CORDIC Based DFT on FPGA for DSP Applications

Padma. V[1]
PG Scholar, Department of E.C.E
SKIT College
Srikalahasti, India

Sudhakara Reddy. P[2] Member IEEE
Associate Professor, Department of E.C.E
SKIT college
Srikalahasti, India

Abstract— **Discrete Fourier Transform (DFT) is a very useful algorithm, playing an important role in various digital signal processing (DSP) applications from radar, sonar, telecommunication, image processing etc. The fast Fourier transform (FFT) is a class of algorithms for efficiently computing DFT. So, the FFT is also widely used in many DSP applications. The computation complexity of DFT is drastically reduced by using FFT. The multiplications as plain rotation in the DFT/FFT algorithm, it is possible to apply a pipelined coordinate rotation digital computer (CORDIC) algorithm for the implementation of DFT/FFT. Hence, the CORDIC technique is applied for calculating a vector rotated through a given angle for computation of FFT. Further by exploiting some trigonometric identities in the DFT/FFT computation CORDIC rotators are effectively used. Here, the implementation of CORDIC based 16-point DFT on FPGA for various DSP applications is presented. The data rate of the DFT is depends only on carry look ahead adder used and number of pipelining stages used. With the advent of VLSI Technology, it is possible to perform each pipeline step is less than 2-5ns and also increases the throughput as per the input data rate. In this we simulate and implement a pipelined CORDIC DFT using FFT algorithm on FPGA for DSP applications. By rewriting the DFT, for $N = 2^n$ point DFT, it requires $2^{n-2}(3n-13) + 4n - 2$ real multiplications and $2^{n-2}(7n-29) + 6n + 2$ real Additions for a real data. The computation of multiplications in DFT is done as plane rotations. So, it is possible to apply a pipelined CORDIC algorithm in a hardware implementation of a long-point DFT using of a single CORDIC pipeline.**

Keywords— CORDIC, DFT, DSP, FFT etc.

I. INTRODUCTION

Discrete Fourier transform (DFT) and Inverse Discrete Fourier transform (IDFT) are used in a wide range of applications, such as, signal processing, image processing, data compression, etc., for transforming signals from time domain to frequency domain and vice-versa. Exponential increase in networking bandwidth and storage capacity has created a need for high throughput DFT/IDFT circuits. Further, low complexity circuits are preferable over high complexity circuits, since, in general, they consume less power and are less expensive from the silicon real-estate perspective. Hence, low complexity DFT/IDFT circuits that have a high throughput are desirable. The Discrete Fourier Transform (DFT) is a very useful algorithm, playing an important role in various digital signal processing (DSP) applications. According to the definition, for N - point DFT/IDFT, $4N^2$ real multiplications are needed. To reduce the computational complexity of DFT, it is required to reduce the number of multiplications and additions in the DFT. The fast Fourier Transform (FFT) algorithm is the best solution why because it uses $2^{n+1} + n + s$ real multiplications for $N = 2^n$ - point FFT. In addition to software realizations of FFT, various hardware implementation of FFT have been designed to meet the high speed real-time requirements of DFT applications. Briefly, an FFT is an algorithm which provides a relatively fast means of computing a DFT by reducing the number of operations from approximately N^2 to $2N\, log_2 N$ as compared to a DFT; this is generally accomplished by reducing the redundant operations and of course also provides a corresponding decrease in computation processing time[1],[12].

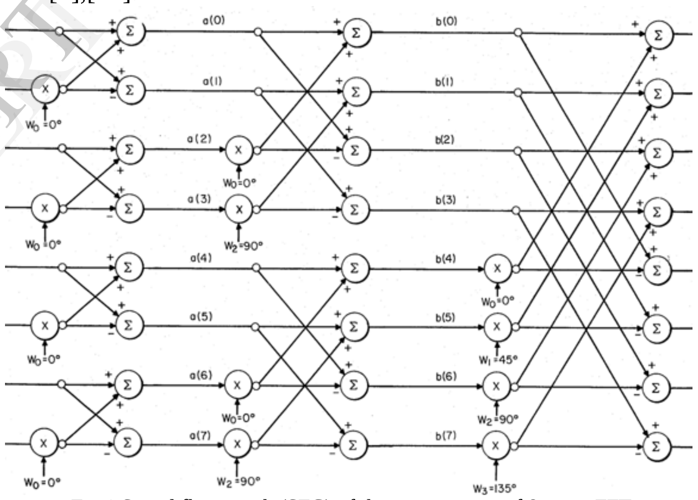


Fig.1 Signal flow graph (SFG) of the computation of 8-point FFT

The inputs, $s(0)$ through $s(7)$ and generally $s(n)$, are a time series of discrete samples. The corresponding outputs are frequency components i.e $S(0)$ to $S(7)$.

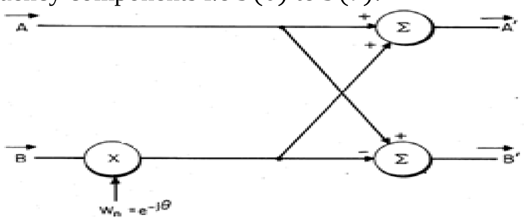


Fig.2 The basic building block of an FFT

The basic building block of an FFT as shown in Figure. 2 is a vector rotation by a complex weighting coefficient, W_n, and a sum and difference of the result with another vector. If the un-rotated vector is $A = R_1 + jI_1$, the rotated vector is

B = $R_2 + \mathrm{j}I_2$, and $w_n = e^{-j\theta}$, the output of the basic building block is given

$$A^{'} = R_1 + (R_2 cos\theta + I_2 sin\theta) + j[I_1 + (I_2 cos\theta - R_2 sin\theta)]$$
$$B^{'} = R_1 - (R_2 cos\theta + I_2 sin\theta) + j[I_1 - (I_2 cos\theta - R_2 sin\theta)] \quad (1)$$

As can be seen from the equation above, A' and B' can be calculated in a digital computer by a combination of four multiplies and six adds in addition to looking up the trigonometric values in a table 1. Recently, the coordinate rotation digital computer (CORDIC) technique was introduced as a method of calculating a vector rotated through a given angle. This technique has also been applied to FFT's.

Table.1 Trigonometric values in a Look up table

	90^0	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
$W_0 = 0^0$	1	0	0	0	0	1	0	1	1	0	0	0	0	1	1	0	0	1
$W_1 = 22.5^0$	1	0	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	1
$W_2 = 45^0$	1	0	0	1	1	1	1	1	0	0	0	0	0	1	0	1	0	1
$W_3 = 67.5^0$	1	0	1	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0
$W_4 = 90^0$	1	0	1	1	1	0	1	0	0	1	1	1	1	0	0	1	1	0
$W_5 = 112.5^0$	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	1
$W_6 = 135^0$	1	1	0	1	1	1	1	1	0	0	0	0	0	1	0	1	0	1
$W_7 = 157.7^0$	1	1	1	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0

However, there is still a need for a CORDIC unit and a digital processor in general wherein increased throughputs can be realized. Furthermore, some applications require that an FFT processor with a high throughput rate also have the capability to be programmable such that FFT's of different numbers of points can be performed. In CORDIC, The plane rotations are involve mainly two trigonometric factors of the expression

$$A(k) = \sum_{n=0}^{N-1} a(n) cos\frac{2\pi}{N}kn \quad (2)$$
$$B(k) = \sum_{n=0}^{N-1} b(n) sin\frac{2\pi}{N}kn \quad (3)$$

are allowing the sums to be split in halves and recombined recursively. The resulting recursive equations are fairly simple, employing plane rotations applied to terms referenced by indices obtained by reversing certain bit patterns. Because of the way multiplications and additions are organized as plane rotations in our DFT algorithm, it is possible efficiently to apply a pipelined CORDIC algorithm in a hardware implementation of the long-point DFT. [11]

In section II mathematical representation of DFT/IDFT for both real and complex data are shown. Section III discusses CORDIC based DFT/IDFT and possible hardware implementations, In section IV are explained the results analysis and implementation. Finally, Section V gives conclusion.

II. MATHEMATICAL REPRESENTATION OF DFT/IDFT

A. Real data DFT/IDFT transformations

The $N = 2^n$ real data DFT/IDFT transformations are defined as follows:

$$F(k) = \frac{1}{N}\sum_{n=0}^{N-1} f(n) e^{-j\frac{2\pi}{N}kn}$$
$$= \frac{1}{N}\sum_{n=0}^{N-1} f(n)\, cos\frac{2\pi}{N}kn - j\frac{1}{N}\sum_{n=0}^{N-1} f(n)\, sin\frac{2\pi}{N}kn, \quad (4)$$
$$f(n) = \sum_{k=0}^{N-1} F(k) e^{j\frac{2\pi}{N}kn}$$
$$= \sum_{k=0}^{N-1} F_r(k)\, cos\frac{2\pi}{N}kn - \sum_{k=0}^{N-1} F_r(k)\, sin\frac{2\pi}{N}kn, \quad (5)$$

Where all $f(n)$ are real data and then $F(k)$ are complex,

$$F(k) = F_r(k) + jF_i(k); k, n = 0,1,2, \dots\dots. N-1 \quad (6)$$

for the computation DFT/IDFT we shall only discuss the following expressions

$$A(k) = \sum_{n=0}^{N-1} a(n) cos\frac{2\pi}{N}kn,$$
$$B(k) = \sum_{n=0}^{N-1} b(n) sin\frac{2\pi}{N}kn,$$
$$k, n = 0,1,2, \dots\dots. N-1$$

According to DFT/FFT properties, the symmetry property further reduces the computational complexity, that means $A(N-k) = A(k)$ and $B(N-k) = -B(k)$, k can be limited to the interval 0 to $N/_2$. So, the $A(k)$ and $B(k)$ $k = 0,1,2, \dots\dots. \frac{N}{2} - 1$ are

$$A(k) = \sum_{n=1}^{N/2-1} [a(n) + a(N-n)] cos\frac{2\pi}{N}kn + a\left(\frac{N}{2}\right) cos\pi k + a(0) \quad (7)$$
$$B(k) = \sum_{n=1}^{N/2-1} [b(n) + b(N-n)] sin\frac{2\pi}{N}kn \quad (8)$$

To continue the splitting, finally for each L, then $A(k)$ and $B(k)$ for $k = L, 3L, 5L, \dots, \frac{N}{2} - L$ are shown as

$$A\ (k) \ = \ p_L^{\frac{N}{8L}}(N_k), \quad (9)$$
$$B\ (k) \ = \ q_L^{\frac{N}{8L}}\left(\frac{N}{8L} - N_k\right). \quad (10)$$

The above computations are iterated until $= \frac{N}{8L}$, this containing trigonometric factors and compute $A(\ k)$ and $B(k)$, $k = 0, 2L, 4L$. It is observed that, the above expressions containing trigonometric factors in the form of plane rotations, which can be realized by the CORDIC algorithm.

For the DFT: $a(n) = b(n) = f(n)$
$$F(k) \ = \ A(k) - jB(k),$$
$$F(N-k) \ = \ A(k) + jB(k) \quad (11)$$
For IDFT: $a(k) \ = \ F_r(k)$,
$$b(k) \ = \ F_i(k)$$
$$f(n) \ = \ A(n) - B(n),$$
$$f(N-n) \ = \ A(n) \ + \ B(n). \quad (12)$$

A diagram of the computations for $N = 64$ DFT using the above expressions is shown in Figure 1. Now we can calculate how many multiplications and additions are needed for an $N = 2^n$DFT/IDFT. For any value of L, the number of CORDIC operations for computing $A(k)$ is

$$1 + \frac{N}{16L} log_2 \frac{N}{8L}. \quad (13)$$

When L runs through the values $1,2,4, \dots, N/8$, the total number of all CORDIC operations is then

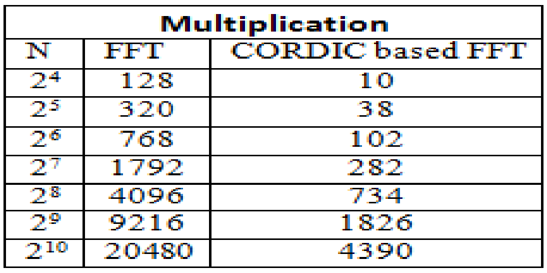
$$\sum_L \left(1+\frac{N}{16L}\log_2\frac{N}{16L}\right) = \frac{N}{8}\log_2\frac{N}{16}+\log_2\frac{N}{2} \qquad (14)$$

All A(n) and B(n) values can be split to find an equivalent number of additions and multiplications:

$$2\left(\frac{N}{8}\log_2\frac{N}{16}+\log_2\frac{N}{2}\right) = [2^{n-2}(n-5)+2(n-1)] + [2^{n-2}-2(n-2)+2(n-2)] \qquad (15)$$

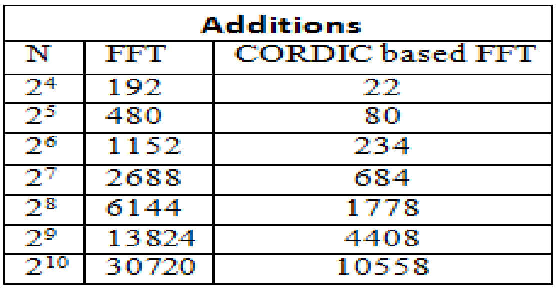
In which each of $2(n-2)$ CORDIC operations computing $p_L^1(0)$ and $p_L^1(\frac{N}{8L})$ need 1 multiplication and 2 additions, each of $2^{n-2}-2(n-2)$ CORDIC operations whose angle is just equal to $\frac{\pi}{4}$ need only2 multiplications and $2+4$ additions , and each of $2^{n-2}(\mathrm{n}-5)+2(\mathrm{n}-1)$ CORDIC operations can be realized by using 3 multiplications and $3+4$ additions. Therefore the total number of real multiplications required $2^{n-2}(3\mathrm{n}-13)+4\mathrm{n}-2$, and the number of real additions required is $2^{n-2}(7\mathrm{n}-29)+6\mathrm{n}+2$. The traditional Fast Fourier Transform (FFT) uses $2\mathrm{n}2^n$ real multiplications and $3\mathrm{n}2^n$ additions. The complex multiplication is realized by 3 real multiplication and 3 real additions. We need only $2^{n-1}(\mathrm{n}-3)+2$ multiplications and $2^{n-1}(3\mathrm{n}-5)+4$ additions. Table 1 shows a comparison between these and our method for different values of N. From the table we see that for large values of N our method use a few more multiplications, but fewer additions. But the total number of operations is lower, which is important if addition and multiplication take the same time, as in many DSP Processors. In the above we have just discussed the DFT/IDFT for real data.

Table 2. Number of real multiplications in FFT and CORDIC based FFT for DFT algorithms.



Multiplication		
N	FFT	CORDIC based FFT
2^4	128	10
2^5	320	38
2^6	768	102
2^7	1792	282
2^8	4096	734
2^9	9216	1826
2^{10}	20480	4390

Table 3. Number of real additions in FFT and CORDIC based FFT for DFT algorithms.



Additions		
N	FFT	CORDIC based FFT
2^4	192	22
2^5	480	80
2^6	1152	234
2^7	2688	684
2^8	6144	1778
2^9	13824	4408
2^{10}	30720	10558

B. *Complex data DFT/IDFT transformations:*

For complex data

$$\mathrm{f(n)} = f_r(n) + jf_i(n),$$

we have , $\mathrm{F(k)} = \frac{1}{N}\sum_{n=0}^{N-1}\mathrm{f(n)e}^{-j\frac{2\pi}{N}kn}$

$$= \frac{1}{N}\sum_{n=0}^{N-1}[\mathrm{f_r(n)+jf_i(n)}]\cos\frac{2\pi}{N}kn +$$

$$\frac{1}{N}\sum_{n=0}^{N-1}[\mathrm{f_i(n)-jf_r(n)}]\sin\frac{2\pi}{N}kn, \qquad (16)$$

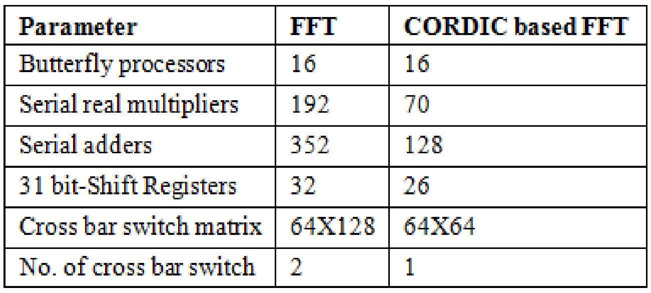
$$\mathrm{F(n)} = \sum_{k=0}^{N-1}\mathrm{F(k)e}^{j\frac{2\pi}{N}kn} + \sum_{k=0}^{N-1}[\mathrm{F_r(k)+jF_i(k)}]\cos\frac{2\pi}{N}kn - \sum_{k=0}^{N-1}[\mathrm{F_i(k)+jF_r(k)}]\sin\frac{2\pi}{N}kn. \qquad (17)$$

So for complex data, using the new algorithm, the number of real multiplications and real additions is twice the number of operations for real data. Compared to the traditional FFT/IFFT algorithm, the new algorithm requires more data address computations, as shown in equations (2),(3), (4) and (5). However, it is easy to realize these data address computations in Hardware or by table look-up, noting that $\sum_{j=1}^{M-1}\mathrm{a_j}2^{M-1-j}$ in binary is just a part of the reversed bit pattern of the binary representation for $k=\sum_{j=0}^{M-1}\mathrm{a_j}2^j$.

III. CORDIC BASED DFT/IDFT

In CORDIC based DFT/IDFT using FFT/IFFT algorithm, it is found that, the number of real multiplications and real additions is respectively reduced to $2^{n-2}(3n-13)+4n-2$ and $2^{n-2}(7n-29)+6n+2$ for real data DFT. The main advantage of this proposed algorithm is to reduce the total number of floating point operations is very less and smaller. Hence, first it is required to rewrite the DFT in terms of FFT either in decimation-in-time (DIT) or decimation-in-frequency (DIF) secondly, expresses the FFT computations in terms of the plane rotations. The number of such rotations are $2^{n-2}(n-4)+2(n-1)$ needed for $N=2^n$- point FFT. The number of additions and multiplications quoted above corresponds to the implementation of each rotation as a complex multiplication.

Table 4. Comparison of some performance parameters



Parameter	FFT	CORDIC based FFT
Butterfly processors	16	16
Serial real multipliers	192	70
Serial adders	352	128
31 bit-Shift Registers	32	26
Cross bar switch matrix	64X128	64X64
No. of cross bar switch	2	1

We iteratively use 16 butterfly processors. Figure 3 shows one butterfly processor architecture. Finally all F(k) are calculated(Note actually only 32 values of $F(k)=A(k)-jB(k)$ are calculated, since $F(N-k)=A(k)+jB(k)$, the other 32 values are easily obtained). Considering all types of butterfly processors computations, it is possible to design a butterfly processor based on standard multipliers as shown in Figure 3.

Where c1, sl, s2 are used to control the computations. Of the 16 butterfly processors, 6 needs five multipliers and the other 10 need four multipliers.

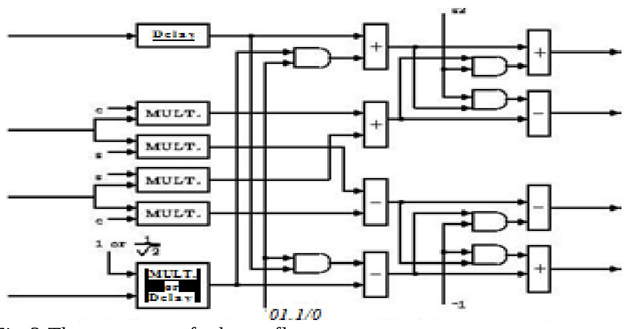


Fig.3.The structure of a butterfly processor

For IDFT, $F_r(k) = F_r(N-k)$ and $F_i(k) = -F_i(N-n)$ Because $f(n) = A(n) - B(n)$ and $f(N-n) = A(n) + B(n)$ for IDFT, an extra step is needed at the end for those computations. The remaining computations are the same as for the DFT. In order to conveniently construct a long point DFT/IDFT by using a pipelined CORDIC processors. It is better design comparatively standard form of DFT/IDFT. A CORDIC computation performs a vector rotation:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} sin\alpha & cos\alpha \\ cos\alpha & -sin\alpha \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} , \qquad (18)$$

by a number of micro rotations using a predetermined set of angles $\emptyset_i$ derived from α. CORDIC processors would be needed to achieve that input rates.

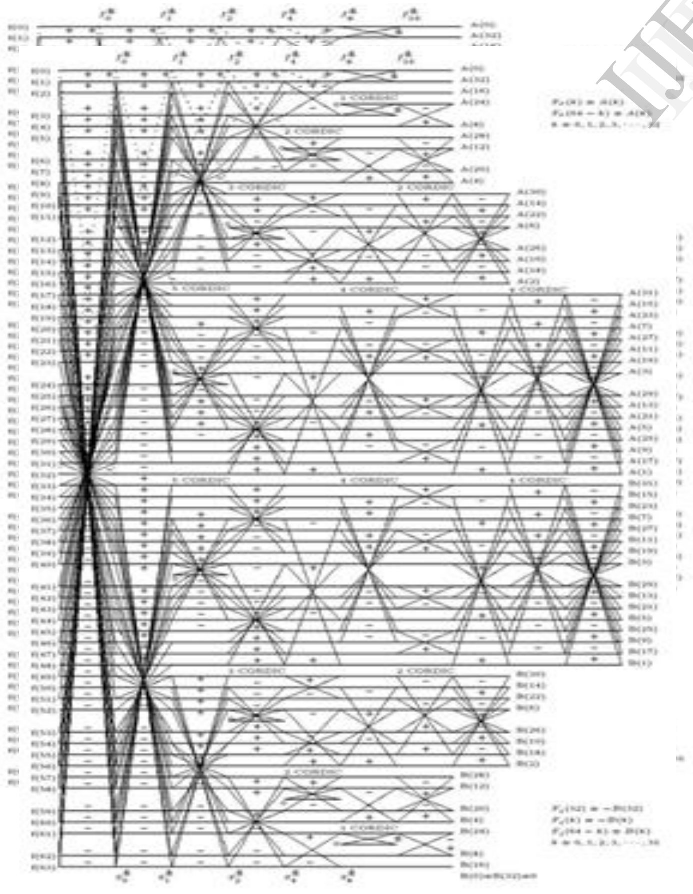
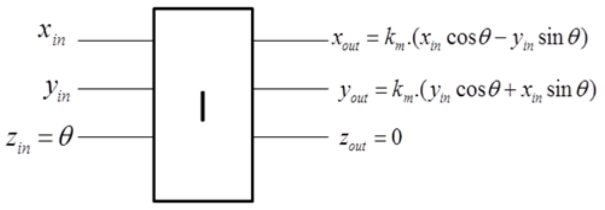


Fig.4 64-point CORDIC based DFT computing algorithm


x_{in}
y_{in}
$z_{in} = \theta$
I
$x_{out} = k_m.(x_{in}\cos\theta - y_{in}\sin\theta)$
$y_{out} = k_m.(y_{in}\cos\theta + x_{in}\sin\theta)$
$z_{out} = 0$
Fig.5 Circular rotation mode

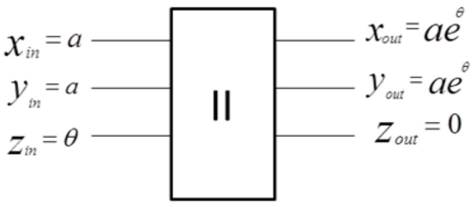


Fig.6 Hyperbolic rotation mode

The realization of CORDIC based DFT can be obtained by using hyperbolic rotation mode as shown in figure 6.

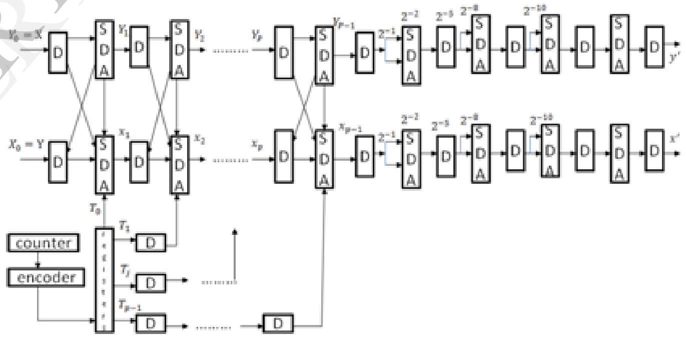


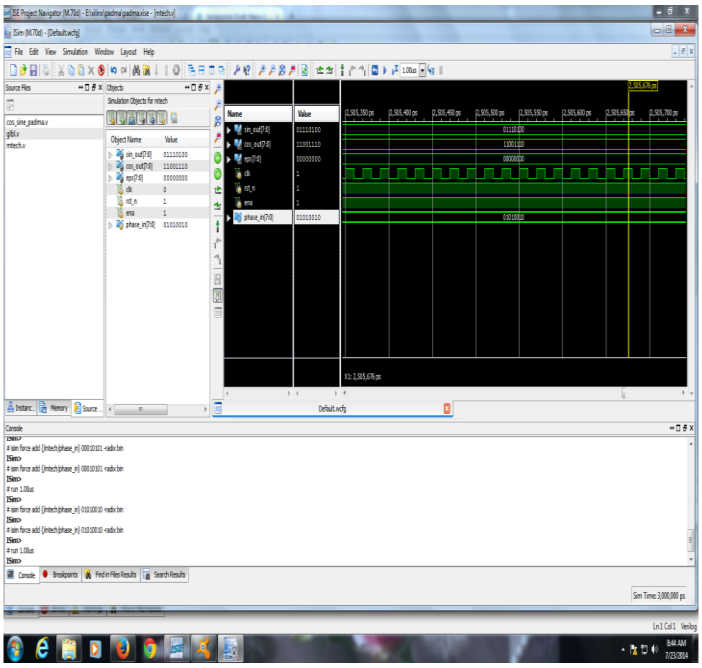
Fig.7. A Pipelined CORDIC structure

The pipelined CORDIC structure is shown in Figure 7, in which the leftmost part is controlled by $T_0, T_1, \dots, T_{p-i}$ performs the equivalent of four multiplications, one addition and one subtraction. But using the CORDIC algorithm, the results must be multiplied by a constant $\frac{1}{K_e} \approx 1.6467603$,

It is approximated as

$$\frac{1}{K_e} \approx \frac{1}{2}(1+\frac{1}{4})(1-\frac{1}{32})\,(1+\frac{1}{256})(1-\frac{1}{1024})$$

We can see from the above analysis that the DFT/IDFT can be calculated using CORDIC computations and additions/subtractions. This was actually our main motivation for the derivation of CORDIC based method. It is quite efficient for a high speed long-point DFT/IDFT.


Object Name	Value
sin_out[7:0]	01110100
cos_out[7:0]	11001110
eps[7:0]	00000010
clk	0
rst_n	1
ena	1
phase_in[7:0]	01010010

```
ISim>
# isim force add {/mtech/phase_in} 00010101 -radix bin
ISim>
# isim force add {/mtech/phase_in} 00010101 -radix bin
ISim>
# run 1.00us
ISim>
# isim force add {/mtech/phase_in} 01010010 -radix bin
ISim>
# isim force add {/mtech/phase_in} 01010010 -radix bin
ISim>
# run 1.00us
ISim>
```

Fig.9 Simulation results

Initially the DFT was decomposed in terms of COS and SINE terms by using Euler's formula, then for the computation of these trigonometric components we used pipelined CORDIC processor. For hardware implementation, we written Verilog code and compiled by using ModelSim software. Further simulated and synthesized by using Xilinx ISE design suite version 12.0 and implemented on Spartan 6.0 FPGA. Finally synthesis report and delay report are noted down. From the results it is observed that, the total real time taken for execution is 1.00secs, the total CPU time taken for execution is 0.94 secs. The macro statistics of CORDIC requires only single ROM, one 4x8-bit ROM, 20-adders and subtractions, three 8-bit adders, one 8-bit subtraction, 32 registers, eight 2-bit registers, 24 8-bit registers and 2 – multiplexers.

V. CONCLUSIONS

By exploiting some trigonometric identities in the DFT computation, the number of multiplications needed for DFT/IDFT has been reduced compared to the traditional FFT. One way to realize the new algorithm in hardware, which is more efficient for a long-point DFT, is to use a pipelined CORDIC algorithm, which simultaneously completes four multiplications. In the pipelined CORDIC structure, the data rate of the DFT/IDFT depends only on one carry look-ahead adder as the critical component in the pipeline. At present integration technology, it is possible to perform each pipeline step is as minimum as possible.

REFERENCES

[1] FENG ZHOU,"A New Fast Discrete Fourier Transform" ,Dept. of Information and Electronic Engineering, Zhejiang University, Hangzhov,P.R.China. PETER KORNERUP,"Dept. of Mathematics and Computer Science, Odense University, 2002,Odense, Denmark

[2] H. L. Groginsky et al., "A Pipeline Fast Fourier Transform", IEEE Trans. on Computers, vol. C-19, No. 11, Nov. 1970, pp. 1015-1019.

[3] J. E. Volder, "The Cordic Trigonometric Computing Technique", IEEE Trans. on Elect. Computers, vol. EC-B, Sep. 1959, pp. 330-334.

[4] Y.H. Hu$_5$ "CORDIC-Based VLSI Architecture for Digital Signal Processing," IEEE Signal Processing Magazine, pp.ló-July 1992

[5] R. Sarmineto, F. Tobajas, et.al, "A CORDIC Processor for FFTComputations and Its Implementation Using Gallium Arsenide Technology," IEEE Trans, on VLSI Systems, pp.18-30, vol.6, no.l, March 1998

[6] C. Ying, S.Chen and J. Chih, "Efficient CORDIC Designs for Multi-Mode OFDM FFT," ICASSP Page(s): 1036 -1039, vol.3, May 2006

[7] B . Heyne, J. Gotze, "A Pure CORDIC based FFT for Reconfigurable Digital Signal Processing, " 12th European Signal Processing Conference (EUSIPCO2004), Vienna, Austria, 2004

[8] B . Heyne, J. Gotze, "CORDIC-Based algorithms for software defined Radio (SDR) baseband Processing, " Adv. Radio Sci. , 4, 179-184, 2006

[9]Kulkarni et al., "Reconfigurable Vector-FFT/IFFT, Vector-Multiplier/Divider, " Patent US 7,082,45 1 B2 HEYNE B ET AL: "A Pure Cordic Based FFT for Reconfigurable Digital Signal Processing" PROCEEDINGS OF THE EUROPEAN SIGNAL PROCESSING CONFERENCE, 6 September 2004 (2004-09-06), pages 1513-1516,

[10] KATSUTOSHI SEKI ET AL: "A Cordic-Based Reconfigurable Systolic Array Processor for MIMO-OFDM Wireless Communications" SIGNAL PROCESSING SYSTEMS, 2007 IEEE WORKSHOP ON, IEEE, PI, 1 October 2007 (2007-10-01), pages 639-644, XP031164039 ISBN: 978-1-4244-1221-1

[11] Ayan Banerjee Anindya Sundar dhar, "FPGA realization of a CORDIC based FFT processor",Micro Processor and Micro system, vol. 25, Issue 3, May. 2001, pp. 131-142.

[12]T.Tran,B.Liu,"Fixed-point fast Fourier transform error analysis", IEEE Trans. on ASSP, 1976, vol.24(6), pp. 563–573.

[13]W.-H. Chang and T. Nguyen, "On the Fixed-Point Accuracy Analysis of FFT Algorithms," IEEE Transactions On Signal Processing, Vol. 56, No. 10, pp. 4673-4682, October 2008.

[14], A.M. Despain, "Fourier Transform Computers Using Cordic Iterations", IEEE Trans. on Computers, vol. C-23, No. 10, Oct. 1974, pp. 993-1001.

[15] . Karthick S, Priya P, valarmathy S, "CORDIC based FFT for Signal processing systems", ISSN 2278-8875, IJAREEIE, Vol.1, Issue 6 Dec 2012.

[16].Choudary P, Karmakar A, " CORDIC based Implementation of FFT", ICCCT, Sept 2011.

[17]. Ajay S P, S.s. Belsare, " Radix – 4 FFT Architecture " IJARCSSE, Vol.4, Issue 5, May 2014.

[18]. Thanuja, Sarada, " CORDIC based 16-point FFT processor", IJRIT, Vol 1 Issue 1, APR 2014.

[19].Dhar A s, Swapna Banerjee, " FPGA realization of CORDIC based FFT processor for biomedical signal processing", Dept ECE,IIT Kharagpur, India.