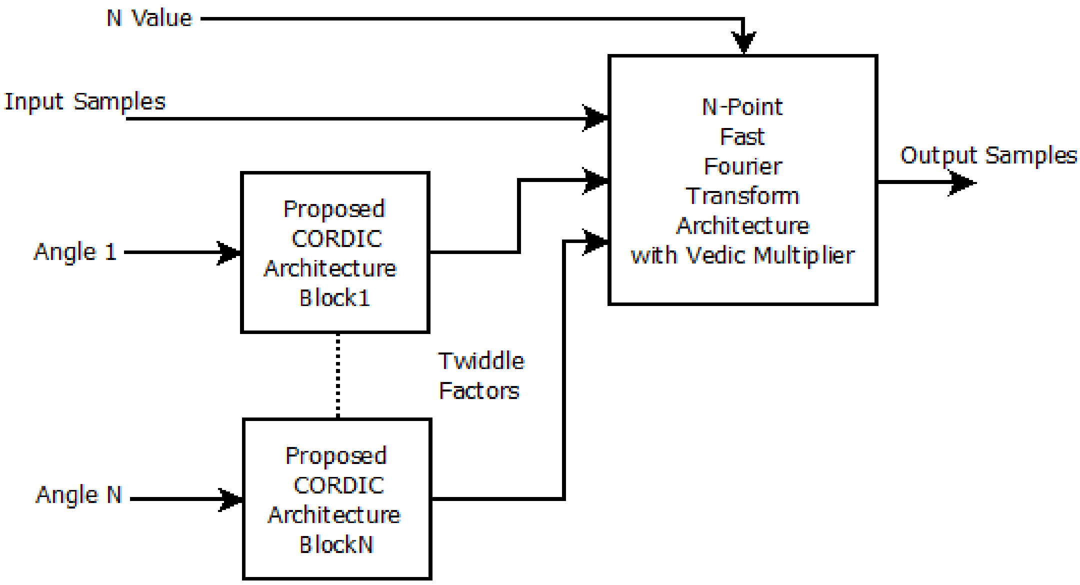

Fig.4. Proposed N-Point CORDIC and Vedic Multiplication Based FFT Architecture

IV.I. FFT Decomposition

The basic FFT equation is

$$X(K) = \sum_{n=0}^{N-1} x(n)\omega_N^{nk} = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi nk}{N}} \tag{11}$$

Let $\theta = -\frac{2\pi nk}{N}$ then

$$X(K) = \sum_{n=0}^{N-1} x(n)e^{\theta} \tag{12}$$

Using Euler's theorem we can decompose the above equation as

$$X(K) = \sum_{n=0}^{N-1} x(n)\{\cos\theta - j\sin\theta\} \tag{13}$$
$$X(K) = \sum_{n=0}^{N-1} x(n)\cos\theta - j\sum_{n=0}^{N-1} x(n)\sin\theta \tag{14}$$

Now by substituting the Θ value we can write

$$X(K) = \sum_{n=0}^{N-1} x(n)\cos(-\frac{2\pi nk}{N}) - j\sum_{n=0}^{N-1} x(n)sin\,(-\frac{2\pi nk}{N}) \quad (15)$$

From trigonometric theory we can write the FFT equation as

$$X(K) = \sum_{n=0}^{N-1} x(n)\cos(\frac{2\pi nk}{N}) + j\sum_{n=0}^{N-1} x(n)sin\,(\frac{2\pi nk}{N}) \quad (16)$$

The FFT module implementation is performed using equation where twiddle factor values are obtained using predefined angles of N-point FFT. Since the existing techniques use ROM based storage and Look Up Table method resulting in large storage and minimal speed. In our proposed method speed of operation of FFT block increases with less hardware and also the output provides a set of additional robust features which consists rapid change of phase information required for accurate matching.

IV.II. Vedic Multiplier

The multiplications required in the Butterfly diagrams are implemented by Vedic technique to reduce total hardware requirements. Four two input AND gates and two half adders are required to implement 2-bit multiplier which is shown in Fig .5. The hardware requirement of 2x2 bit Vedic multiplier is same as the hardware requirements of 2x2 bit conventional Array Multiplier. The multiplication of 2 bit binary numbers by the Vedic method does not make a significant effect on improvement of the multiplier's efficiency.

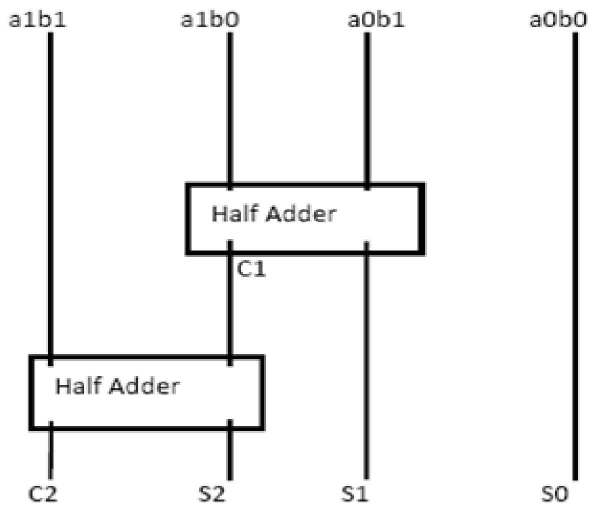

Fig.5. 2x2 Vedic Multiplication Architecture

The 4-bit multiplier [4-6] can be designed with the help four 2-bit Vedic multipliers. All the four multipliers generate the partial products in parallel. The generated products are added with the help of adder circuits. Here we are using the carry save adder with a modified full adder.

Let A and B are the two inputs to the multiplier each is of four bit width. Q is the product output of the multiplier which is of eight bit width. Hardware realization of four bit multiplier is shown in Fig. 6.

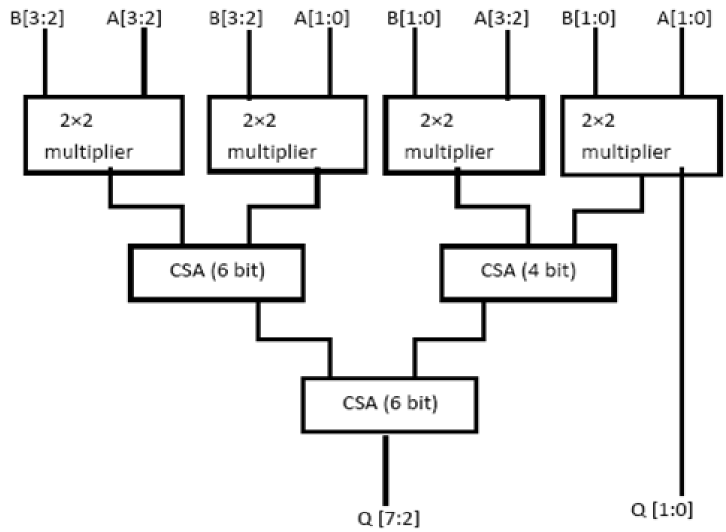

Fig. 6. 4x4 Vedic Multiplication Architecture

The Hardware realization of 8-bit Vedic multiplier [4-6] is shown in Fig. 7. It uses the four 4-bit multipliers to produce partial products and three carry save adders of eight and twelve bit width to produce the 16-bit product.

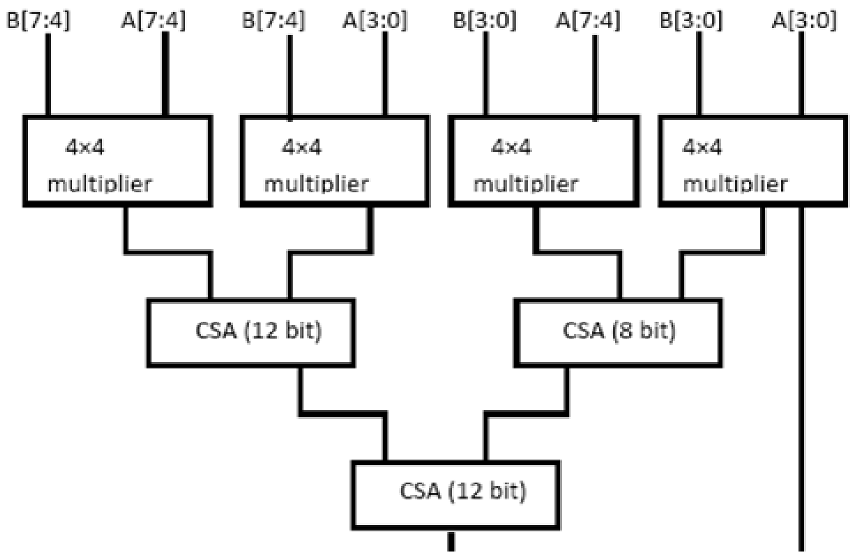

Fig. 7. 8x8 Vedic Multiplication Architecture

IV.III. Modified CORDIC Implementation

The basic CORDIC equation consists of a large amount of constant multiplier. We replace the constant multiplication by corresponding shifts to increase the operating frequency and reduce area. So, the modified equation for CORDIC [16] is given below.

$$X_{i+1} = X_i + \delta_i(\gg i)Y_i \tag{17}$$
$$Y_{i+1} = Y_i - \delta_i(\gg i)X_i \tag{18}$$
$$\omega_{i+1} = \omega_i - \delta_i\alpha_i \tag{19}$$

Where, (>>i) is left shift by i^{th} position.

Pseudo code:

```
Consider N=Iteration Step Length;
Consider Ө"=Input Angle;
Consider Ө=Intermediate Angle.
{
if(0° ≤ θ'' < 90°)
    θ = θ'';
    sign_sinθ = +ve;
    sign_cosθ = +ve;
else if(90° ≤ θ'' < 180°)
    θ = (180 − θ'');
    sign_sinθ = −ve;
    sign_cosθ = +ve;
else if(180° ≤ θ'' < 270°)
    θ = (270 − θ'');
    sign_sinθ = −ve;
    sign_cosθ = −ve;
else if(270° ≤ θ'' < 360°)
    θ = (360 − θ'');
    sign_sinθ = −ve;
    sign_cosθ = −ve;
else
    θ = θ'';
    sign_sinθ = +ve;
```

```
    sign_cosθ = +ve;
end if;
}
{
Consider X0=0.6073;
Consider Y0=0;
      for(i = 0; i ≤ N; i++)
            α_i = tan^-1(2^-i)
end for;
          if(0° ≤ θ ≤ 90°)
           if(0° ≤ θ ≤ 5°)
                 ∅ = 0°
      else if(85° ≤ θ ≤ 90°)
                ∅ = 90°
              else ∅ = θ;
                 end if;
          else ∅ = Invalid;
                 end if;
                X1 = X1 − Y0;
                Y1 = Y0 + X0;
                Z1 = (∅ − α0);
      for(j = 1; j ≤ N; j++)
          if(Zj is Positive)
             X_{j+1} = X_j − (Y_j * α_j);
             Y_{j+1} = Y_j + (X_j * α_ij);
                Z_{j+1} = (Z_j − α_j);
                   else
             X_{j+1} = X_j + (Y_j * α_j);
             Y_{j+1} = Y_j − (X_j * α_ij);
                Z_{j+1} = (Z_j − α_j);
                 end if;
                end for;
}
                     cos θ = X_N;
                      sin θ = Y_N;
```

The multiplier less CORDIC architecture is shown in the Fig.8 and also to increase parallelism in the structure, we replace the ROM used to store predefined angle by fixed constants.

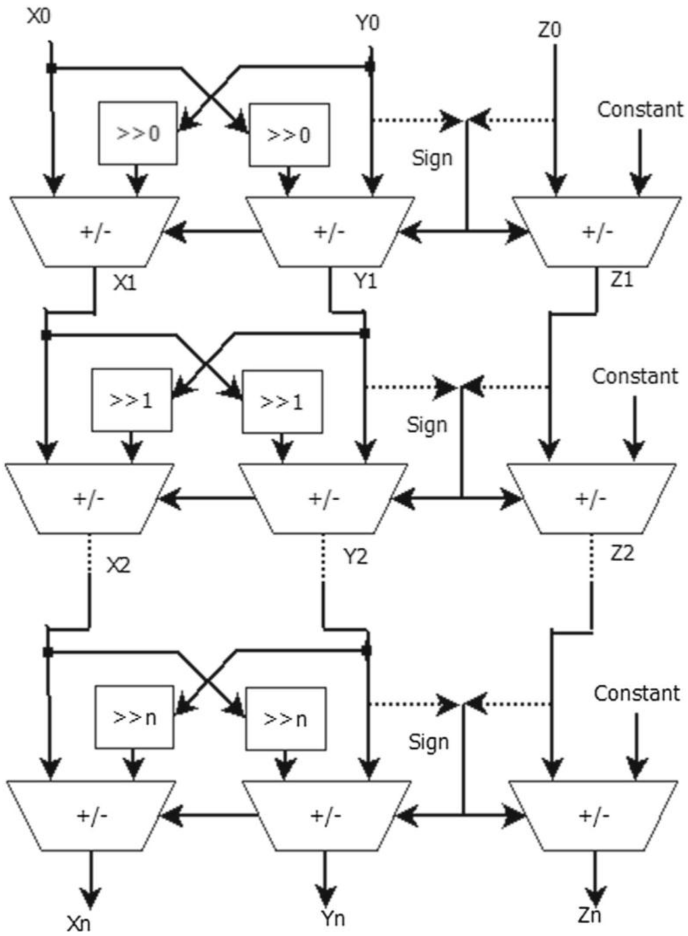

Fig. 8. Internal Structure of Modified CORDIC

V. RESULTS AND DISCUSSIONS

The Hardware descriptions for the proposed designs of reversible multipliers and linear contrast enhancement algorithm are written using VHDL and the designs are simulated and synthesized using Xilinx 14.5 ISE. The designs of proposed CORDIC based FFT are implemented on Spartan-6 FPGA with XC6SLX45 device, CSG324 package and speed grade of −3.

V.I. CORDIC Implementations

The implementation result of modified CORDIC is shown in Table 2. The proposed architecture uses 32 slice registers, 332 slice LUTs, 16 LUT flip-flop pairs and 37 IOBs. The maximum operating frequency of the block is 678.771 MHz.

Table 2: Hardware Utilizations of Proposed CORDIC Method

Parameters	Values
No. of Slice Registers	32
No. of Slice LUTs	332
No. of fully used LUT-FF pairs	16
No. of bonded IOBs	37
No. of BUFG/BUFGCTRLs	1
Maximum Operating Frequency (MHz)	678.771

V.II. Vedic Multiplier

The implementation result of Vedic Multiplier is shown in Table 3. The multiplier requires 116 slice LUTs and 32 IOBs for operation.

Table 3: Hardware Utilizations of Proposed Vedic Multiplier

Parameters	Values
No. of Slice LUTs	116
No. of fully used LUT-FF pairs	0
No. of bonded IOBs	32

V.III. FFT based on CORDIC and Vedic Multiplier

The implementation result of proposed FFT based on CORDIC and Vedic Multipier is shown in Table 4. It uses 475 slice registers, 2177 slice LUTs, 418 LUT-FF pairs and 24 DSP48A1s. The maximum operating frequency is 82.717 MHz. Those values are different than addition of the above table values which is due to proper synchronization purpose where some extra logic elements are needed.

Table 4: Hardware Utilizations of Proposed CORDIC-FFT

Parameters	Values
No.of Slice Registers	475
No.of Slice LUTs	2177
No. of fully used LUT-FF pairs	418
No. of bonded IOBs	138
No.of BUFG/BUFGCTRLs	1
No. of DSP48A1s	24
Maximum Frequency	82.717MHz

VI. COMPARISON OF PROPOSED METHOD WITH EXISTING METHODS

VI.I. Modified CORDIC

The comparisons of modified CORDIC and actual value are compared in the Table 5. In this table we can observe that the actual and calculated values are very near.

Table 5: Output Comparisons of trigonometric values with Proposed CORDIC

Input Angle	From CORDIC Method		Actual Value	
	CosΘ	SinΘ	CosΘ	SinΘ
0	0.9888	0.0976	1	0
10	0.9804	0.1757	0.9848	0.1736
20	0.9296	0.3554	0.9396	0.3420
30	0.8671	0.5039	0.8660	0.5000
40	0.7578	0.6484	0.7660	0.6427
50	0.6484	0.7578	0.6427	0.7660
60	0.5039	0.8671	0.5000	0.8660
70	0.3554	0.9296	0.3420	0.9396
80	0.1757	0.9804	0.1736	0.9848
90	0.0976	0.9888	0	1

Also comparisons in terms of hardware utilizations are given in Table. 6. The proposed architecture is compared with architecture presented by Yasodai and Ramaprasad [17] and Rudagi and Vinayak [18]. The comparison result shows that the proposed architecture is better than existing due to replacement of constant multiplier/divider by shifters.

Table 6: Comparisons of Existing CORDIC with Proposed CORDIC

Parameters	Yasodai and Ramaprasad [17]	Rudagi and Vinayak [18]	Proposed Technique
Board	Virtex-4	Virtex-6	Spartan-6
No. of Slices	520	965	32
No. of Slice Flip Flops	936	----	332

VI.II. CORDIC-FFT

The comparisons of proposed CORDIC-FFT with existing FFT are given in Table 7. The proposed method is better compared to existing techniques with improvement in speed and area utilization, since trigonometric method of CORDIC calculations for twiddle factor with FFT has been used.

Table 7: Comparison of existing FFT techniques with Proposed CORDIC-FFT

Parameters	Aman et al., [19]	Akashadip et al., [20]	Mayura and Vaishali [21]	Proposed Technique
FPGA Board	Spartan-3	Virtex-7	Spartan-3	Spartan-6
Number of Slice Registers	33280	1536	----	475
Number of Slice LUTs	66560	54465	----	2177
Number of fully used LUT-FF pairs	66560	----	----	418
Number of bonded IOBs	----	----	----	138
Number of BUFG/BUFGCTRLs	----	----	----	1
Number of DSP48A1s	----	----	----	24
Maximum Frequency (MHz)	----	32.897	----	82.717
Total Power (Watts)	----	----	0.609	0.037

VII. CONCLUSION

In this paper, the efficient hardware architecture of FFT algorithm for Advanced OFDM applications is proposed. In the proposed architecture, the multiplications presents in FFT butterfly architectures are replaced by the multiplier architecture derived from Vedic multiplication techniques. This reduces the area requirements and increase overall operating frequency. Moreover in the proposed architecture the existing ROM based twiddle factor implementation is replaced by modified CORDIC to reduce the uses of memory elements in the total architecture. That optimization gives the reduction in area and increase in operating frequency.

REFERENCES

[1] Prashant D. Pawale, Venkat N Ghodke, High speed Vedic multiplier design and implementation on FPGA, International Journal of Applied Research, 1(7), 2015, 239-244.

[2] Sreelekshmi M. S., Farsana F. J., Jithin Krishnan, Rajaram S and Aneesh R, Implementation of MAC by using Modified Vedic Multiplier, International Journal of Advanced Computer Research, 3(12), 2013, 11-15.

[3] Gaurav Sharma, Arjun Singh Chauhan, Himanshu Joshi and Satish Kumar Alaria, Delay Comparison of 4 by 4 Vedic Multiplier based on Different Adder Architectures using VHDL, International Journal of IT, Engineering and Applied Sciences Research, 2(6), 2013, 28-32.

[4] Premananda B.S., Samarth S. Pai, Shashank B. and Shashank S. Bhat, Design and Implementation of 8-Bit Vedic Multiplier, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(12), 2013, 5877-5882.
[5] Pushpalata Verma, Design of 4x4 bit Vedic Multiplier using EDA Tool, International Journal of Computer Applications, 48(20), 2012, 32-35.
[6] Poornima M, Shivaraj Kumar Patil, Shivukumar , Shridhar K P , Sanjay H, Implementation of Multiplier using Vedic Algorithm, International Journal of Innovative Technology and Exploring Engineering, 2(6), 2013, 219-223.
[7] Irine Padma B.T and Suchitra. K, Pipelined Floating Point Multiplier Based On Vedic Multiplication Technique, International Conference On Innovations & Advances In Science, Engineering And Technology, 2014, 130-137.
[8] Aniket Shukla and Mayuresh Deshmukh, Comparative Study Of Various FFT Algorithm Implementation On FPGA, IJETS, 1(1), 2012, 19-22.
[9] Abhishek Gupta, Amit Jain, Anand Vardhan Bhalla and Utsav Malviya, Design Of High Speed FFT Processor Using Vedic Multiplication Technique, IJERA, 2(5), 2012, 1501-1504.
[10] T.Subha Sri , K.Seshu Kumar and R. Muttaiah, Review of CORDIC Architectures, IJET, Vol. 5(2), 2013, 578-585.
[11] Suresh Kumar Dunna, B Vijaya Bhaskar and R Suryaprakash, Implementation of Higher Order FFT Processor Using FPGA, IJERT, 1(6), 2012, 1-4.
[12] Abhishek Gupta, Amit Jain, Anand Vardhan Bhalla and Utsav Malviya, Design Of High Speed FFT Processor Using Vedic Multiplication Technique, IJERA, 2(5), 2012, 1501-1504.
[13] Narayanam Ranganadh and Muni Guravaiah P, Performance Evaluations of Grigoryan FFT and Cooley-Tukey FFT onto Xilinx Virex-II pro and Virtex-5 FPGA, International Journal of Scientific and Research Publications, 3(1), 2013, 1-6.
[14] Sneha Kherde, Design and Implementation of FFT, International Journal of Electronics, Communicatin and Soft Computing Science and Engineering, 2015, 248-252.
[15] P.K. Meher, J. Valls, T.-B. Juang, K. Sridharan, K. Maharatna, 50 Years of CORDIC: algorithms, architectures and applications. IEEE Transactions on Circuits and Systems, 56(9), 2009, 1893–1907.
[16] Satish S. Bhairannawar, Sayantam Sarkar, K. B. Raja and K. R. Venugopal, Implementation of Fingerprint Based Biometric System using Optimized 5/3 DWT Architecture and Modified CORDIC Based FFT, International Journal of Circuits, Systems and Signal Processing, Springer, 37(1), 2018, 342-366.
[17] Yasodai .A and Ramprasad.A.V, A New Memory Reduced Radix-4 CORDIC Processor for FFT Operation, IOSR Journal of VLSI and Signal Processing, 2(5), 2013, 9-16.

[18] J. M. Rudagi and Vinayak Dalavi, Radix-2 CORDIC Method with Constant Scale Factor, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(7), 2013, 3408-3413.
[19] Arman Chahardahcherik, Yousuef S. Kavian, Otto Strobel and Radha Rajeb, Implementing FFT Algorithm on FPGA, International Journal of Computer Science and network Security, 11(11), 2011, 148-156.
[20] Akashadip A. Jinwane, Prashant R. Indurkar and Ravindra D. Kadam, Design and Simulation of Floating Point FFT Processor based on Radix-4 Algorithm using VHDL, International Journal of innovative Research in Computer and Communication Engineering, 4(7), 2017, 13223-13229.
[21] Mayura Patrikar and Vaishali Thre, Design and Power Measurement of 2 and 8 Point FFT using Radix-2 Algorithm for FPGA Implementation, IOSR Journal of VLSI and Signal Processing, 7(1), 2017, 44-48.