EFFICIENT MEMORYLESS CORDIC FOR FFT COMPUTATION

M. Garrido and J.Grajal

Departamento de Seales, Sistemas y Radiocomunicaciones
ETSIT, Universidad Politécnica de Madrid
Ciudad Universitaria s/n, 28040, Madrid, Spain
mgarrido@gmr.ssr.upm.es

ABSTRACT

A new memoryless CORDIC algorithm for the FFT computation is proposed in this paper. This approach calculates the direction of the micro-rotations from the control counter of the FFT, so the area of the rotator hardly depends on the number of rotations, which is particularly suitable for the computation of FFTs of a high number of points. Moreover, the new CORDIC presents other advantages such as the simplification of the basic CORDIC processor used to calculate the micro-rotations, or an easy way to compensate the intrinsic gain of the CORDIC algorithm. Additionally, the VLSI implementation of the algorithm is a pipeline architecture with high performance in terms of speed, throughput and latency.

Index Terms— Memoryless systems, Pipeline processing, Discrete Fourier transforms, Very-large-scale integration.

1. INTRODUCTION

Nowadays, signal processing systems are characterized by a high complexity and real time operation. To satisfy these requirements of real time operation, FPGA-based hardware acceleration systems provide a reasonable speedup with very low cost. The enormous integration density of nowadays technologies has allowed current FPGAs to hold a complete system in a single chip, the so-called System on Chip (SoC) approach. In this context, any improvement in speed or area of a particular implementation drastically improves the performance and scope of the whole signal processing system.

The FFT is a key element in most signal processing applications. When analyzing its structure, we can see that it presents important demands of memory, both for the rotations and samples. Given that the multiplication carried out to perform rotations in the FFT requires very large area, most architectures use the CORDIC algorithm [1] instead of conventional multipliers.

In this work we propose an efficient architecture for the CORDIC algorithm that allows very large FFT implementations without a significant increase in area. Moreover, the proposed architecture improves the performance of previous approaches, what results in a very efficient implementation. The main contributions of our implementation are the following:

- The rotator requires no memory, so the area hardly depends on the number of rotations which permits the calculation of FFTs of a high number of points.
- The circuit fulfills real time requirements due to a high speed, a low latency and a throughput of one sample per clock cycle.
- The basic CORDIC processor, which computes the micro-rotations, has been simplified based on a novel modification of the rotation sequence.
- The intrinsic scale of the conventional CORDIC has been reduced and an easy compensation is proposed.
- The implementation is highly parametrizable, which makes it adaptable to any application.

Important work has been carried out on the implementation of the FFT rotators but, to the best of our knowledge, no previous work presents a memoryless scheme like the one we propose here. In some cases the memory size has been reduced, but the reduction is only to log_2N [2], or $0.5N$ [3], where N is the number of points of the FFT. Additionally, other important aspects of the CORDIC rotators that can be found in the literature are the design of the basic CORDIC processor [2, 4, 5], the scale factor of the CORDIC [2, 5], the computation of the rotation vector [6], or the performance in terms of speed [4, 7] and latency [6]. All these points have also been addressed in this work.

2. CORDIC ALGORITM

The CORDIC algorithm [1] decomposes the desired rotation angle, θ, into a sum of a set of M predefined angles, α_i, with $i = 0, \dots, M-1$:

$$\theta = \sum_{i=0}^{M-1} \alpha_i + \epsilon \qquad (1)$$

where ϵ is the error of the approximation, and,

$$\alpha_i = \pm tg^{-1}(2^{-i}) \tag{2}$$

Thus, the rotation is accomplished iteratively according to:

$$\begin{aligned} x_{i+1} &= x_i - y_i\delta_i 2^{-i} \\ y_{i+1} &= y_i + x_i\delta_i 2^{-i} \end{aligned} \tag{3}$$

where δ_i indicates the direction of the so called micro-rotation.

In a conventional CORDIC, $\delta_i \in \{-1, 1\}$. It means that all the rotations are accomplished, what leads to a constant gain of the rotator. This intrinsic gain can be compensated by multiplying the output by:

$$K = \prod_{i=0}^{M-1} cos(\alpha_i) = \prod_{i=0}^{M-1} cos(tg^{-1}(2^{-i})) \tag{4}$$

3. MEMORYLESS CORDIC ARCHITECTURE

Figure 1 shows the architecture of the designed CORDIC. The main difference between this circuit and other CORDIC rotators is the fact that no memory is required. Thus, the rotation sequence is obtained from the counter used to control the whole N-point FFT, which counts from 0 to $N-1$. Firstly, the rotation angle, θ, is calculated and then the corresponding rotation sequence is generated. As occurs in a conventional CORDIC, $\delta_i \in \{-1, 1\}$, leading to a constant gain. Finally, the sequence is adapted to the module that rotates the signal in order to simplify the design.

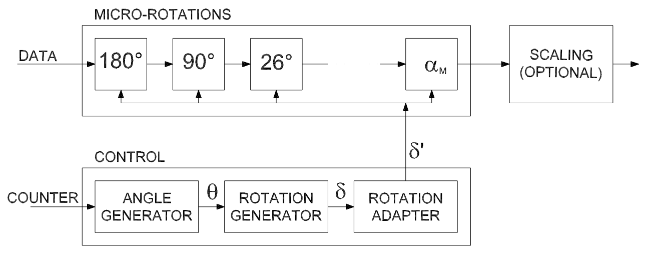


Fig. 1. CORDIC Rotator.

The micro-rotations are calculated by means of a pipeline, where each stage rotates the input data a certain angle, according to the CORDIC algorithm. The number of stages can be selected by the user depending on the desired precision. Besides, it must be noted that the rotation of $45°$ is not considered. In a conventional CORDIC rotator the $45°$ rotation places the remainder angle in the range $[-45°, 45°]$. By contrast, in our design this is achieved by the $180°$ and $90°$ rotations, which introduces no error and reduces the intrinsic gain of the CORDIC. On the other hand, due to the pipeline structure of the micro-rotations, the circuit works at a rate of 1 sample per clock cycle.

Finally, the output data can be scaled in order to compensate the intrinsic gain of the CORDIC rotator.

3.1. Angle generator

According to the Cooley-Tukey algorithm [8], an N-point FFT can be divided into $n = log_r N$ stages where r is the radix of the FFT. We will assume that both N and r are a power of 2. In this context, the angle sequence at the stage $s \in \{1 \dots n\}$ has a length $L = r^s$ and is repeated N/r^s times in order to perform the N required rotations.

The angle sequence, depends on the chosen radix. Thus, the following sequence must be generated for radix 2:

$$a_s = \underbrace{0,0,0,0,\dots}_{2^{s-1}}, \underbrace{0,1,2,3,\dots}_{2^{s-1}} \tag{5}$$

and, for radix 4,

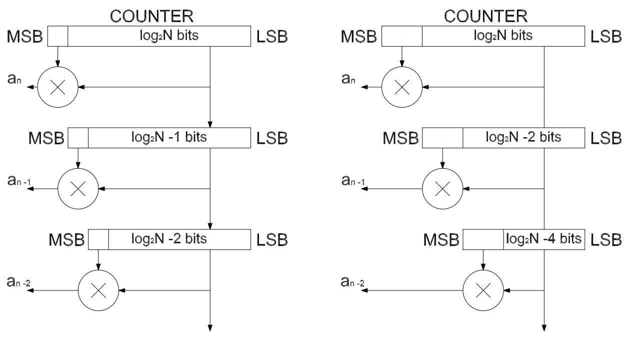
$$a_s = \underbrace{0,0,0,0,\dots}_{4^{s-1}}, \underbrace{0,1,2,3,\dots}_{4^{s-1}}, \underbrace{0,2,4,6,\dots}_{4^{s-1}}, \underbrace{0,3,6,9,\dots}_{4^{s-1}} \tag{6}$$

Generalizing for any value of radix, the angle sequence will be generated by concatenating r subsequences:

$$0, p, 2p, \dots, (r^{s-1} - 1)p \tag{7}$$

where $p = 0, 1, 2, \dots, r-1$.

Figure 2 shows how these sequences can be obtained from the 0 to $N-1$ counter of an N-point FFT. According to (7), p is the value of the $log_2 r$ most significant bits of the counter at each stage, and the rest of the bits counts from 0 to $(r^{s-1}-1)$.



(a) Radix 2 (b) Radix 4

Fig. 2. Angle generation

In order to normalize all the sequences of the stages of the FFT to the minimum rotation angle, $\theta_{min}(rad) = 2\pi/N$ the values of the sequences must be multiplied by $q = N/r^s$ or, equivalently, shifted $log_2(q)$ bits:

$$m_s = q \cdot a_s = \underbrace{0,0,0,0,\dots}_{2^{s-1}}, \underbrace{0,q,2q,3q,\dots}_{2^{s-1}} \tag{8}$$

The sequence m_s represents the rotation angle if the circumference is divided into N identical angles, i.e., if $\theta \in [0, \ldots, N-1]$ is one of the angles of the sequence,

$$\theta(rad) = -\frac{2\pi}{N}\theta \qquad (9)$$

The explained method is used to generate the angles of both DIT (*Decimation In Time*) and DIF (*Decimation In Frequency*) decompositions, considering that $s = 1$ is the input stage in the DIT case and the output stage in a DIF FFT.

3.2. Generation of the rotation vector

Given the rotation angle, θ, the rotation generator calculates the vector δ that controls the micro-rotations. The procedure is similar to the one described in [6].

Firstly, the circuit checks the three more significant bits of the angle to determine if the $180°$ and $90°$ must be fulfilled. This is possible because the angle ranges from 0 to $N-1$ and N is a power of 2, so these bits divide the circumference in 8 sectors.

On the other hand, the generation of the CORDIC terms of the rotation vector is based on the idea that $\phi \approx tg(\phi)$ when $\phi \to 0$, so:

$$\alpha_i \approx tg(\alpha_i) \equiv 2^{-i} = 2 \cdot 2^{-i-1} \equiv 2 \cdot tg(\alpha_{i+1}) \approx 2 \cdot \alpha_{i+1} \qquad (10)$$

Table 1 shows that this approximation is not valid for low values of i. However, for higher values of i, it can be considered that $\alpha_i/\alpha_{i+1} \approx 2$.

i	$\alpha_i\ (deg)$	α_i/α_{i+1}
0	45.00000000	1.69395495
1	26.56505118	1.89260405
2	14.03624347	1.96999457
3	7.12501635	1.99226795
4	3.57633437	1.99805195
5	1.78991061	1.99951204
6	0.89517371	1.99987795
7	0.44761417	1.99996948
8	0.22381050	1.99999237
9	0.11190568	1.99999809
10	0.05595289	1.99999952

Table 1. Relation between CORDIC angles.

According to this, the rotation generator operates such as a conventional CORDIC in order to calculate the first rotations, i.e., it checks if the remainder angle is positive or negative to decide the direction of the rotation and then adds or subtracts α_i from this angle.

Next, the rest of the rotations are calculated considering the described approximation. With this purpose, the remainder angle is normalized by the minimum rotation angle, α_M, that corresponds to the last micro-rotation. Consequently, the result offers the rest of the δ_i values.

Logically, there is a compromise between the precision of the calculation and the hardware resources. Thus, in order to adequate the design to a certain application, the number of micro-rotation stages, the bits used in the calculations of the rotation vector and the selection of the first angle where the approximation is considered are parameters of the circuit.

3.3. Micro-rotations and rotation adapter

The $180°$ and $90°$ rotations previous to the CORDIC stages are easily calculated by interchanging the vector components and/or changing their sign. On the other hand, the CORDIC rotations are accomplished by the following circuit:

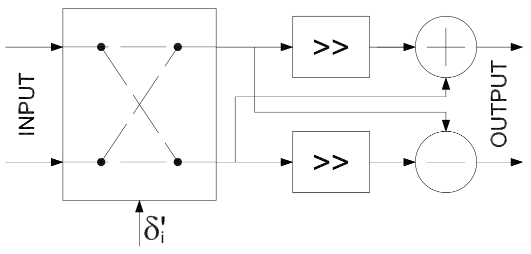


Fig. 3. Basic CORDIC processor.

This is a simplified version of other basic CORDIC processors [5], which usually requires multiplexers and two add/subtractor components. By contrast, this novel implementation only needs a switch, an adder and a subtractor, taking into account that the shifts are hard wired. Thus, both the area and the latency of each CORDIC stage are reduced.

The design is based on the idea that each micro-rotation always accomplishes an addition and a subtraction. According to this, the switch decides which of the shifted component must be added and which subtracted.

However, it may change the output of the vector components and, therefore, the input components to the following stage may also arrive at any of the inputs. This is why the rotation adapter is needed. Thus, it calculates δ_i' from the δ_i values by means of simple combinational logic.

3.4. Scaling

Scaling is one of the critical problems of some CORDIC rotator designs, mainly in redundant and unidirectional architectures, where $\delta = 0$ is a possible value. According to this, some micro-rotations may be skipped and, thus, the scale factor is not constant. On the contrary, in a conventional CORDIC the compensation factor is constant and known *a priori*:

$$K = \prod_{i=0}^{M} cos(tg^{-1}(2^{-i})) \approx 0.6073 \qquad (11)$$

assuming that $cos(tg^{-1}(2^{-i})) \approx 1, \forall i > M$, which can be considered a good appoximation for $M \geq 7$. However, in the proposed design the $45°$ rotation has been removed and the compensations factor is:

$$K = \prod_{i=1}^{M} cos(tg^{-1}(2^{-i})) \approx 0.8588 \qquad (12)$$

Consequently, the scale factor, K^{-1}, has been reduced from 1.6468 to 1.1644, which improves the performance of the algorithm. In many applications the compensation of the scale factor is not necessary. However, a good compensation of the CORDIC gain can be achieved just by using 2 adders, considering that [5]:

$$K = 0.8588 \approx 0.8594 = 1 - 2^{-3} - 2^{-6} \qquad (13)$$

4. EXPERIMENTAL RESULTS

The described algorithm has been programmed in VHDL. The results discussed in this section has been obtained for the case of a Virtex-II xc2v4000-6 FPGA.

MAXIMUM OPERATING FREQUENCY (MHz)							
CORDIC	INPUT BITS OF THE SAMPLES						
STAGES	8	10	12	14	16	18	20
8	307	299	292	285	278	272	266
16	278	272	266	260	255	249	244

Table 2. Maximum operating frequency.

Firstly, table 2 shows the speed for some of the possible configurations of the system. These results are better than the ones obtained in recent works that uses FPGAs: up to 222 MHz in [4] and 176 MHz in [7].

These speed results are possible due to the fact that the hardware implementation of the basic CORDIC processors includes registers after the switch and at the output. Even so, the latency of the circuit is low. In fact, the input counter of the system is in advance, so the rotation sequence is already calculated when the input samples are received. Consequently, in a continuous data flow the latency only depends on the micro-rotations stages an can be calculated as $l = 2 \cdot M + 5$ clock cycles.

Figure 4 compares the area of a conventional and the designed CORDIC rotators for 16 input bits and 16 micro-rotation stages, depending on the length of the angle sequence, L.

As the conventional CORDIC stores the rotation vectors in a memory, the area of this design strongly depends on L. On the other hand, the area of the memoryless architecture hardly varies when L increases.

According to this, when few rotations are accomplished, the memory of the conventional rotator takes up less area than the control module of the memoryless one. However, for high values of L, the memoryless CORDIC obtains much better performance. Therefore, the designed rotator is much more adequate than other CORDIC rotators for the computation of FFTs of a large number of points.

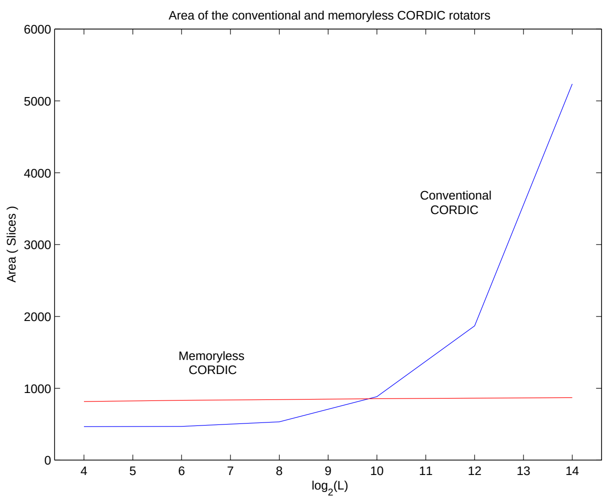


Fig. 4. Area of the conventional and memoryless CORDIC rotators vs. the length of the angle sequence.

5. CONCLUSIONS

Our novel CORDIC algorithm calculates the rotations without any memory. Additionally, an optimized design leads to a high performance in terms of operating frequency and latency. The improvement of these figures of merit opens new possibilities for the calculation of real time long FFTs.

6. REFERENCES

[1] J.E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computers*, vol. EC-8, pp. 330–334, Sep. 1959.

[2] Cheng-Ying Yu, Sau-Gee Chen, and Jen-Chuan Chih, "Efficient CORDIC Designs for Multi-Mode OFDM FFT," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 1036–1039, May 2006.

[3] Yun-Nan Chang and Keshab K. Parhi, "An efficient pipelined FFT architecture," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, pp. 322–325, Jun 2003.

[4] F. Angarita, A. Perez-Pascual, T. Sansaloni, and J. Vails, "Efficient FPGA Implementation of CORDIC Algorithm for Circular and Linear Coordinates," *International Conference on Field Programmable Logic and Applications*, pp. 535–538, Aug 2005.

[5] Y. H. Hu, "CORDIC-based VLSI Architectures for Digital Signal Processing," *IEEE Signal Processing Magazine*, vol. 9, Juj 1992.

[6] D. Timmermann, H. Hahn, and B.J. Hosticka, "Low Latency Time CORDIC Algorithms," *Computers, IEEE Transactions on*, vol. 41, pp. 1010–1015, Aug 1992.

[7] T. Zaidi, Q. Chaudry, and S. A Khan, "An Area and Time Efficient Collapsed Modified CORDIC DDFS Architecture For High Rate Digital Deceivers," *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, pp. 677–681, Dec 2004.

[8] J. W. Cooley and J.W Tukey, "An algorithm for machine computation of complex fourier series," *Math. Computation*, vol. 19, pp. 297–301, Apr. 1965.