Computation of Twiddle Factor using CORDIC Algorithm for FFT Processor

Sureshwari R. Bichwe
Asst. Professor, Dept. of Electronics & Communication,
Kavikulguru Institute of Technology & Science, Ramtek
srbichwe@gmail.com

Dipti Bichwe
Asst. Professor, Dept. of Electronics & Telecommunication,
Guru Nanak Institute of Engineering & Technology, Nagpur
dipti.bichwe@gmail.com

Abstract— In the last decade, CORDIC (**CO**-ordinate **R**otation **DI**gital **C**omputer) algorithm has drawn wide attention from academia and industry for various applications such as DSP, biomedical signal processing, software defined radio, neural networks, and MIMO systems etc. CORDIC is employed for the Twiddle factor multiplication in the FFT. The unrolled CORDIC structure is used, since the iterative approach consumes routing resources heavily. The basic unrolled CORDIC structure is modified for use in FFT, which results in the reduction of ROM size drastically. In this paper, a simple method for determining the depth of the CORDIC is explained for the design of a DIF (Decimation in Frequency) FFT processor. The design is simulated using Xilinx ISE 8.1i software implemented on Spartan-III FPGA kit.

Keywords- *CORDIC, Twiddle factor, FFT*

*****

I. Introduction

Many of the algorithms used in DSP and matrix arithmetic require elementary functions such as trigonometric, inverse trigonometric, logarithm, exponential, multiplication, and division functions. The commonly used software solutions for the digital implementation of these functions are table lookup method and polynomial expansions, requiring number of multiplication and additions/subtractions [1]. However, digit by- digit methods exist for the evaluation of these elementary functions, which compute faster than software solutions. These are iterative pseudo division and pseudo multiplication processes, which resemble repeated-addition multiplication and repeated-subtraction division. In 1959, Volder has proposed a special purpose digital computing unit known as COordinate Rotation DIgital Computer (CORDIC) [2]. This algorithm was initially developed for trigonometric functions which were expressed in terms of basic plane rotations. CORDIC or Coordinate Rotation Digital Computer is a simple and hardware-efficient algorithm for the implementation of various elementary, especially trigonometric, functions. Instead of using Calculus based methods such as polynomial or rational functional approximation, it uses simple shift, add, subtract and table look-up operations to achieve this objective. Implementing FFT using CORDIC algorithm reduces the number of computations during processing & increases the accuracy of reconstruction of the signal.

II. The CORDIC Algorithm

The CORDIC algorithm proposed by Jack E Volder is usually implemented in either Rotation mode or Vectoring mode. In either mode, the algorithm is rotation of an angle vector by a definite angle but in variable directions. This fixed rotation in variable direction is implemented through an iterative sequence of addition/subtraction followed by bit-shift operation. The final result is obtained by appropriately scaling the result obtained after successive iterations. Owing to its simplicity the CORDIC algorithm can be easily implemented on a VLSI system. This algorithm calculates the sine and cosine values of a given angle (in radians) by transforming the co-ordinates from polar representation to representation in Cartesian form. To measure the values of sine and cosine, the coordinates corresponding to the angle on a unit circle is found, the x-coordinate gives the cosine values while the y-coordinate gives the sine value. The conventional method of implementation of 2D vector rotation shown in Figure 1

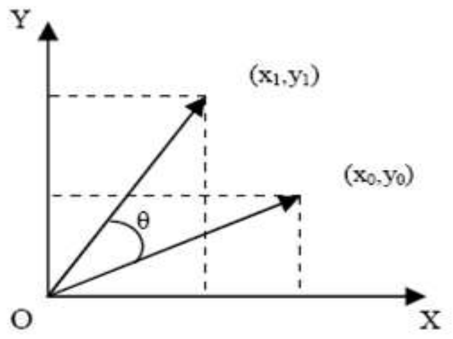

Fig 1: CORDIC Algorithm Schematic [1]

From the fig.1 it can be shown that when the vector with co-ordinates (x_0, y_0) is rotated by an angle θ, the new co-ordinates (x_1, y_1) will be

$$x_1 = x_0 \cos\theta - y_0 \sin\theta \quad \text{--- (1)}$$
$$y_1 = y_0 \cos\theta + x_0 \sin\theta \quad \text{--- (2)}$$

The hardware realization of these equations requires four multiplications, two additions/subtractions and accessing the table stored in memory for trigonometric coefficients. The CORDIC algorithm computes 2D rotation using iterative equations employing shift and add operations.

In the constant iteration, a certain vector converges to the required objective. In the most general form, one CORDIC iteration can be written as

$x(i+1) = x(i) - d(i)\, y(i)\, 2^{-i}$
$y(i+1) = y(i) - d(i)\, x(i)\, 2^{-i}$
$z(i+1) = z(i) - d(i)\, \mathrm{const}(i)$
$\mathrm{const}(i) = \arctan(2^{-i})$

Where $[x_i, y_i]$ is the input vector $[x_{i+1}, y_{i+1}]$ is the output vector which is the result of rotating $[x_i, y_i]$ by an angle θ. Z_i specifies the angle accumulator vector.

It can be easily shown mathematically that rotating a vector (xi, yi) by an angle θ in the anticlockwise (clockwise) direction is equivalent to multiplying a complex no. by $e^{j\theta}$ ($e^{-j\theta}$). This property of the CORDIC algorithm is utilized to simplify the computations in a FFT processor [3].

III. The Fast Fourier Transform

In many applications frequency analysis is necessary and desirable. The applications ranging from radar to spread spectrum communication employ the FFT for spectral analysis and frequency domain processing [4].

The FFT operates by decomposing an N – point time domain signal into N time domain signals. Then calculate the N frequency spectra of these N time domain signals and these N spectra are synthesized into a single frequency spectrum. The DFT of length N is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad 0 \leq k \leq N-1 \qquad \text{--- (3)}$$

where $\mathbf{W_N = e^{-j2\pi/N}}$ is known as **Twiddle factor** and denotes the Nth primitive root of unity, with its exponent evaluated modulo N[5]. Several architectures have been proposed for FFT implementations. FFT architectures can be classified broadly based on the Decimation in Time (DIT) or Decimation in Frequency (DIF) [2].The DIF class of algorithms has been found to introduce less computational noise for fixed point hardware implementations [7]. The DIF type of decomposition is used for the design of FFT in this paper.

The derivation of the DIF algorithm is straight forward and is found by calculating the odd and even output frequency samples separately, where as the DIT algorithm is based on subdividing the input data into its odd and even components. From the stand point of general hardware, the DIT algorithms are optimum for real input data sequences and DIF is appropriate when the input is complex [8]. The flow graph of the basic butterfly structure of DIF algorithm is shown in fig 2.

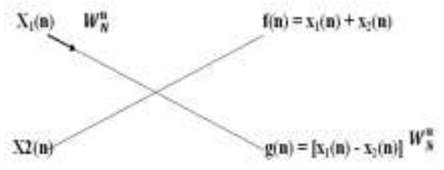

Fig.2 Butterfly diagram of DIF algorithm

The radix-2 structure of the FFT is a combination of an addition, a subtraction and a complex multiplication. The input data is multiplied by a complex cosine/sine pair. The result is a data pair of the same magnitude but rotated by the angle of cosine/sine pair. The CORDIC algorithm fulfils the requirements of the radix because it is one of its basic functionalities to rotate a real and imaginary pair of numbers at any angle [6]. The CORDIC computation is to decompose the angle of the twiddle factor into the weighted sum of a set of pre defined elementary location angles. The main idea of the CORDIC is to realize vector rotation through the iterative.

It is possible to implement an iterative architecture using a single CORDIC element and updating the registers in every clock pulse. This requires a barrel shifter which consumes routing sources so heavily that the advantage of iterative architecture is lost. It is preferable to use a loop unrolled architecture that uses a separate CORDIC element for each iteration. The number of CORDIC elements depends upon the quantization error for the application.

The basic structure of unrolled CORDIC is shown in Fig.3 which consists of simple shift registers & add/subtract blocks.

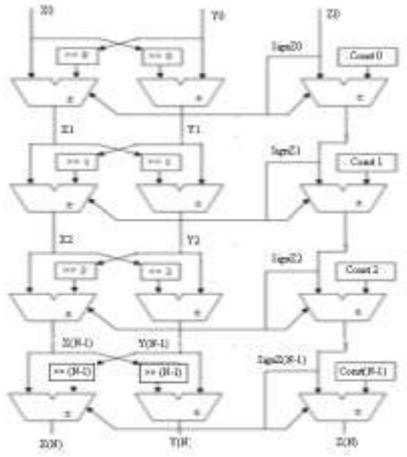

Fig.3 Basic structure of unrolled CORDIC

IV. Proposed CORDIC based FFT

The butterfly of fig.2 is modified using the CORDIC for multiplication with the twiddle factor as shown in fig.4 below

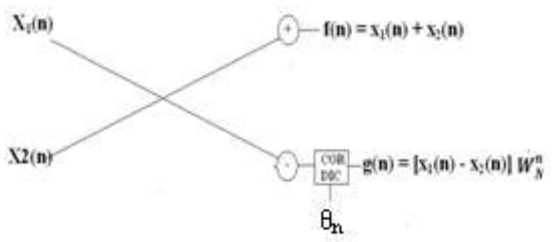

Fig.4 Radix-2 DIF with CORDIC

The butterfly operation requires a complex addition, a complex subtraction and a complex multiplication. The complex multiplication can be done with *n+s* CORDIC iterations and scaling iterations. Two complex addition and subtraction will need just one more clock cycle [9]. The design of 8-point FFT involves the repetition of this basic butterfly structure. An N point FFT requires N/2 twiddle factor computations, which can be pre calculated and stored in the ROM for FFT computation. By using the CORDIC structure, only the twiddle factor angle values need to be supplied which reduces the ROM requirement drastically.

The basic CORDIC structure can be further modified as shown in fig 5.The angle vector values can be pre calculated and stored in the ROM, instead of the actual angle values. The modified structure will reduce the amount of hardware required to compare the input angle with the constant values supplied, thus also increasing the speed of computation.

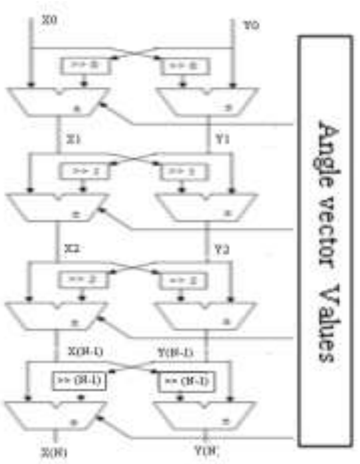

Fig.5 Modified structure of unrolled CORDIC

V. Determining the Depth of the CORDIC Structure

During each iteration of the CORDIC algorithm, the input angle is compared with the constant value given by
Const (i) = arc tan (2^{-i}), where i is an integer value. Table 1 shows the values of const (i) for different values of i. For better understanding, the values of θi are multiplied by 10 and converted to nearest integer values.

It is very clear from the table that const (i) equals zero after 12 iterations. If more accuracy is needed, then we can consider a depth of 12 for the unrolled CORDIC structure. Fairly good results can be obtained by considering a depth of 7, since after that the value of θi is less than 0.5, and we know that tan θ is approximately equal to θ for very small values of θ.

Table 1

i	θ=arctan2^{-i}	θ x 10	Approx. Integer val.
0	45	450	450
1	26.565	265.65	265
2	14.036	140.36	140
3	7.125	71.25	71
4	3.576	35.76	35
5	1.79	17.9	18
6	0.895	8.95	9
7	0.448	4.48	4
8	0.224	2.24	2
9	0.111	1.11	1
10	0.056	0.56	1
11	0.028	0.28	0
12	0.014	0.14	0

VI. Synthesis & Simulation Results

The proposed architecture can be modeled in hardware descriptive language VHDL. The hardware to determine the angle vector was modeled in VHDL and simulated in Xilinx ISE. The angle vectors from 0^0 to 45^0 in steps of 5^0 were determined using the hardware designed and verified with the actual values. The simulation results are shown in fig.6.

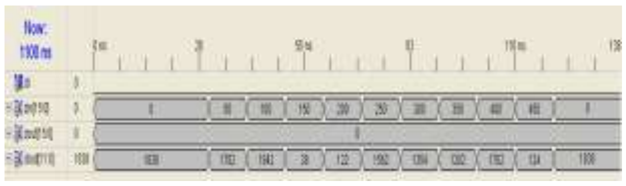

Fig 6 : Simulation Waveforms of Angle Vector

Table 2 shows the percentage saving on various resources when modified CORDIC structure is compared to the basic CORDIC structure.

Table 2

	Basic CORDIC	Modified CORDIC	% saving
Advanced HDL Synthesis Report			
Macro Statistics			
# Adders/Subtractors	:36	: 24	29.4
16-bit addsub	:36	: 24	29.4
Low Level Synthesis			
Design Statistics			
# IOs	:80	: 76	5
Cell Usage :			
# BELS	:1640	: 1140	30.49
# INV	:1	: 12	-91.66
# LUT1	:14	nil	100
# LUT2	:182	nil	100
# LUT3	:352	:384	9.1
# MUXCY	:525	:360	31.43
# XORCY	:542	:384	29.15
# IO Buffers	:80	:76	5
# IBUF	:48	:44	8.33
# OBUF	:32	:32	0
Device utilization summary: Selected Device : 3s50pq208-5			
Number of Slices:	: 280 out of 768 36%	: 192 out of 768 25%	31.42

Number of 4 input LUTs:	: 548 out of 1536 35%	: 384 out of 1536 25%	29.93
Number of bonded IOBs:	: 80 out of 124 64%	: 76 out of 124 61%	5
Timing Summary:			
Speed Grade:	: -5	: -5	
Maximum combinational path delay:	: 59.074ns	: 40.203ns	31.94
CPU	: 6.28 / 6.55 s	: 5.47 / 5.89 s	12.89 / 10.07
Elapsed	7.00 / 7.00 s	: 5.00 / 6.00 s	28.57 / 14.29

VII. Conclusion & Future Work

In this paper the basic CORDIC algorithm was modified to reduce the number of computations when used in the FFT processor. A very simple method of determining the depth of the CORDIC is proposed. The hardware simulation results of angle vectors have been shown.

The CORDIC algorithm is limited to the range $-\pi/4$ to $+\pi/4$. By adding a little hardware and using the trigonometric inequalities, the project can be extended for four quadrant calculations.

References

[1] B Lakshmi and A.S. Dhar, “Review Article CORDIC Architectures: A survey”, Hindawi publication corp., VLSI Design, vol. 2010, article ID 794891, 19 pages.

[2] J. E. Volder, “The CORDIC Trigonometric Computing Technique ”, IRE Trans. Electronic Computers, Vol. EC – 8, No. 3, pp. 330 – 34, September 1959

[3] Shakeel S. Abdula, Haewoon Nam, Mark Mc Dermol and Jacob A. Abraham, “A High Throughput FFT Processor with No Multipliers”, IEEE Transactions, 2009, pp. 485 – 490.

[4] K. Padmanabhan, S. Ananthi, R. Vijayarajeswaran, “A Practical Approach to Digital Signal Processing”, pp. 105 – 150.

[5] Jiang Murong, Yang Jun, Guo Yuedong, Du Xio gang, Li Na, “Combining CORDIC Algorithm and FPGA to Design Duel Core FFT Processor”, Proceedings ICIMA 2009, pp. 68 – 71.

[6] Stephan W. Mond Wurf, “Benefits of the CORDIC Algorithm in a versatile COFDM Modulator / Demodulator Design”, Fourth IEEE International Caracas Conference on Devices and Systems, Aruba, pp. TO26 – 1 to 6.

[7] Z. Wang, M. Dong and Y. Zhao, “Design and Implementation of Efficient FFT Processor for Multicarrier System”, proceedings IEEE Canadian Conference on Electrical and Computer Engineering, May 2005, pp. 1384 – 1387.

[8] Richard G. Lyons, “Understanding Digital Signal Processing”, second edition, Pearson Education, Chapter 4, pp. 125-150.

[9] Yu Hen Hu, “VLSI Architectures for Digital Signal Processing”, IEEE Signal Processing Magazine, July 1992, pp. 16-35.