CORDIC Algorithm Implementation in FPGA for Computation of Sine & Cosine Signals

Hunny Pahuja, Lavish Kansal, Paramdeep Singh

Abstract: Many hardware algorithms exist to handle the hardware intensive signal processing problems. Among these algorithms is a set of shift-add algorithms collectively known as CORDIC for computing a wide range of functions including certain trigonometric, hyperbolic, linear and logarithmic functions. While there are numerous articles covering various aspects of CORDIC algorithms, very few survey more than one or two, and even fewer concentrate on implementation in FPGAs. The work presented here associated with FPGA implementation of CORDIC systems for fast and silicon area efficient computation of the sine and cosine functions. The results of theoretical investigation into redundant CORDIC have been presented here. Summary of CORDIC synthesis results based on Actel and XILINX FPGAs is given. Finally applications of CORDIC sine and cosine generators in small satellites are discussed.

Keywords: CORDIC, sine, cosine, FPGA, synthesis, redundant signed-digit system.

1. Introduction

The name CORDIC is an acronym for Coordinate Rotation Digital Computer. In 1959 Jack E. Volder described the Coordinate Rotation Digital Computer or CORDIC for the calculation of trigonometric functions, multiplication, division and conversion between binary and mixed radix number systems. The CORDIC-algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shift and add.

2. Approaches to CORDIC Hardware Implementation

The CORDIC algorithm can be implemented in hardware using three approaches: a sequential approach - the structure is unfolded in time, a parallel approach – the structure is unfolded in space or a combination of the two. These three approaches and the resulting structures are also referred to in the literature as iterative, cascaded and cascaded fusion, respectively. A sequential CORDIC design performs one iteration per clock cycle and consists of three n-bit adders/subtractors, two sign extending shifters, a look-up table (LUT) for the step angle constants and a finite state machine. A parallel CORDIC design is similar to an array multiplier structure consisting of rows of adders/ subtractors. A combined CORDIC design is based on a sequential structure where the logic for several successive iterations is cascaded and is executed within one clock cycle. Since algebraic addition is the main operation in the CORDIC algorithm, the efficiency of the hardware implementation of the algorithm depends significantly on the type of adder used. Adders based on the conventional two-digit binary system have time delay dependent on the bit length n and in the best case of fast hierarchical adder structures the time delay for execution of one iteration is of logarithmic order . Bit-serial and binary adders have been used in sequential CORDIC implementations and all types of adders have been tried in cascaded CORDIC designs – bit-serial adders, carry-save adders, binary adders, redundant adders, combinations of both binary and redundant adders. Obviously, a combination of sequential approach and bit-serial adders will result in the slowest design with minimal area, parallel approach and redundant adders – in the fastest design with maximal area.

- *Hunny Pahuja is currently Assistant Professor in Department of Electronics & Electrical Engineering, Lovely Professional University, Jalandhar, Punjab, India, Email: - pahuja.hunny77@gmail.com.*
- *Lavish Kansal is currently Assistant Professor in Department of Electronics & Electrical Engineering, Lovely Professional University, Jalandhar, Punjab, India, Email: - lavish.s690@gmail.com.*
- *Paramdeep Singh is currently Assistant Professor in Department of Electronics & Electrical Engineering, Lovely Professional University, Jalandhar, Punjab, India, Email: - paramdeepsingh1987@gmail.com.*

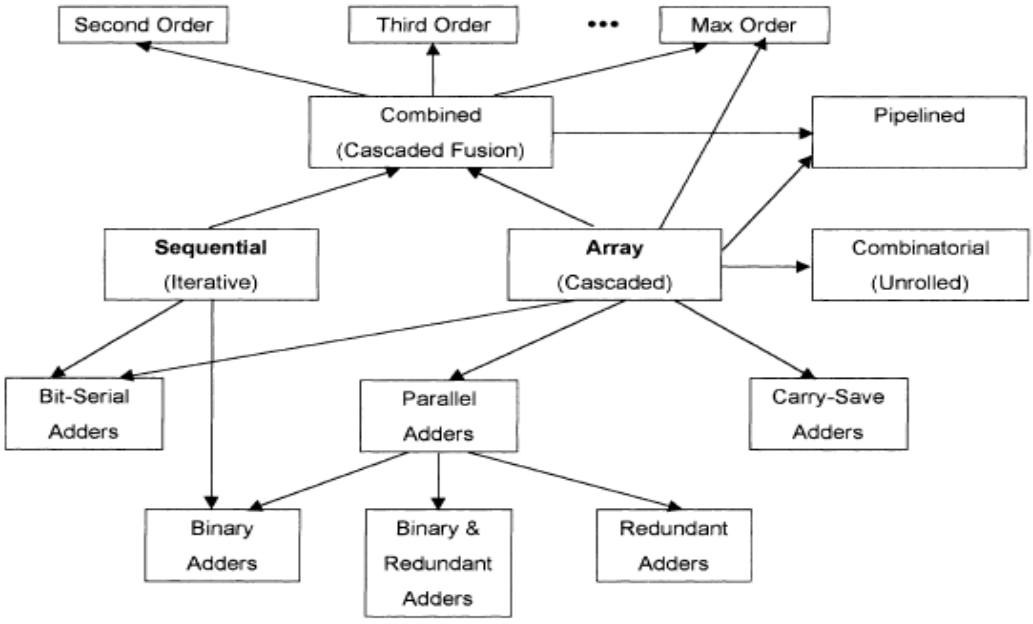

Figure 1. CORDIC hardware implementations

3. Redundant CORDIC Schemes

The introduction of the RSD system into the internal computation of the CORDIC method is considered to be one of the most effective ways to accelerate the algorithm. Cascaded designs of redundant CORDIC schemes have outperformed array implementations of CORDIC based on carry-save adders according to a comparative study of these methods in. A comparison of the latency of conventional CORDIC and different modifications of redundant. CORDIC has been carried out with all designs being of array type.

The latency of the designs expressed as a function of the bit-length is given in Table 1, where t - delay of a full adder; t (log 2 n) - the upper bound of a n-bit non-redundant fast addition; d - delay of a redundant adder, independent of the bit-length; m – an arbitrary integer in the correcting method. The termination algorithm originally proposed by allows quitting the iteration process as early as possible, modified Booth encoding can be used for the same purpose.

Name	Latency expression as a function of the bit length n
Non-redundant method	$n\ t\log_2 n$
Double rotation method	$nt \quad 2nd \quad t\log_2 n$
Correcting method	$n \quad n \quad 1 \quad m \quad t \quad d \quad 2 \quad n \quad 1 \quad m \quad \log_2 n\,(t \quad d)$
Prediction method	$nd \quad t\log_3 n \quad 1\ \log_2 n \quad t\log_2 n$
Prediction with termination method	$d\,(n \quad 1)\ 2 \quad t\ \log_3((n \quad 1)\ 2 \quad 1\ \log_2 n \quad t\log_2 n\ d\log(n\ 2)$

Table1.Latency Expression of CORDIC implementations

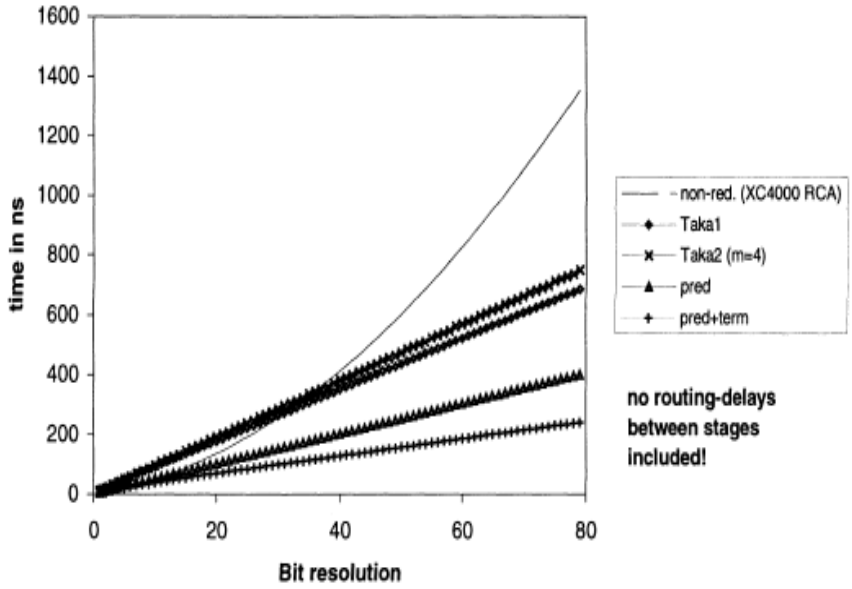

Figure 2. Estimated latency of CORDIC implementations in XC4000XL

Figure 2 shows graphically the latency of the CORDIC implementations using estimated delays for XC4000XL. The latency results are shown in Figure 4, where the ripple-carry adder is referred to as RCA and the redundant adder is referred to as ISDA.

4. Redundant Adder Implementation

The conventional one-bit full adder assumes positive weights to all of its three binary inputs and two binary outputs. Such adders can be generalized to four types of adder cells by imposing positive and negative weights to the binary input/output terminals. The addition of two redundant signed-digit numbers Y and Z can be performed by cascading two levels of generalized full adders of types 1 and 2 as shown in Figure 3.

The main drawback of this computation scheme with two numbers in redundant form is the amount of hardware, which is twice that in the carry-save case.

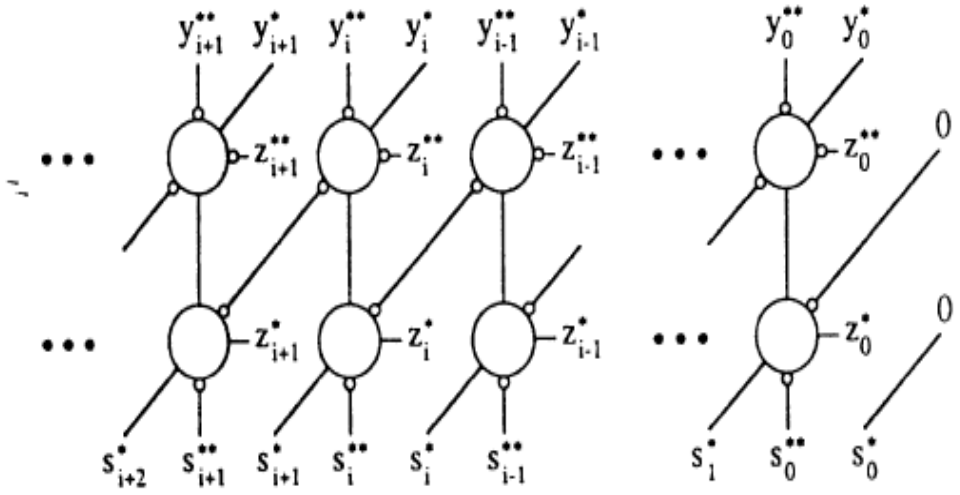

Figure3. Redundant signed digit adder

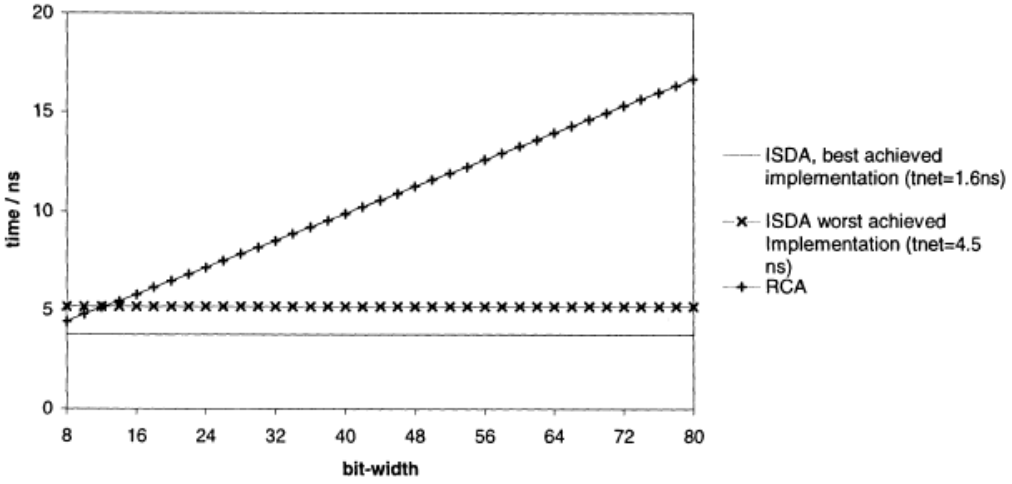

Figure4. Latency comparison between a ripple-carry adder and a redundant adder

5. Experimental Results

We have implemented iterative and cascaded sine and cosine CORDIC-based generators in Actel and XILINX FPGAs using fast binary adders. The number of the iterations in all designs was equal to the bit-length. The bit-

lengths used were 12, 14,16, 24 and 32 bit for the iterative designs and 12, 14 and 16 bit for the cascaded designs. Synthesis results in terms of module count and speed are summarised in Table 2 where results for both area and delay optimised designs are presented. Four different synthesis tools have been used – Actmap 3.5.04, Symplify 5.1.4, Spectrum 5.69 and XILINX Foundation Series Express 1.5i. The speed estimates in the two rightmost columns of the tables are based on back-annotated delays and indicate the value of the maximal data rate achieved and the maximal clock frequency. The experimental results show that module count and operating speed depend significantly on the used synthesis tool. The Actel-based designs are

faster than the XILINX-based ones. A 32-bit 1.9 Msps

iterative sine/cosine generator can be implemented in a small FPGA (Actel SX16-3). A 16-bit cascaded design is not possible to be fitted in a XC4010XL device, this is not surprising, the parallel implementation approach is a trade-off of area for speed where the area increase is of quadratic order with respect to the bit-length.

Table 2. CORDIC synthesis results based on ACTEL FPGAs.

Designs A54SX16-3	Length	Actmap [1] 3.5.04	Synplify [1] 5.1.4	Spectrum [1] 5.69	Speed [2]	Data rate	Frequency
	bits	Area/Delay [4]	Area/Delay [4]	Area/Delay [4]	ns	Msps	MHz
Iterative	12	420/574	307/334	347/424	169.5	5.9	71.4 [5]
Iterative	14	538/784	399/414	428/536	192.3	5.2	72.5 [5]
Iterative	16	674/958	424/462	501/633	232.5	4.3	68.5 [5]
Iterative	24	1170/----	694/727	995/1248	357.2	2.8	66.6 [6]
Iterative	32	1963/----	887/1000	1419/1710	526.3	1.9	62.5 [6]
Cascaded	12	----/----	862/888	1326/1378	44.8	22.3 [6]	
Cascaded	14	----/----	1970/----	2164/2164	192.3	5.2 [3]	
Cascaded	16	----/----	2853/----	2941/3718	222.2	4.5 [3]	

Direct digital synthesis (DDS) generates a new frequency based upon an original reference frequency. Virtually all DDS architectures include a lookup table that performs a sine computation function for generating sinusoidal output signals. For comparison purposes we have designed and synthesized a LUT that is an improved version of the modified Sutherland architecture.

6. Conclusions

This paper presents theoretical and practical aspects of implementing sine/cosine CORDIC-based generators in FPGAs. The main results can be summarized as follows: A trade-off speed/area will determine the right structural approach to CORDIC FPGA implementation for

an application. Module count and operating speed depend significantly on the used synthesis tool. Simulation has shown that the redundant adder can improve the efficiency of CORDIC FPGA implementations for bit-lengths higher than 32-bit.

REFERENCES

[1] R.Andraka. A Survey of CORDIC Algorithms for FPGA Based Computersn – Proc. Of the 1998 CM/SIGDA Sixth International Symposium on FPGAs, February1998, Monterey, CA, pp.191-200

[2] P.W.Baker. Suggestion for a Binary Cosine Generator, IEEE Transactions on Computers, February, 1975, pp. 1134-1136.

[3] M.Marx. FPGA Implementation of sin(x) and cos(x) Generators Using the CORDIC Algorithm, Final Year Project Report, School of Electronic Engineering, University of Surrey, Guidford, UK, 1999.

[4] P.Pirsch. Architectures for Digital Signal Processing, John Wiley & Sons,1998.

[5] N.Takagi. Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation, IEEE Trans. On Comput., vol. 40, n 9, 1991, pp. 989-994.

[6] D.Timmerman, H.Hahn, B.J.Hosticka, B.Rix. A New Addition Scheme and Fast Scaling Factor Compensation Methods for CORDIC algorithms, Integration – the VLSI Journal, vol. 11, n 1, 1991, pp. 85-100.

[7] A.Vlachos. Design and Implementation of CORDIC Modules for ADCS, MSc Project Report, School of Electronic Engineering, University of Surrey, Guidford, UK, 1999.

[8] J.Volder. The CORDIC Computing Technique, IRE Trans. Comput., Sept.1959, pp.330-334.

[9] J.S. Walther. A Unified Algorithm for Elementary Functions, Proc. AFIPS Spring Joint Computer Conference, pp.379-385, 1971.

[10] S.Wang, V.Piuri. A Unified View of CORDIC Processor Design, in Application Specific Processors, Ed. By Earl E. Swatzlander, Jr., Kluwer Academic Press, 1996, pp.121-160.