International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878, Impact Factor (2014): 3.05

Pipelined Implementation of CORDIC and 64-Point FFT with Memory Interfacing Module

K. Naga Chaitanya[1], Dr. P. Trinatha Rao[2]

[1]M.Tech VLSI, Department of ECE, GITAM University Hyderabad GITAM University, Hyderabad, India
[2]Associate.Professor, Department of ECE, GITAM University Hyderabad GITAM University, Hyderabad, India

Abstract: ***Signal processing plays an important role in the field of communication Engineering. The amount of data that is received is processed by using different algorithms. Discrete Fourier Transform (DFT) is one of the technique that is used to compute the data. Direct computation of DFT results in complexity; as a result Fast Fourier Transform (FFT) is one of the algorithms to reduce the complexity. The proposed paper presents the parallel-pipelined implementation of radix-2 fixed point 32-point FFT algorithm using state machine as controller and fixed point pipelined implementation of Linear CORDIC that operates for the angles*** $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$***. The result of 64-point FFT are obtained in M=*** $log_2 N$ ***clock cycles due to multi processor technique. Initially FFT functionality is checked using MATLAB and finally simulated and synthesized using Xilinx ISE 14.1.Besides CORDIC algorithm is implemented in both MATLAB and in Xilinx on Virtex-7. The main objective of this work is to obtain an area efficient FFT and CORDIC without performance loss that could be used as a part of Signal processing.***

Keywords: Signal processing, DFT, FFT, CORDIC, state machine controller

1. Introduction

A signal in communication systems is referred as a function that conveys information about the behavior or attributes about information contained in the signals, signal processing is done on the signals. As the signals produced from physical quantities are analog in nature, these signals are converted in to discrete by sampling technique before processing. Therefore Discrete Time Signal processing is for sampled signals, defined only at discrete points in time and as such are quantized in time, but not in magnitude. In Digital Signal Processing (DSP) working on frequency domain is advantageous when compared to time domain [1]. This is overcome by Discrete Fourier Transform (DFT) that converts a signal in discrete time domain to discrete frequency domain. Computation of DFT is performing various mathematical operations like addition, multiplication etc on the received data. Computation directly by using DFT algorithm is quite tedious and complex which influences in global computation cost of design that consumes N^2 additions and $N(N-1)$ multiplications. Cooley and Turkey developed the well-known radix-2 FFT algorithm to reduce the computational load of the DFT [2].

The paper is organized into five sections. In section-III we discuss the brief view of FFT, in section-IV about CORDIC algorithm and finally section-V gives the opted methodology and in VI Simulation results of CORDIC and FFT.

2. Literature Review

Brett W. Dickson and Albert A. Conti proposed a Pipelined Fast Fourier Transform (FFT) architectures, which are efficient for long instances (32k points and greater), are critical for modern digital communication and Radar systems. For long instances, Single-Path Delay-Feedback (SDF) FFT architectures minimize required memory, which can dominate circuit area and power dissipation. Their paper presents a parallel Radix-22 SDF architecture capable of significantly increased pipelined throughput at no cost to required memory or operating frequency. A corresponding parallel coefficient generator is also presented. Resource utilization results and analysis are presented targeted for a 45nm silicon-on-insulator (SOI) application-specific integrated circuit (ASIC) process. Multipliers implemented using Vedic mathematics is superior in terms of area efficiency. Carry Select Adders (CSLA) are one of the fastest adders used in several processors to perform fast and complex arithmetic functions. In the proposed paper, the Vedic multiplier which is developed using the Urdhva Tiryakbayam Sutra along with modified carry select adder is used to perform a Radix-22pipeline Fast Fourier Transform. The Fast Fourier Transform (FFT) which is implemented using Vedic multiplier and adder is then compared with FFT implemented using traditional multipliers and adders and its performance is verified.

3. Fast Fourier Transform Algorithm

Fast Fourier Transform (FFT) is a commonly used technique for the computation of Discrete Fourier Transform (DFT) [3]. DFT computations are required in the fields like filtering, spectral analysis, video processing etc. to calculate the frequency spectrum or to identify a system's frequency response from its impulse response and vice versa. Based on how one divides a set of N inputs into two sets of N/2 numbers, there are two types of radix-2 FFT algorithm or Cooley-Turkey algorithm. They are [4]

i Decimation in time FFT (DIT-FFT)
ii Decimation in frequency FFT (DIF-FFT).

In the proposed paper implementation of FFT algorithm is done using DIT-FFT.

The N-point discrete Fourier transform is defined by

$$X[n] = \sum_{n=1}^{N-1} x(n)\, W^{nk} \quad (1)$$

$$X[n] = \sum_{n=1}^{\frac{N}{2}-1} x(2n)\, W^{2nk} + \sum_{n=1}^{\frac{N}{2}-1} x(2n+1)\, W^{(2n+1)k} \quad (2)$$

Therefore

$$X[n] = \sum_{n=1}^{\frac{N}{2}-1} x(2n)\, W_{\frac{N}{2}}^{nk} + W_N^{nk} \sum_{n=1}^{\frac{N}{2}-1} x(2n+1)\, W_{\frac{N}{2}}^{nk} \quad (3)$$

$$X[n] = G(n) + W_N^{nk} H(n) \quad (4)$$

This equations state that the input sequence is divided into consecutive sets of even and odd samples. In order to obtain the sets the input samples digit values are bit reversed and the resultant set is applied as input to FFT block. To calculate the inverse transform, the real and imaginary part of the input and output are swapped. From the above equations, W^{nk} is twiddle factor term. This twiddle factor term can be realized as butterfly operation which is the important building block of FFT algorithm [3]. The difference between Decimation In Time (DIT) and Decimation in Frequency (DIF) lies in the position of the twiddle factor multiplication, which is either performed before or after the subtraction and addition [4]. The technique use for FFT is based on divide and conquer, it will be most efficient if the input sequence is of length N = r^p, where N is called point, r is called radix, and p is a positive integer. An N-point FFT can be computed by using p stages of which each stage having N/2 butterflies [5].

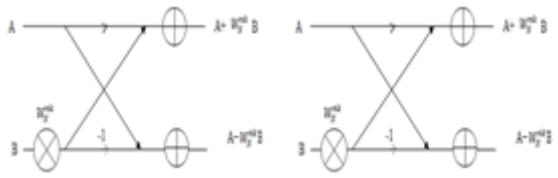

Figure 1: A radix-2 DIT and DIF Butterfly architecture

Twiddle factor generator is a key component in IFFT/FFT computation. There exist many popular generation techniques for twiddle factor; Coordinate Rotation DIgital Computer (CORDIC) algorithm, pipelined CORDIC algorithm, polynomial-based approach, ROM-based scheme, and the recursive function generators. For small lengths such as 64-point to 512 point, ROM-based is a better choice[6].Therefore in the proposed paper the implementation of pipelined CORDIC along with the synchronous parallel FFT is done and discussed in the next section.

4. CORDIC Algorithm

The key concept of CORDIC arithmetic is based on the simple and ancient principles of two-dimensional geometry. But the iterative formulation of a computational algorithm for its implementation was first described in 1959 by Jack E. Volder for the computation of trigonometric functions, multiplication and division [6]. CORDIC based computing received increased attentionin1971, when John Walther showed that, by varying a few simple parameters, it could be used as a single algorithm for unified implementation of a wide range of elementary transcendental functions involving logarithms, exponentials, and square roots along with those suggested by Volder [7]. CORDIC is attractive due to the simplicity of its hardware implementation, since the same iterative algorithm could be used for all the above mathematical applications using the basic shift-add operations of the form $x \pm y2^{-i}$. The conventional method of implementation of 2D vector shown in the Figure.1 using Givens rotation transform is represented by the equations [9].

$$x_{out} = x_{in} \cos\theta - y_{in} \sin\theta, \quad (5)$$

$$y_{out} = x_{in} \sin\theta + y_{in} \cos\theta. \quad (6)$$

where (x_{in}, y_{in}) and (x_{out}, y_{out}) are the initial and final coordinates of the vector respectively. The hardware realization of these equations requires four multiplications, two additions/subtractions and accessing the table stored in the memory for trigonometric coefficients. The CORDIC rotator performs 2D rotation using a series of specific incremental rotation angles selected so that each is performed by a shift and add operation iteratively.

The three basic equations of CORDIC algorithm are:

$$x_{i+1} = x_i - m\sigma_i y_i \rho^{-S_{m,i}} \quad (7)$$

$$y_{i+1} = y_i + \sigma_i x_i \rho^{-S_{m,i}} \quad (8)$$

$$z_{i+1} = z_i - \sigma_i \alpha_{m,i} \quad (9)$$

Based on the value of **m** the algorithm can operate in one of three configurations: Linear ($m = 0$), Circular($m = 1$) and Hyperbolic($m = -1$). Within each of these configurations the algorithm functions in one of two modes – rotation or vectoring.$\boldsymbol{\sigma_i}$ represents either clockwise or counter clockwise direction of rotation, **ρ** represents the radix of the number system and the shift sequence $S_{m,i}$ depends on the coordinate system and the radix of number system. $S_{m,i}$ affects the convergence of the algorithm . In rotation mode, the input vector is rotated by a specified angle, while in vectoring mode the algorithm rotates the input vector to the **x**-axis while recording the angle of rotation is required. The value of α_i also changes according to the configuration. Depending on the mode of operation z and y are the steering variables in rotation and vectoring mode respectively. The length of the vector increases if required micro rotations are not perfect, so in order to maintain a constant vector length, the obtained results have to be scaled by the scale factor K and it is given by the equation.

$$K = \prod_i k_i \quad (10)$$

The below flow graph represents the flow of CORDIC for computation of values:

Figure 2: Flow Chart of CORDIC Algorithm

Although no popular architectures are known to us for fully parallel implementation of CORDIC, different forms of pipelined implementation of CORDIC have however been proposed for improving the computational throughput [10].To handle latency bottlenecks, various architectures are present. Most of the well-known architectures could be grouped under bit parallel iterative CORDIC, bit serial iterative CORDIC and pipelined CORDIC architecture. Since the CORDIC iterations are identical, it is very much convenient to map them into pipelined architectures. The main emphasis in efficient pipelined implementation lies with the minimization of the critical path.

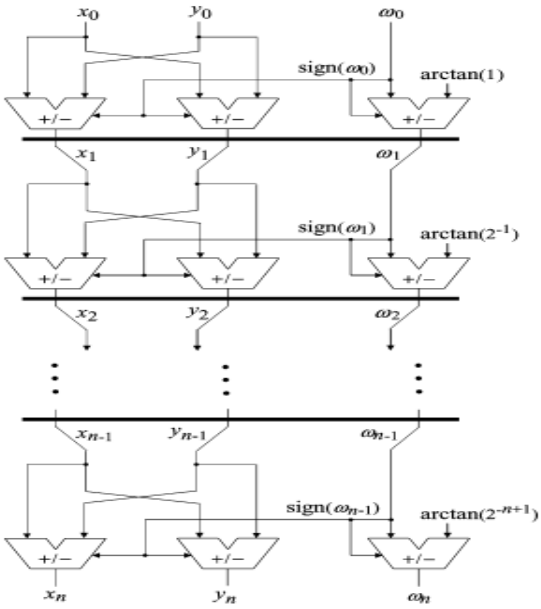

Figure 3: Basic structure of a pipelined CORDIC unit

5. Implementation of 64-Point FFT and CORDIC

In the proposed paper as the results of 64 point FFT can't be represented exactly. So the design methodology that is used for 64-point FFT is explained by using 16-point FFT. The implementation of FFT is done using DIT-FFT algorithm. The implementation of FFT consists of bit reversing module, state machine design that acts as a controller for the total design, butter fly module and finally shifter module design.

A. Architecture of FFT: Initially in order to realize the hardware module of the FFT parallel iterative architecture is chosen as the latency for the parallel iterative architecture is less when compared to the other architectures. The values computed are stored iteratively into the intermediate registers that increase the throughput.

B. Angle Storage of FFT:
Twiddle factor is as cosine value of an angle. These angles are in decimal point form. So as the angles can't be represented directly. In Xilinx the conversion of the values need to be done for the ease of computation. The values can be represented in either fixed-point or floating point format. Representing and operating in Floating point format is tedious that consumes a lot of hardware. Therefore Fixed-point format is opted for the storage of values. The scaling of values is done by multiplying with 2^8 and the resultant is stored in ROM, and at the time of computation the angles are accessed based on the address.

C. Design of FFT controller:
The entire FFT module operations are controlled by using State machine as a controller. The FSM operates in 4 states that are named as S_0,S_1,S_2 and S_3 Each stage of the FSM performs an operation.

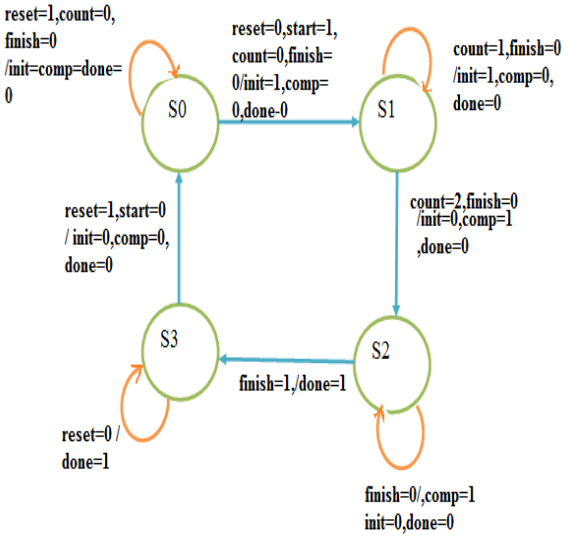

Figure 4: State machine controller of FFT Algorithm

D. Design Consideration of CORDIC in Xilinx:

In design of CORDIC element using fixed point arithmetic the accuracy and latency of the result mainly depends on the iteration count and its implementation. To achieve n-bit accuracy, the word length of x and y data path should be $(n + 2 + log_2 n)$ bit width and for the computation of the angle θ the bit width should be $(n + log_2 n)$.The design of CORDIC is made by considering 22 bit width of x and y along with overflow 20 bit width for angle representation.

6. VHDL Simulation Results of CORDIC and FFT

The below table states the values of CORDIC obtained in Xilinx and MATLAB. On comparing both the tabulated values there is 0.01% error in one of the coordinates which is negligible .So maximum amount of accuracy is obtained through this opted methodology.

Table 1: Tabulated values of CORDIC in VHDL

Iteration Number	Value of X-coordinate	Value of Y-coordinate	Value of Angle
Iteration 1	0	262144	-61440
Iteration 2	131072	262144	47104
Iteration 3	65536	294912	-10388
Iteration 4	102400	286720	18496
Iteration 5	84480	293120	4147
Iteration 6	75320	295760	-3184
Iteration 7	79941	294584	483
Iteration 8	77640	295208	-1351
Iteration 9	78793	294905	-435
Iteration 10	79368	294752	24
Iteration 11	79081	294829	205
Iteration 12	79224	294791	-123
Iteration 13	79295	294772	-82
Iteration 14	79330	294763	-57
Iteration 15	79347	294759	-44
Final scaled value	0.184	0.682	

Table 2: MATLAB values of 16 iterations of CORDIC

Iteration Number	Value of X-coordinate	Value of Y-coordinate	Value of Angle
Iteration 1	0	1.000	-0.2618
Iteration 2	0.5	1.000	0.2018
Iteration 3	0.2500	1.125	-0.0431
Iteration 4	0.3906	1.0938	0.0812
Iteration 5	0.3223	1.1182	0.0188
Iteration 6	0.2873	1.1282	-0.0124
Iteration 7	0.3050	1.1237	0.0032
Iteration 8	0.2962	1.1261	-0.0046
Iteration 9	0.3006	1.1250	-0.007
Iteration 10	0.3028	1.1244	0.0012
Iteration 11	0.3017	1.1247	0.0003
Iteration 12	0.3011	1.1248	-0.002
Iteration 13	0.3014	1.1248	0.000
Iteration 14	0.3014	1.1248	0.000
Iteration 15	0.3014	1.1248	-0.001

Table 3: Hardware utilization of CORDIC Algorithm

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	71	93120	0%
Number of Slice LUTs	351	46560	0%
Number of Fully Used LUT-FF pairs	71	351	20%
Number of Bounded IOBS	132	240	55%
Number of BUFGCTRLS	1	32	3%

Table 4: Hardware utilization of 16-point FFT Algorithm

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	942	2443200	0%
Number of Slice LUTs	2795	1221600	0%
Number of Fully Used LUT-FF pairs	729	3008	24%
Number of Bounded IOBS	962	1200	80%
Number of BUFGCTRLS	1	128	0%
Number of DSP48E1s	19	2160	0%

Figure 5: VHDL simulation result for 16 point FFT Algorithm

7. Conclusion

In the proposed paper the conventional implementation of CORDIC algorithm for computing the X and Y coordinates for a particular angle is done using pipelined architecture with maximum accuracy and less hardware. Along with the CORDIC 64-point FFT algorithm is also implemented with an accuracy lose of 0.4% and the final resultants are shown in the figure 5. The final excess bits of the FFT are truncated by using truncating module that reduces the bits without the loss of accuracy. Due to the multi processing modules the output of N point FFT is obtained in $log_2 N$ clock cycles that are shown in the above figure. The final values of FFT are stored in micro semi SRAM module.

References

[1] Design, Simulation, Implementation, and Performance Analysis of a fixed-point 8 Point FFT Core for Real Time Application in Verilog HDL, International Journal of Applied Research and Studies (IJARS) ISSN: 2278-9480 Vol 3, Issue 5 (May – 2014)
[2] Serin Sera Paul, Simy M Baby "An Efficient Design of Parallel Pipelined FFT Architecture"in IJECS Vol.3 Oct. 2014
[3] Neha V. Mahajan, Dr. J. S. Chitode "Simple Computation of DIT FFT" IJARCSSE, Vol 4, Issue 5, May 2014
[4] Sudha Kiran G , Brundavani P "FPGA Implementation of 256-Bit, 64-Point DIT-FFT Using Radix-4 Algorithm" Vol 3, Issue 9, September 2013
[5] Venkata Subbarao Gutta, S. Malarvizhi "FPGA Implementation of a CORDIC-based Radix-8 FFT Processor for Real-Time Harmonic Analyzer" IJCA (0975 – 8887) in National conference on VSLI and Embedded systems 2013
[6] J.Volder, "The CORDIC trigonometric computing technique", IEEE Transactions on Electronic

Computers, vol.EC-8, no. 8, pp 330-334, September 1959
[7] Naveen Kumar, Amandeep Singh Sappal " CORDIC Design and Architecture", (IJACSA) Vol. 2, No. 4, 2011.
[8] Pipelined Parallel FFT Architectures via Folding Transformation, Manohar Ayinala, Student Member, IEEE, Michael Brown, and Keshab K. Parhi, Fellow, IEEE transactions on very large scale integration (vlsi) systems, vol. 20, no. 6, June 2012.
[9] J M Rudagi, Srikant, Basavaraj B Patil, Dr S Subbaraman ,"Performance Analysis of Radix 4 CORDIC Processor in Rotation mode with Parallel Scale factor Computation",IJETAE Vol 2, Issue 7, July 2012.
[10] Amritakar Mandal* and Rajesh Mishra" Reconfigurable Design of Pipelined CORDIC Processor for Digital Sine-Cosine" Journal of Signal Processing Theory and ApplicationsOct.20 (2012).
[11] Pramod, K.Sridharan "50Yearsof CORDIC: Algorithms, Architectures, and Applications", IEEE transactions on circuits and systems I:regular papers, vol.56,no.9,september2009
[12] John F. Wakerly, Digital Design Principles and Practices, Fourth Edition, Pearson Education, Inc. 2006.
[13] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in Proc. 10th Int. Parallel Processing Symp., 1996, pp. 766–770.
[14] H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," IEEE Trans. Comput., vol. C-33, no. 5, pp. 414–426, May 1984.
[15] J.S. Walther, "A unified algorithm for elementary functions", in: Proceedings of Spring. Joint Computer Conference, 1971, pp. 379–385.
[16] J. W. Cooley and J. Tukey, "An algorithm for machine calculation of complex fourier series," Math. Comput., vol. 19, pp. 297–301, Apr. 1965.
[17] VHDL programming by J.Baskar