Dhaka Univ. J. Sci. 58(1): 67-72 2010 (January)

Implementation of DFT (Direct & FFT) Using Radix-2-4 CORDIC Algorithm

Meher Nigar[1] and Suraiya Pervin[2]

[1]Therap (BD) Ltd., [2]Department of Computer Science and Engineering, Dhaka University, Dhaka-1000, Bangladesh

Received on 25.11.2008. Accepted for Publication on 02.06.2009

Abstract

This paper attempts to implement the fast and hardware efficient mathematical technique for computing CORDIC (Coordinate Rotation Digital Computer) based DFT (Direct & radix-2 butterfly FFT), using original circular rotation of the mixed radix CORDIC based method. CORDIC algorithms are hardware efficient methods of producing a wide range of trigonometric, hyperbolic, linear and logarithmic functions using only shift and add operations. To speed up the CORDIC based DFT (Direct & radix-2 butterfly FFT) calculation a new method has been proposed. This new scheme reduces the last $N/2$ CORDIC iterations to $N/4$ by using radix-2-4 CORDIC algorithm, where the required precision is N. Radix-2-4 CORDIC algorithm uses radix-4 micro rotation in the second half of the CORDIC iterations. This scheme reduces the number of iterations in the CORDIC rotation unit by about 25%. Reduction of iterations also trims down the calculation time.

Keywords: CORDIC, DFT, FFT.

I. Introduction

Discrete Fourier Transform (DFT) [1], is a powerful computational tool for analyzing the characteristics of discrete-time signals in the frequency domain. A number of papers have been published on efficient algorithms for computing DFT. Among these, CORDIC [2] based algorithm gained significant attention as it reduces the number of multipliers. But it encounters the disadvantage of large number of iterations, which impedes the speed & performance in practical implementations, which also makes the hardware realization costlier. Many different versions of the classic CORDIC algorithm have been developed to enhance the performance of computing these elementary functions. High radix CORDIC algorithm [3]-[6] is the outcome of such novel effort of the researchers that is very fast and hardware effective compared to the basic CORDIC algorithm.

The main focus of this paper is the calculation of efficient, fast and cost effective CORDIC based DFT using direct computation & FFT (Fast Fourier Transform) algorithm. In this work, an attempt has been taken to introduce a new mathematical technique to compute CORDIC based DFT. This development is based on mixed radix CORDIC algorithm in order to improve the speed and cost of computation significantly while maintaining the performance.

The organization of the remaining part of this paper is as follows: the basic equation of DFT (Direct & FFT) is given in section-II. Section-III deals with CORDIC algorithm. In section-IV the radix-2-4 CORDIC based DFT & FFT computation technique is proposed followed by hardware analysis of the proposed method in section-V.

II. DFT & FFT

For a complex-valued sequence $x(n)$ of N points, the DFT may be expressed as

$$X_R(k) = \sum_{n=0}^{N-1}[x_R(n)\cos 2\pi kn/N + x_I(n)\sin 2\pi kn/N]$$

$$X_I(k) = -\sum_{n=0}^{N-1}[x_R(n)\sin 2\pi kn/N - x_I(n)\cos 2\pi kn/N]$$

where, $x(n) = x_R(n) + x_I(n)$

$$X(k) = X_R(k) + X_I(k),\quad 0 \le k \le N-1 \tag{1}$$

Radix-2 algorithms are by far the most widely used FFT [1] algorithms. The equation of decimation-in-time algorithm is

$$X(k) = \sum_{n\ even} x(n)W_N^{nk} + \sum_{n\ odd} x(n)W_N^{nk} \tag{2}$$

While computing FFT the basic computation performed at every stage is to take two complex numbers, the pair (p,q), multiply q by W_N^k, and then add and subtract the product from p to form two new complex numbers (X,Y). So the equations are, $X = p + W_N^k q$ and $Y = p - W_N^k q$ (3)

III. The CORDIC Algorithm

In the late 1950's J.E.Volder [2] developed the underlying method of computing the rotation of a vector in a Cartesian coordinate system and evaluating the length and angle of a vector. For rotation mode, the CORDIC equations are,

$$x_{i+1} = x_i - d_i 2^{-i} y_i,$$

$$y_{i+1} = y_i + d_i 2^{-i} x_i$$

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i})$$

$$\text{Where, } d_i = \begin{cases} -1\ if\ z_i < 0 \\ +1\ if\ z_i \ge 0 \end{cases} \text{ and } i = 0,1,2,\ldots,n-1 \tag{4}$$

Here, i denotes the iteration step, x_0 and y_0 are the coordinates of the initial vector and z_0 is the rotation angle, d_i is specifying the direction of rotation. It can be noted that the CORDIC method performs rotations only within $-\pi/2$ and $\pi/2$. This limitation comes from the use of 2^0 for the tangent in the first iteration. However, since a sine wave is symmetric from quadrant to quadrant, every sine value from 0 to 2π can be represented by reflecting and/or inverting the first quadrant appropriately.

High radix CORDIC rotation

When a higher radix [3], [4] is considered, more bits of the result are calculated each iteration, reducing the total number of iterations. In radix r, where r is a power of two, the rotation angle is decomposed into a series of elementary angles, or micro rotation angles, whose values are $\alpha_i = \tan^{-1}(d_i.r^{-i})$ (i = 1,2,........., n). The coefficients d_i can take values from the set $\{-r/2,....,0...,r/2\}$. This is a minimally redundant coefficient set, over which each angle may present different decompositions. For radix higher than 4, the d_i coefficients are not integer powers of two and the complexity of each iteration increases significantly [4]. When radix-4 is used, the total number of iterations of the CORDIC algorithm is halved. In [3] a radix-4 CORDIC algorithm has been developed for the rotation of a vector in circular coordinates (rotation mode). A new selection function for the calculation of the d_i coefficients has also been proposed and valid both for carry save redundant arithmetic and nonredundant arithmetic. The radix-4 CORDIC algorithm is given by the following recursive equations:

$$x_{i+1} = x_i - d_i 4^{-i} y_i$$
$$y_{i+1} = y_i + d_i 4^{-i} x_i$$
$$z_{i+1} = z_i - \alpha_i,$$
$$\alpha_i = \tan^{-1}(d_i.r^{-i}) \quad (\text{i} = 1,2,.........,\text{n}) \tag{5}$$

Here i denotes the iteration step, α_i is the angle of the i^{th} rotation and the rotation digit $d_i \in \{-2,-1,0,1,2\}$. x_0 and y_0 are the coordinates of the initial vector and z_0 the rotation angle. Consequently, x_{i+1} and y_{i+1} are the coordinates of the vector resulting from applying $i+1$ micro rotations and z_{i+1}, the angle remaining to be rotated. The final coordinates are scaled by,

$$K = \coprod_{i \ge 0} K_i = \prod_{i \ge 0} \sqrt{1 + d_i^2 4^{-2i}} \tag{6}$$

The scale factor is not constant as it depends on the sequence of d_i's. This factor must be evaluated for each rotation angle and compensated to preserve the norm of the vector. The selection function has been obtained by using the scheme proposed in [3]. It is interesting to note that in some cases it is better to begin the radix-4 micro rotations from $i = n/2+1$ to have a constant scale factor [6].

For $i = 0$ For $i > 0$

$$d_i = \begin{cases} +2 \; if & 5/8 \le w_0^{'} \\ +1 \; if & 3/8 \le w_0^{'} \le 5/8 \\ 0 \; if & -1/2 \le w_0^{'} \le 3/8 \\ -1 \; if & -7/8 \le w_0^{'} \le -1/2 \\ -2 \; if & w_0^{'} < -7/8 \end{cases} \quad d_i = \begin{cases} +2 \; if & w_0^{'} \ge 3/2 \\ +1 \; if & 1/2 \le w_0^{'} < 3/2 \\ 0 \; if & -1/2 \le w_0^{'} < 1/2 \\ -1 \; if & -3/2 \le w_0^{'} \le -1/2 \\ -2 \; if & w_0^{'} < -3/2 \end{cases} \tag{7}$$

IV. Proposed Method

Radix-2-4 CORDIC based Direct Method

Equation (1) can be written as

$$X_R(k) = \sum_{n=0}^{N-1} \left[\cos(2\pi kn/N)\left(x_R(n).+x_I(n).\tan 2\pi kn/N\right)\right]$$
$$X_I(k) = \sum_{n=0}^{N-1} \left[\cos(2\pi kn/N)\left(x_I(n).-x_R(n).\tan 2\pi kn/N\right)\right]$$

for $0 \le k \le N-1$ (8)

Now let, $\theta = 2\pi kn/N$, $x = x_I(n)$, $y = x_R(n)$. Then (8) will become

$$X_I(k) = \sum_{n=0}^{N-1} \left[\cos\theta\left(x - y.\tan\theta\right)\right]$$
$$X_R(k) = \sum_{n=0}^{N-1} \left[\cos\theta\left(y.+x.\tan\theta\right)\right] \tag{9}$$

In the CORDIC method the rotation by an angle θ is implemented as an iterative process consisting of a sequence of micro rotations θ_i such that $\theta = \sum_i \theta_i$; during which the initial vector is rotated by some predetermined step angles. So, the multiplication by the tangent term in (9) can be avoided if the rotation angles and therefore $\tan\theta$ are restricted such that, $\tan\theta_i = \pm r^{-i}$ for $i = 0,1,2,\ldots,m-1$ and $r = 2\, or\, 4$. More precisely, for radix-2 CORDIC: $\tan\theta_i = \pm 2^{-i}$, $i = 0,1,2,\ldots,g-1$ and for radix-4 CORDIC: $\tan\theta_i = \pm 4^{-i}, i = g,\ldots,m-1$. In digital hardware this denotes a simple shift operation. Then (9) can be written as

$$X_I(k) = \sum_{n=0}^{N-1} \left[\sum_{i=0}^{m-1} \left[\cos\theta\left(x - y.d_i.r^{-i}\right)\right]\right]$$
$$X_R(k) = \sum_{n=0}^{N-1} \left[\sum_{i=0}^{m-1} \left[\cos\theta\left(y + x.d_i.r^{-i}\right)\right]\right] \tag{10}$$

The value of $\sum_{i=0}^{m-1} \cos\theta$ converges to an aggregate constant value, 0.607253 (scale factor) when i starts at zero. Thus (10) becomes:

$$X_I(k) = \sum_{n=0}^{N-1} \left[0.607253 \sum_{i=0}^{m-1} \left(x - y.d_i.r^{-i}\right)\right]$$
$$X_R(k) = \sum_{n=0}^{N-1} \left[0.607253 \sum_{i=0}^{m-1} \left(y + x.d_i.r^{-i}\right)\right] \tag{11}$$

For radix-2 CORDIC: $d_i \in \{-1,1\}$ and for radix-4 CORDIC: $d_i \in \{\pm 0, \pm 1, \pm 2\}$. From the previous knowledge, it can easily be inferred that, the calculation of DFT for each data points (for each value of k) can be fully implemented by mixed CORDIC algorithm. To preserve the accuracy of the computation $r = 2$ has been considered for the first $n/2$ iterations, where n is the required precision. And in the rest $n/4$ iterations $r = 4$ has been considered. There by saving $n/4$ iteration's time & cost as compared to the traditional radix-2 CORDIC based DFT implementation. In performance analysis no significant difference is seen due to this changed in radix. The multiplication by 0.607253 can be replaced by a simple right-shift operation i.e. multiplication by 0.5 in hardware realization. In that case only an extra shifter is to be added in the CORDIC block. Another important adaptation is that, to ensure $\cos\theta$ to converge to 0.607253, the θ must be in the 1st or 4th quadrant of the coordinate system, that is in the interval $-90^0 \le \theta \ge 90^0$. So, if θ falls in 2nd or 3rd quadrant, it is plotted in the corresponding 2nd and 3rd quadrant and to reflect this change, the data sequence of input signal is negated before entering them into the CORDIC block.

Radix-2-4 CORDIC based Radix-2 Butterfly FFT

In Radix-2 butterfly FFT, at each butterfly the calculations performed are

$$X = p + qe^{-j2\pi k/N}, \; Y = p - qe^{-j2\pi k/N} \quad (12)$$

Considering the second part of the equation, q is complex-valued. So, its DFT can be written as,

$$\begin{aligned} Q_I &= q_I \cos(2\pi k/N) - q_R \sin(2\pi k/N) \\ Q_R &= q_R \cos(2\pi k/N) + q_I \sin(2\pi k/N) \end{aligned} \quad (13)$$

Now let, $\theta = 2\pi kn/N$. Then (13) will become

$$Q_I = \cos\theta[q_I - q_R \tan\theta]$$

$$Q_R = \cos\theta[q_R + q_I \tan\theta] \quad (14)$$

As stated in direct computation of DFT, the same scheme is also applicable in case of FFT. So, (14) finally becomes:

$$\begin{aligned} Q_I &= \left[0.607253 \sum_{i=0}^{m-1} \left(q_I - q_R d_i r^{-i}\right)\right] \\ Q_R &= \left[0.607253 \sum_{i=0}^{m-1} \left(q_R + q_I d_i r^{-i}\right)\right] \end{aligned} \quad (15)$$

For the proper implementation $r = 2$ has been considered for the first $n/2$ iterations, where n is the required precision. And in the rest $n/4$ iterations $r = 4$ has been considered. The multiplication by 0.607253 can be replaced by a simple right-shift operation. The basic operations in an FFT are multiplication of the complex data inputs by the FFT coefficients at each stage in the signal flow graph followed by their summation or subtraction. Hence, the only thing left is to do the addition and subtraction of the result returned from the CORDIC with the data P. That is to perform the operation,

$$\begin{aligned} X &= p + \text{Scaled CORDIC OUTPUT} \\ Y &= p - \text{Scaled CORDIC OUTPUT} \end{aligned} \quad (16)$$

V. Simulation & Hardware Analysis

For 64 bit precision 64 radix-2 CORDIC micro rotations are needed. But if radix-2-4 CORDIC is used then the same precision can be obtained by using only 48 micro rotations. The proposed method has been simulated for different input signals. In every case the proposed method achieves almost the same result as in direct DFT & radix-2 butterfly FFT.

DFT (Direct) and radix-2-4 CORDIC based DFT (scale factor 0.607253)

In the proposed method the CORDIC block is invoked N^2 times to compute N point DFT. Each invocation of CORDIC module requires two extra multiplications to multiply the CORDIC output by the scale factor 0.607253. Now-a-days table look up based approach is used to evaluate the trigonometric functions; so, the multiplication & addition needed to calculate these functions have been omitted in the calculation. Each CORDIC block contains 3 adders and 2 shift registers. So, N^2 no of CORDIC invocation requires $3N^2$ additions. Again equation (11) shows that another $2N(N-1)$ numbers of addition are needed. Table 1, 2 and 3 shows some comparative results, Figure 1 shows the output signal and Figure 2 & 3 shows the hardware comparison. For 64-bit precision roughly 0.0000107364356667062495% deviation of output result is got compared to the output of direct DFT. Its output is almost equal to the radix-2 CORDIC based DFT.

Table. 1. Radix-2-4 CORDIC based DFT and Direct computation of DFT.

	Radix-2-4 CORDIC based DFT	Direct Computation of DFT
Computation Time	$N^2(g+f)t_c + 2N^2 t_m + 2N(N-1)t_a$	$4N^2 t_m + 4N(N-1)t_a$
Hardware Consulted	N^2 *CORDICBlock* $+ 2N^2$ *multiplications* $+ 2N(N-1)$ *additions*	$4N^2$ *multiplications* $+ 4N(N-1)$ *additions*

Notes:

N = equalizer order, (N point DFT),	t_c = time required to perform one CORDIC iteration.
t_a =time required to perform one addition,	t_m = time required to perform one multiplication.
g = number of radix-2 iterations,	f = number of radix-4 iterations

Table. 2. Radix-2 and radix-2-4 CORDIC based DFT. Where $m > (g+f)$

	Basic CORDIC based DFT	Radix-2-4 CORDIC based DFT
Computation Time	$N^2 mt_c + 2N^2 t_m + 2N(N-1)t_a$	$N^2(g+f)t_c + 2N^2 t_m + 2N(N-1)t_a$

Table. 3. Radix-2-4 CORDIC based DFT and direct computation of DFT.

	Radix-2-4 CORDIC based DFT	Direct Computation of DFT	Result
Multiplications	$2N^2$	$4N^2$	Reduced by 50%.
Additions	$5N^2 - 2N$	$4N^2$	Increased by $N(N+2)$.

Input Signal: $7\sin(8\pi n)+\cos(2\pi n)$

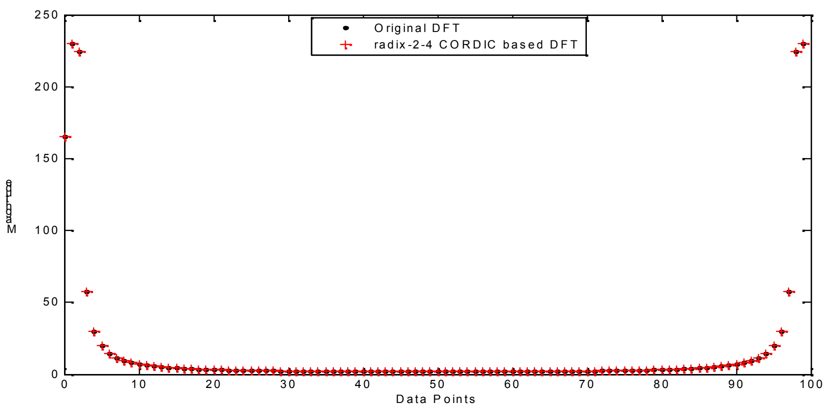

Fig.1. Comparing the two output signals (DFT, Scale factor 0.607253)

Table.4. Radix-2-4 CORDIC based and direct computation of DFT (scale factor 0.5).

	Radix-2-4 CORDIC based DFT	**Direct Computation of DFT**	**Result**
Multiplications	*NIL*	$4N^2$	Total elimination of multiplications.
Additions	$5N^2-2N$	$4N^2$	Increased by $N(N+2)$.

DFT (Direct) and radix-2-4 CORDIC based DFT (scale factor 0.5)

The multiplication by 0.5 can be replaced by a simple right-shift operation in hardware realization. So the need for multiplication is totally eliminated. The Table 4 reflects the result. With this architecture, the simulation program shows roughly 17% deviation in result when radix-2-4 CORDIC based DFT is used in comparison with direct DFT.

Traditional FFT and radix-2-4 CORDIC based FFT (scale factor 0.607253)

As described in the proposed method, for each butterfly one CORDIC block is needed. So, for N points FFT, the CORDIC block is invoked $(N/2)\log_2 N$ number of times. Two multiplications are needed in the FFT architecture to multiply the CORDIC output by the scale factor. $(N/2)\log_2 N$ numbers of CORDIC invocation requires $3*(N/2)\log_2 N$ additions. Again equation (7) shows that $4N\log_2 N$ additions are needed to add and subtract the output of the CORDIC unit. Table 5, 6 and 7 shows some comparative results. For 64 bits precision roughly 0.000010760225177720726% deviation of output result as compared to the output of traditional computation of FFT. Its output is almost equal to the radix-2 CORDIC based FFT. Figure 4 shows the comparison of the output signals and Figure 5 & 6 shows the hardware comparison.

Traditional FFT and radix-2-4 CORDIC based FFT (Scale factor 0.5)

In hardware realization, multiplication by 0.5 can be totally replaced by a simple right-shift operation. The Table 8 reflects the result. With this architecture, the simulation program shows roughly 17% deviation in result when radix-2-4 CORDIC based FFT is used in comparison with traditional FFT.

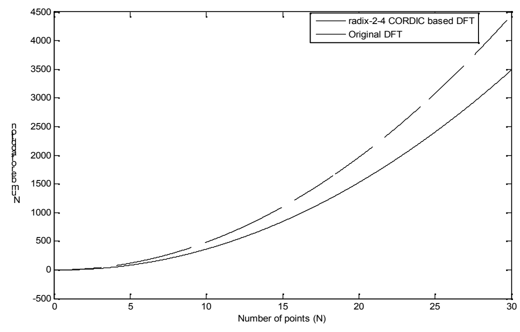

Fig. 2. Comparison of Addition required (DFT, Scale factor 0.607253)

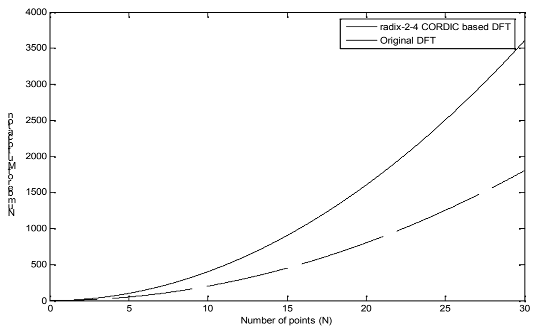

Fig. 3. Comparison of Multiplication required (DFT, Scale factor 0.607253)

Input Signal: $7\sin(8\pi n)+\cos(2\pi n)$

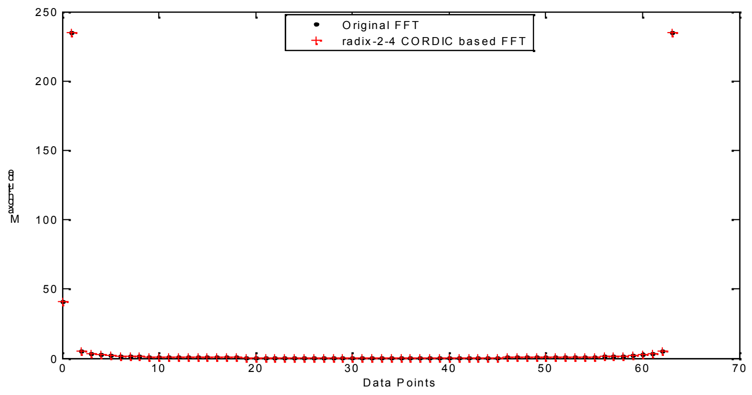

Fig.4. Comparing the two output signals (FFT, Scale factor 0.607253)

Table. 5. Radix-2-4 CORDIC based and traditional computation of FFT.

	Radix-2-4 CORDIC based FFT	**Traditional Computation of FFT**
Computation Time	$[(N/2)\log_2 N][g+f]t_c+[N\log_2 N]t_m+[4N\log_2 N]t_a$	$[2N\log_2 N]t_m+[4N\log_2 N]t_a$
Hardware consulted	$[(N/2)\log_2 N]$*CORDIC Block* $+[N\log_2 N]$ *multiplications* $+4N\log_2 N$ *additions*	$[2N\log_2 N]$ *multiplications* $+[4N\log_2 N]$ *additions*

Table. 6. Radix-2 and radix-2-4 CORDIC based FFT. Where $m>(g+f)$

	Basic CORDIC based FFT	**Radix-2-4 CORDIC based FFT**
Computation Time	$[(N/2)\log_2 N]mt_c+[N\log_2 N]t_m$ $+[4N\log_2 N]t_a$	$[(N/2)\log_2 N][p+q]t_c+[N\log_2 N]t_m$ $+[4N\log_2 N]t_a$

Table. 7. Radix-2-4 CORDIC based and Traditional computation of FFT.

	Radix-2-4 CORDIC based FFT	**Traditional Computation of FFT**	**Result**
Multiplications	$N\log_2 N$	$2N\log_2 N$	Reduced by 50%.
Additions	$((3N/2)+4N)\log_2 N$	$4N\log_2 N$	Increased by $3*(N/2)\log_2 N$.

Table.8. Radix-2-4 CORDIC based and Traditional computation of FFT (scale factor 0.5).

	Radix-2-4 CORDIC based FFT	**Traditional Computation of FFT**	**Result**
Multiplications	*NIL*	$2N\log_2 N$	Total elimination of multiplications.
Additions	$((3N/2)+4N)\log_2 N$	$4N\log_2 N$	Increased by $3*(N/2)\log_2 N$.

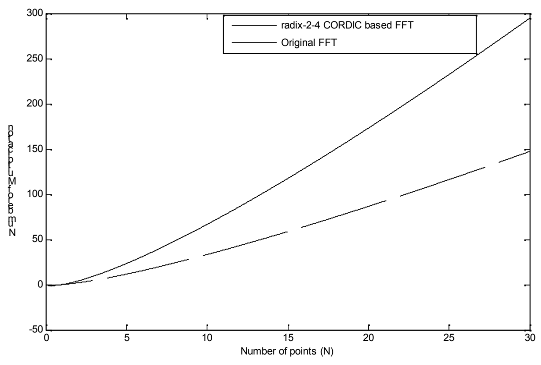

Fig.5. Comparison of Multiplication required (FFT, Scale factor 0.607253)

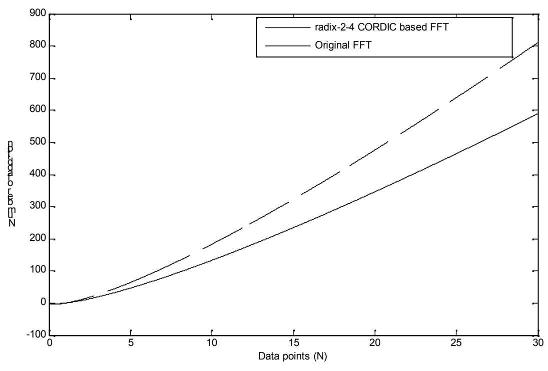

Fig.6. Comparison of Addition required (FFT, Scale factor 0.607253)

VI. Conclusion

The effort in this paper has been directed towards the computation of DFT (Direct and FFT) using the internal step by step rotations, which consists of only shift and add operations, of the radix-2-4 CORDIC with a view to reduce the calculation time. In this work, the computation of DFT (Direct & FFT) based on radix-2-4 CORDIC algorithm is proposed. The result shows that the computation of DFT (Direct & radix-2 butterfly FFT) can be implemented using the radix-2-4 CORDIC block. In both cases 50% reduction in multiplication has been achieved, by using the scale factor 0.607253. The use of radix-2-4 CORDIC with scale factor 0.5 (which can be done by only right shifting) in computing DFT fully eliminates the use of multipliers which further trim down the hardware cost and complexity tremendously. But that occurs in cost of around 17% deviation of result to the actual DFT result.

----

1. Proakis G. John & Dimitris. "Digital Signal Processing" 3rd edition, Prentice Hall of India.
2. Volder J.E., September 1959, "The CORDIC Trigonometric computing technique". IRE Trans. Electron Comput., EC **8(3)**, 330-334.
3. Antelo Elisardo, Villalba Julio, Bruguera D. Javier and Zapata L. Emilio, August 1997, "High Performance Rotation Architectures Based on the Radix-4 CORDIC Algorithm". IEEE Transactions on computers, 46(8).
4. AOKI Takafumi, Kitaori Ichiro and Higuchi Tatsuo, June 2000, "Radix-2-4-8 CORDIC for fast Vector Rotation". IEICE Trans. Fundamentals, E83-A (6).
5. Lee J.A. and T. Lang, August 1992, "Constant-Factor Redundant CORDIC for Angle Calculation and Rotation". IEEE Trans. Computers, **41(8)**, pp. 1016-1025.
6. Antelo Elisardo and Bruguera D. Javier, 1995, "Redundant CORDIC rotator based on parallel prediction". IEEE Trans. Comp.