ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2018.0272

DSP IMPLEMENTATION OF THE FAST FOURIER TRANSFORM USING THE CORDIC ALGORITHM

Youness Mehdaoui[1] and Rachid El Alami[2]
[1]Department of Physics, Multidisciplinary Faculty, University Sultan Moulay Slimane, Morocco
[2]Department of Physics, Faculty of Sciences, Sidi Mohamed Ben Abdellah University, Morocco

Abstract
Fourier transform is a tool enabling the understanding and implementation of a large number of numerical methods for signal and image processing. This tool has many applications in domains such as vocal recognition, image quality improvement, digital transmission, the biomedical sector and astronomy. This paper proposes to focus on the design methodology and experimental implementation of Fast Fourier Transform (FFT). The interest of this work is an improvement which makes it possible to reduce the processing time of calculates the FFT while preserving the best performances by using the operator CORDIC and the fixed point, so this work is compared with the results found in the literatures.

Keywords
FFT, CORDIC, Fixed Point, DSP, Time of Processing

1. INTRODUCTION

Discrete Fourier Transformation (DFT) is a mathematical tool for processing the digital signal, which is the discrete equivalent of the continuous Fourier transform that is used for analog signal processing. The calculation of the DFT of the complex sequences in the time domain will convert these sequences into frequency domain and the inverse procedure is done by the Inverse Discrete Fourier Transform (IDFT) [1]. But only drawback is the computation time it requires.

The interest of an algorithm of Fast Fourier transform FFT makes it possible to reduce the number cycle of machine, therefore the time of computation. The FFT was one of the 10 major algorithms of the 20th century [19]. It is also one of the most used and in 1990 it was estimated [20] that on an installed base of Cray supercomputers of 200 machines (at 25 million US dollar each), 40% of CPU cycles are dedicated FFT calculations.

The COordinate Rotation Digital Computer (CORDIC) algorithm [2] - [4] was originally created by Volder [2]. The algorithm approximates most functions based on trigonometry. It performs rotations without using multiplication operations. Another advantage of this algorithm is that it makes it possible to obtain a precision determined in advance by performing a given number of iterations.

In the FFT, to calculate the twiddle, we will use the sine and the cosine, the algorithm CORDIC implemented with fixed point will allow speed of calculating sinus and cosines, all this will allow us to have a reduced time with a better precision.

In this work an implementation of the FFT on a (DSP Digital Signal Processor) c64x+ and compare the results with what was found in [5], to come out with a conclusion of the utility of the use of specialized circuits like the DSP.

This paper is organized as follows: In section 2 we will present the related works, in section 3, we introduce the Fast Fourier Transform, we will explain the algorithm CORDIC in section 4. The fixed point development is given in section 5. The methodology of the proposed implementation is presented in section 6 and 7. The results will be presented in section 8. We will end with a conclusion in section 9.

2. RELATED WORKS

In the work [5] an architecture is presented that allows the calculation of the Discrete Fourier Transform using the CORDIC algorithm. The twiddle factors i.e. the phase rotation factors where the calculations required in the DFT are calculated by the algorithm CORDIC. Moreover, using some trigonometric identities in DFT calculation CORDIC rotators are used effectively.

This algorithm is applied more widely in various applications, including the radar signal processors [23] and robotics, HP-35 calculator, math coprocessors 8087 [22].

It has been observed when the number of N-point samples increases, the time and hardware requirements of the system increase. A faster algorithm like FFT can solve this problem and dedicated circuits can be used as Digital Signal Processors (DSP).

3. FAST FOURIER TRANSFORM

The Fast Fourier Transform algorithm was initially discovered by Gauss in 1805, but was not successful until 1965 after the publication [21] of it by Cooley and Tukey. It is for this reason that the basic algorithm usually carries their names. The algorithm of the FFT described by Cooley and Tukey [21] in their article is then derived like this:

For an input sequence $x(n)$, the DFT of N points is defined as follows in the Eq.(1):

$$X(k)=\sum_{n=0}^{N-1} x(n)\cdot W_N^{nk} \tag{1}$$

where: k=0,1,…,N-1, n is the time index, the integer k is the frequency index and the complex number W_N^{nk}, which corresponds to the nth root of the unit, commonly called twiddle factor, is defined as follows in the Eq.(2):

$$W_N^{nk}=\exp\left(\frac{-2i\pi nk}{N}\right)=\cos\left(\frac{2\pi nk}{N}\right)-i\cdot\sin\left(\frac{2\pi nk}{N}\right) \tag{2}$$

Starting at the Eq.(3):

$$X(K)=\sum_{n=0}^{N-1}x(n)\cdot W_N^{nk} \tag{3}$$

The Eq.(2) is rewritten as,

$$X(K)=\sum_{r=0}^{\left(\frac{N}{2}\right)-1}x(2r)\cdot W_N^{2rk}+\sum_{r=0}^{\left(\frac{N}{2}\right)-1}x(2r+1)\cdot W_N^{(2r+1)k} \tag{4}$$

and

$$X(K)=\sum_{r=0}^{\left(\frac{N}{2}\right)-1}x(2r)\cdot\left(W_N^2\right)^{rk}+W_N^k\sum_{r=0}^{\left(\frac{N}{2}\right)-1}x(2r+1)\cdot\left(W_N^2\right)^{rk} \tag{5}$$

Returning to the definition of W_N:

$$W_N=e^{-j\frac{2\pi}{N}} \tag{6}$$

Note that:

$$W_N^2=e^{-j\frac{2\pi}{N}2}=e^{-j\frac{2\pi}{N/2}}=W_{N/2} \tag{7}$$

Finally, the samples of the transformed signal X (K) are found in the Eq.(8):

$$X(K)=\sum_{r=0}^{\left(\frac{N}{2}\right)-1}x(2r)\cdot W_{N/2}^{2rk}+W_N^k\sum_{r=0}^{\left(\frac{N}{2}\right)-1}x(2r+1)\cdot W_{N/2}^{2rk} \tag{8}$$

It is noticed that the algorithm of the FFT factorizes the DFT to reduce the number O(N^2) à O(N·log$N$)

4. CORDIC OPERATOR

4.1 CORDIC ALGORITHM

The CORDIC algorithm makes it possible to perform calculations such as vector rotations or Cartesian-Polar and Polar Cartesian coordinates changes in the plane Euclidean.

We can cite, for example, applications where the CORDIC algorithm is used: single-sideband modulation, discrete, direct and fast Fourier transform [6] [7], frequency filtering (Gray-Marke trellis, orthogonal filters) and wave filters [8]), adaptive modeling of non-stationary processes (optimal recursive filtering, Kalman filter [9]). He is also involved in the resolution of a large number of linear algebra problems like the orthogonal algorithms of Givens [10], Fadeeva, singular value decomposition [11], QR and Cholesky decomposition.

4.2 PRINCIPLE OF THE CORDIC ALGORITHM

The CORDIC algorithm is based on trigonometric function calculations; its principle is to perform rotations on a base vector for a given angle.

Suppose the rotation of the vector $V(x,y)$ by an angle φ as illustrated in Fig.1

The coordinates of the vector V' are expressed according to the Eq.(9):

$$\begin{aligned}x'&=x\cos\varphi-y\sin\varphi\\ y'&=y\cos\varphi+x\sin\varphi\end{aligned} \tag{9}$$

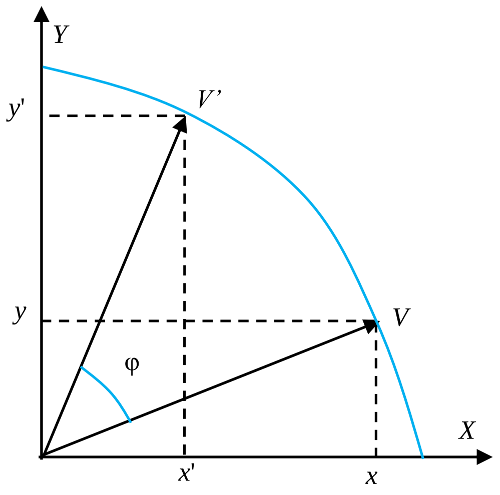

Fig.1. Rotation of the vector V in the Cartesian plane

If we restrict the angles of rotation $\tan^{-1}(2^{-i})$ where i = 0, 1, 2, 3,…, we then obtain φ by a series of successive elementary rotations of the order of:

$$\theta_{i+1}=\theta_i-d_i\cdot\tan^{-1}(2^{-i}) \tag{10}$$

where, $d_i=\pm1$.

The index d_i indicates the direction of rotation of the angle for each iteration, this index is determined at each iteration according to the result of a comparison.

Each iterative vector $V_{i+1}(x_{i+1}, y_{i+1})$ is represented by Eq.(11):

$$\begin{aligned}x_{i+1}&=K_i\left[x_i-y_id_i\left(2^{-i}\right)\right]\\ y_{i+1}&=K_i\left[y_i-x_id_i\left(2^{-i}\right)\right]\end{aligned} \tag{11}$$

where, $K_i=\cos(\tan^{-1}(2^{-i}))=(1+2^{-2i})^{0.5}$.

Since for a relatively high number of iterations, the product tends towards a constant result, it is possible for us to apply it later in the algorithm. In fact, for a given sequence of elementary rotations, the factors K_i can be grouped together and applied at one time. Thus we obtain a set of simplified and specific equation for calculating the mathematical operations sought:

$$\begin{aligned}x_{i+1}&=x_i-d_idy_i\\ y_{i+1}&=y_i-d_idx_i\end{aligned} \tag{12}$$

where:

$$dy_i=y_i\ 2^{-i} \tag{13}$$

$$dx_i=x_i\,2^{-i} \tag{14}$$

$$d\theta_i=\tan^{-1}(2^{-i}) \tag{15}$$

$$d_i=\pm1 \tag{16}$$

According to the sign of θ_i or y_i, we calculate:

$$x'=A_nx \tag{17}$$

where, the constant A_n depends only on the sequence of elementary rotations given by the Eq.(18):

$$A_n=\prod_{i=0}^{n}K_i \tag{18}$$

We notice that the main interest of this constant is that it does not depend on θ but only on the number of stages. For an increasing number of stages, this constant tends to the value equal to 0.607252935.

These equations are therefore used to calculate operations. The general principle of the CORDIC algorithm is to rotate the rotation vector in the appropriate direction by an increasingly smaller angle until the angle θ or the x and y values are approximately equal to 0.

Cosine and sine

In the paper [12], it is shown that the sine and cosine of the input angle can simultaneously be calculated as follows in Eq.(19) by CORDIC in rotation mode.

$$\begin{aligned} X_n &= A_n.X_0 \cos Z_0 \\ Y_n &= A_n.X_0 \sin Z_0 \end{aligned} \tag{19}$$

By defining $X_0 = 1/A_n$, the rotation produces a scaled sine and cosine of the angle Z_o.

5. FIXED-POINT DEVELOPMENT

The diagram below illustrates a typical development scenario in use today:

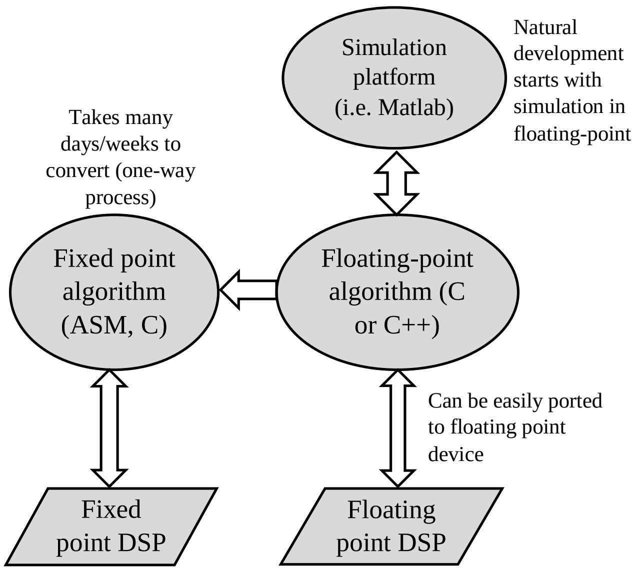
Simulation platform (i.e. Matlab)
Natural development starts with simulation in floating-point
Takes many days/weeks to convert (one-way process)
Fixed point algorithm (ASM, C)
Floating-point algorithm (C or C++)
Can be easily ported to floating point device
Fixed point DSP
Floating point DSP

Fig.2. The dilemma of fixed-point development

The design may initially start with a simulation (i.e. MatLab) of a control algorithm, which typically would be written in floating-point math (C or C++). Existing methodologies [13, 14] achieve a floating-to-fixed-point transformation leading to an ANSI-C code with integer data types. This algorithm can be easily ported to a floating-point device. However, because of the commercial reality of cost constraints, most likely a 16-bit or 32-bit fixed-point device would be used in many target systems.

The effort and skill involved in converting a floating-point algorithm to function using a 16-bit or 32-bit fixed-point device is quite significant. A great deal of time (many days or weeks) would be needed for reformatting, scaling and coding the problem. Additionally, the final implementation typically has little resemblance to the original algorithm [18, 15].

For digital signal processors (DSPs), the methodology aim is to define the optimized fixed point specification which minimizes the code execution time and leads to sufficient accuracy [16], some experiments [17] can represent up to 30% of the global implementation time.

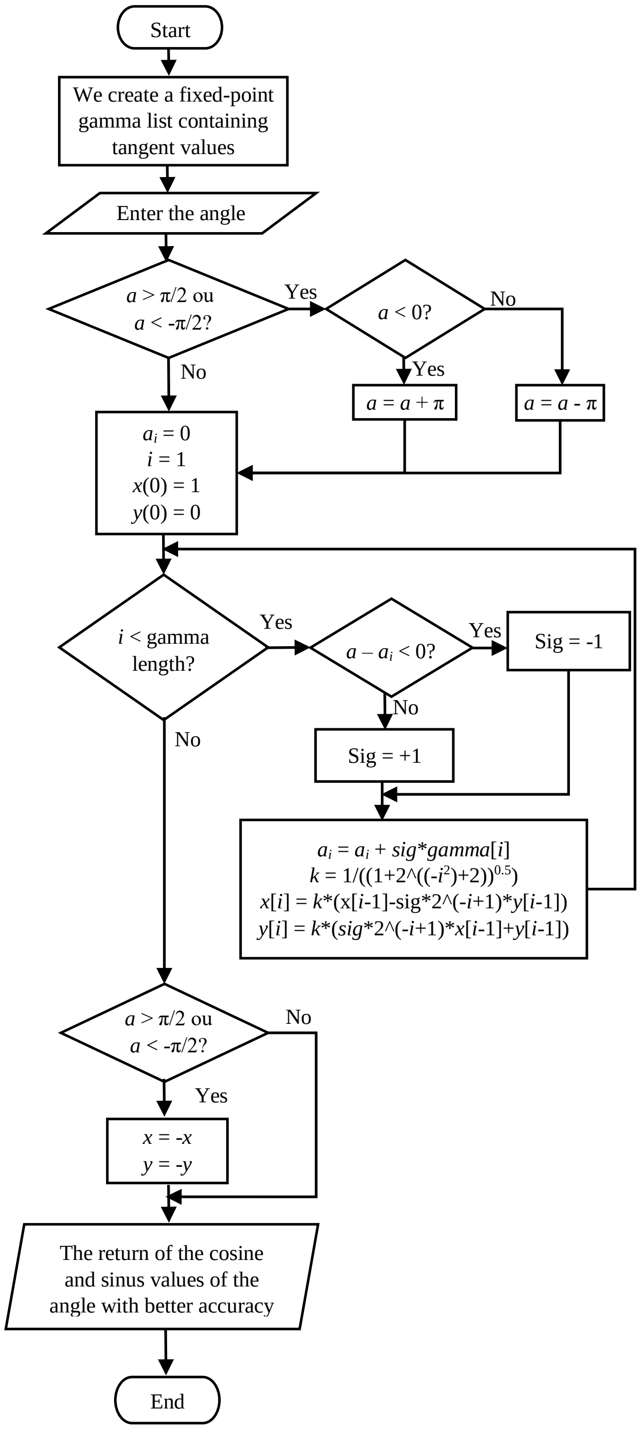
Start
We create a fixed-point gamma list containing tangent values
Enter the angle
a > π/2 ou a < -π/2?
Yes
a < 0?
No
Yes
No
a = a + π
a = a - π
a_i = 0
i = 1
x(0) = 1
y(0) = 0
i < gamma length?
Yes
a – a_i < 0?
Yes
Sig = -1
No
Sig = +1
No
a_i = a_i + sig*gamma[i]
k = 1/((1+2^((-i²)+2))^0.5)
x[i] = k*(x[i-1]-sig*2^(-i+1)*y[i-1])
y[i] = k*(sig*2^(-i+1)*x[i-1]+y[i-1])
a > π/2 ou a < -π/2?
No
Yes
x = -x
y = -y
The return of the cosine and sinus values of the angle with better accuracy
End

Fig.3. The proposed implementation by CORFAST

6. THE PROPOSED IMPLEMENTATION OF THE CORDIC ALGORITHM

The Fig.3 shows the proposed implementation of CORDIC which will be integrated in the FFT, we will name this proposition algorithm by CORFAST.

7. PROPOSED IMPLEMENTATION OF FFT

In this section the proposed FFT is presented, it is implemented on a fixed-point DSP and its performance will be evaluated and compared with the results in [5]. The flow diagram for FFT Computation is shown in Fig.4. In the butterfly calculation part of the flowchart, the custom CORFAST is used.

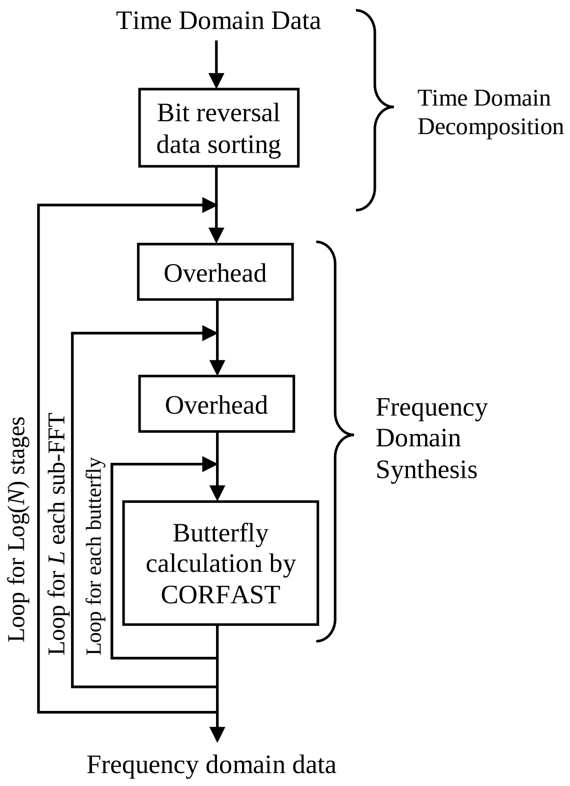

Fig.4. Diagram for FFT Computation using CORFAST

8. DSP IMPLEMENTATION

Using the Code Composer Studio software to do the simulations, this software uses the internal hardware of the DSP C64x+ very efficiently. The algorithms are implemented using DSP processor C64x+ and tested for different input data lengths. The following results are obtained for 10, 12 and 20 point FFT length; the clock cycle is equal to 1GHz (1ns). We will compare the results found with the results in the paper [5]. The Table.1 summarizes the results.

Table.1. Cycle's number, time taken and their ratios

Number of input sequences	**FFT (Proposed work)**		**DFT [5]**		**Ratio**
	Benchmark (cycles)	**Time taken (in ns)**	**Benchmark (cycles)**	**Time taken (in ns)**	
10	17058	17058	24179	483580	28.35
12	25093	25093	28999	579980	23.11
20	37176	37176	96509	1930180	51.92

It is noted that our implementation on the DSP C64x+ gives better results by comparison with the results found in [5], it is because of the use of the fixed point and the algorithm CORDIC which allows us to minimize the processing time of the data by keeping the best performances.

9. CONCLUSIONS

In this work we implemented a FFT using a DSP which is specialized in this kind of application, we can conclude that our implementation is faster (with a ratio of 28.35, 23.11, 51.92 for a number of sequences 10, 12, 20 respectively) compared work [5]. The results found leads us to conclude that the use of specialized circuits like the DSP will give better results than the use of circuits like FPGA (Field-Programmable Gate Array) which makes the implementation very expensive at the time level, architecture complexity.

REFERENCES

[1] J.G. Proakis and D.G. Manolakis, "*Digital Signal Processing, Principles, Algorithms and Applications*", 4th Edition, Prentice Hall, 2006.

[2] J.E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Transactions on Electronic Computers*, Vol. 8, No. 3, pp. 330-334, 1959.

[3] R. Andraka, "A Survey of CORDIC Algorithms for FPGA based Computers", *Proceedings of 6th International Symposium on FPGAs*, pp. 191-200, 1998.

[4] B. Parhami, "*Computer Arithmetic*", Oxford University Press, 2010.

[5] Debaprasad De, K. Gaurav Kumar, Archisman Ghosh and Anurup Saha, "FPGA Implementation of Discrete Fourier Transform using CORDIC Algorithm", *AMSE Journals*, Vol. 60, No. 2, pp. 332-337, 2017.

[6] A.M. Despain, "Very Fast Fourier Transform Algorithms Hardware for Implementation", *IEEE Transactions on Computers*, Vol. 28, No. 5, pp. 333-341, 1979.

[7] A.M. Despain, "Fourier Transform Computers using CORDIC Iterations", *IEEE Transactions on Computers*, Vol. 23, No. 10, pp. 993-1001, 1974.

[8] S.K. Rao, "Orthogonal Digital Filters for VLSI Implementation", *IEEE Transactions on Circuits and Systems*, Vol. 31, No. 11, pp. 771-778, 1984.

[9] Tze-Yun Sung et al., "VLSI Implementation of Real-Time Kalman Filter", *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2223-2226, 1986.

[10] H.M. Ahmed, J.M. Delosme and M. Morf, "Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing", *Computer*, Vol. 15, No. 1, pp. 65-82, 1982.

[11] Joseph R. Cavallaro and Franklin T. Luk, "CORDIC Arithmetic for an SVD Processor", *Proceedings of IEEE 8th Symposium on Computer Arithmetic*, pp. 271-290, 1988.

[12] Ray Andraka, "A Survey of CORDIC Algorithms for FPGA based Computers", *Proceedings of ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays*, pp. 191-200, 1998.

[13] K.I. Kum, J. Kang and W. Sung, "AUTOSCALER for C: An Optimizing Floating-Point to Integer C Program Converter for Fixed-Point Digital Signal Processors", *IEEE*

Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 47, No. 9, pp. 840–848, 2000.

[14] M. Willems, V. Bursgens and H. Meyr, “FRIDGE: Floating Point Programming of Fixed-Point Digital Signal Processors”, *Proceedings of 8th International Conference on Signal Processing Applications and Technology*, pp. 23-29, 1997.

[15] Texas Instrument, “C28x IQmath Library”, Available at: http://www.ti.com/lit/sw/sprc990/sprc990.pdf.

[16] D. Menard, D. Chillet and O. Sentieys, “Floating-to-Fixed-Point Conversion for Digital Signal Processors”, *EURASIP Journal on Applied Signal Processing*, Vol. 2006, pp. 1-19, 2006.

[17] T. Grotker, E. Multhaup and O. Mauss,” Evaluation of HW/SW Tradeoffs using Behavioral Synthesis”, *Proceedings of 7th International Conference on Signal Processing Applications and Technology*, pp. 781-785, 1996.

[18] M Mehdaoui and M. Mrabti. “A Faster MC-CDMA system using a DSP Implementation of the FFT”, *Proceedings of 5th International Symposium On I/V Communications and Mobile Network*, pp. 662-668, 2010.

[19] B.A. Cipra, “The Best of the 20th Century: Editors Name Top 10 Algorithms”, *SIAM News*, Vol. 33, No. 4, pp. 1-2, 2000.

[20] J.R. Johnson and R.W. Johnson, “Challenges of Computing the Fast Fourier Transform”, Available at: https://pdfs.semanticscholar.org/e452/3079aa489b27f8438 562166ad92a928ba83f.pdf.

[21] J.W. Cooley and J.W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series”, *Mathematics of Computation*, Vol. 19, No. 90, pp. 297-301, 1965.

[22] J. Duprat and J.M. Muller, “The CORDIC Algorithm: New Results for Fast VLSI Implementation”, *IEEE Transactions on Computers*, Vol. 42, No. 2, pp. 168-178, 1993.

[23] R.J. Andraka, “Building a High Performance Bit Serial Processor in an FPGA”, Available at: http://fpga-guru.com/files/supercn.pdf.