VLSI Architecture for Pre-computation of Rotation Bits in Unidirectional Flat-CORDIC

Satish Ravichandran and Vijayan K. Asari
Department of Electrical and Computer Engineering
Old Dominion University, Norfolk, VA 23529

Abstract – *CORDIC is an iterative algorithm to compute the trigonometric, logarithmic and transcendental functions for digital signal processing applications. Flat-CORDIC unravels the basic add and shift operations in CORDIC and uses pre-computed values of direction of rotation to perform the computations simultaneously. A new algorithm for the implementation of CORDIC with unidirectional vector rotation is proposed in this paper, which reduces the number of iterations significantly. A novel architecture for pre-computing the rotation bits for flat-CORDIC implementation is also presented. The architectures are being implemented using Altera Quartus II for programming the Apex II FPGA chip, and the results are encouraging.*

1. Introduction

The digital signal processing techniques have been used in many fields today from Internet to wireless phones. DSP requires complex mathematical computations and there is a need for a high-speed mathematical processor for an efficient signal processing application. CORDIC is an acronym for COordinate Rotation Digital Computer was first introduced by Volder [1] in 1959 and later was bought into light again by Walther [2] in 1971. CORDIC is an iterative hardware efficient algorithm that does high-speed mathematical operation in linear, circular and hyperbolic coordinate system. CORDIC architectures find its use in many digital signal and image processing [3][6], digital filter design, matrix inversion, Eigen value computation, SVD algorithms [10] etc. This paper attempts to develop a novel method for fast and area efficient CORDIC architecture.

The architecture design for CORDIC has been undergoing various modifications to improve the performance as well as the VLSI area required for its implementation. The number of iterations is halved by using a radix-4 architecture [4] but the complexity of the hardware is increased. Carry-save additions [5] are incorporated in the iterations thereby reducing the complexity of the hardware. Other architectures and algorithms such as Backward Angle Recording [7] algorithm, Hybrid algorithms [8] and pipeline architecture have been used to reduce the number of iterations and the hardware used. Most of the architectures discussed reduce the number of iterations by using a constant scaling factor for multiplication. One proposed architecture to improve the performance is unrolled or flat CORDIC architecture where, each iteration is done independently and concurrently, thereby reducing the time of operation but increasing the hardware real estate.

Flat CORDIC involves unrolling the iterations and executing the iteration in parallel. To build this architecture, the sign of rotation (clockwise or counterclockwise) is pre-computed and these signed bits are used to generate the angle. Therefore, generation of signed bits play a very important and crucial role in the performance of the CORDIC architecture. One of the

advantages of using the flat architecture is that the speed of operation of the device would be increased at the cost of the VLSI area needed to implement the CORDIC algorithm. Since, the architecture is highly parallel, the amount of space used in building this architecture is more compared to other conventional architectures. In this paper, a technique is proposed to pre compute the signed digits and also optimize the speed, area and power of the architecture. The conventional CORDIC rotates in two directional (clockwise or counterclockwise). A new algorithm is proposed which uses only unidirectional rotation. Therefore the value to be measured is obtained by iterating through only counter-clockwise direction and skipping the angle in the other direction. Thus, the number of iterations for any given value is reduced significantly depending on the input value.

2. The CORDIC Algorithm

The algorithm of CORDIC proposed by Volder, is derived from the two rotation equations [9],

$$x' = x\cos\phi - y\sin\phi \quad (1a)$$

$$y' = y\cos\phi + x\sin\phi \quad (1b)$$

The equations represent a vector (x, y) that is rotated by an angle ϕ. This equation is modified such that the rotations are characterized by an iterative shift operation in digital logic. Therefore, the modified equation is given by,

$$x_{i+1} = x_i - y_i .m.d_i.2^{-i} \quad (2a)$$

$$y_{i+1} = y_i + x_i .m.d_i.2^{-i} \quad (2b)$$

where $m = -1, 0, 1$ and it represents the rotation in the specific coordinate system ($m=0$ for linear, $m = 1$ for circular and $m = -1$ for hyperbolic system) and $d_i = \pm 1$ represents the sign of rotation ($d_i = 1$ for counterclockwise rotation and $d_i = -1$ for clockwise rotation).

As discussed above, in Flat CORDIC architecture, the angle is fed as the input and the signs of rotations are to be pre-computed from the given angle. A mathematical treatment of pre-computing the angles for any given angle is presented in [11]. It can be seen that we need to compute only the first (N/3 –1) bits of the angle and the signed bits are taken directly from the remaining bits. Initially, all the values of the signed bits corresponding to the first (N/3 –1) digits are pre-stored in a ROM.

3. Implementation of the unidirectional CORDIC Algorithm

The basic idea behind the proposed CORDIC algorithm is that the rotation is done only in the counter-clock direction. Initially, the angle whose corresponding trigonometric values are to be computed is stored in a register. As explained before, the signed bits for the first (N/3–1) bits are calculated separately, as these bits dominate in calculating the accuracy of any given angle. The LSB (Least Significant Bits) (N/3–1) to N bits is found out from the remainder of the angle obtained after the first (N/3–1) bits are calculated. The corresponding value of the remainder found is added to the remaining angle from the input angle (N/3 – N). The algorithm used in the calculation of the first (N/3–1) bits is given below. The angles corresponding to the (N/3 – 1) bits are calculated and loaded in the registers.

Algorithm:

Step 1: Take the first (N/3 – 1) bits from the input and is stored in register. The remaining bits are stored in another register REM.

Step 2: The first of the (N/3 –1) bit is checked for a 0 or 1. If it's a 1, then sign bit [0] = 1 and the REM is fed back with the value of REM – the stored angle for the corresponding bit position. If the bit in the REM is a '0' then sign bit [0] = 0 and REM value is not changed.

Step 3: Check whether (N/3 – 1) bits are done. If yes goto Step 4, else goto Step 2 checking the next bit value of the REM register.

Step 4: Add the remainder with the last 16 bits of the input.

Step 5: From the remainder values, the corresponding last 0 – N/3 bits are updated in the sign register.

It is seen from the algorithm that in step 2, the register bit is taken and put onto the sign bit. The remaining angle is found only when the bit value is 1, thereby reducing the number of subtractions required. Once the calculation for the first (N/3 –1) bits is done, the remainder is added with the last N/3 to N bits of the input and the resultant is given to the LSB's of the sign bit. On an addition, it is noted that there is a possibility of an occurrence of a carry bit, which also should be taken into account and is stored as the $N/3^{th}$ bit in the sign register. Therefore the sign register always consists of N bits as compared to the N–1 input bits. The extra carry bit is placed as the $(N/3 -1)^{th}$ bit in the sign register. Thus the entire N signed bits values representing the rotation are got to the sign register.

Architecture:

From this algorithm, architecture is built that generates the sign bits given any angle. The architecture is built and simulated using Altera Quartus II. The objective of building architecture is to reduce the size such that the number of cells occupied when burnt on an FPGA is minimal. From the algorithm, it is seen that the architecture contains no ROM, as is done the conventional Flat CORDIC architecture. The memory access time is thus reduced in the proposed method. An architecture based on the algorithm is shown in Fig. 1.

The architecture shown is for 24-bit input. Therefore bits 0 – 7 are considered critical bits and the generation of the sign bit for these bits is through a combinational logic. For the remaining 16 bits the sign bits are generated from the adder. Initially, the first 8 bits are extracted and stored in a separate register REM. The remaining angle (16 bits) is stored in another register REM1. Since the number of critical angle bits is 8, eight angle registers are loaded with their respective angle values. The first 8 bits are extracted one after another and passed through a combinational logic and is loaded onto the sign register. Fig 2 shows the detailed working of the combinational logic. Once the first 8 bits are found, the values of registers REM and REM1 are added and the last 16 bits of the sign bits are loaded directly into it. Once the signed bits are found and these are issued to the main CORDIC architecture and the respective trigonometric and logarithmic values are obtained. The details about the hardware simulation used and results obtained from the experimental set-up are discussed in the next section. Thus, the pre-computation of signed digits is performed to build the Flat CORDIC.

3.2.4

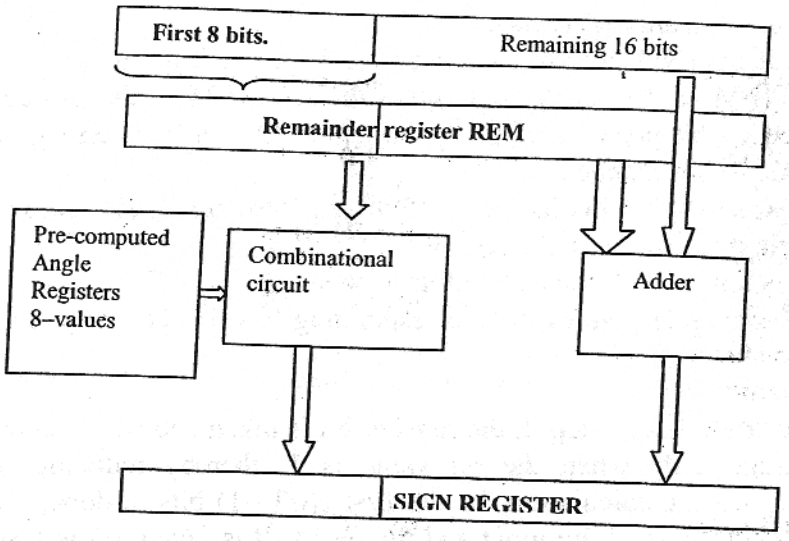

Figure 1: Pre-computation signed digits evaluation architecture

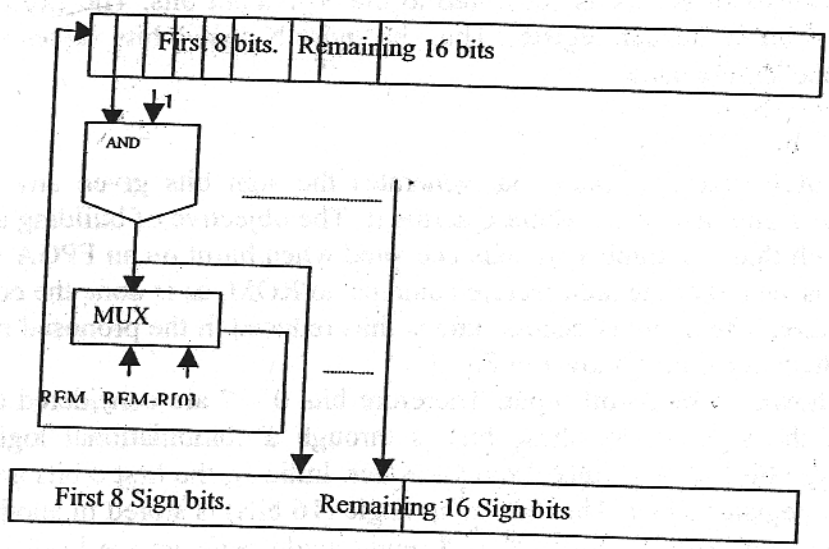

Figure 2: Combinational logic circuit used in the architecture

4. Simulation Results and Discussion

The architecture designed is simulated using Altera Quartus II VHDL. A special case of N=24 is selected to simulate the architecture. For N=24 the first N/3 critical angle is the first 8 bits. A particular angle of 35° is selected as an example. The binary equivalent of this angle in radians is given to the simulation as input. The details about the angles and the results are given in Table 1. The 24-bit angle for the eight different cases is stored in eight different registers and the signed digit calculation for the first eight input bits is done using simple combinational logic and stored in the final sign bit register. The direction of rotation is stored, as 1 or 0, where 1 represents a

rotation in counterclockwise direction and 0 represent no rotation. Once the first eight bits are calculated, the remaining angle is added and is stored in the sign register.

The number of clock cycles required is 8 (for the first eight bits) + 1 (for addition). It can also be seen that dividing the input angle to N/3 bits is the optimal way for generating the signed bits, which approximately has a 24-bit precision. Additional clock cycles are saved when computing other trigonometric functions of the input angle. This is because the iterations undergo no rotation during a 0 sign bit. The number of savings in the clock cycle depends on the number of 0's present while computing the sign bits.

With regard to the real estate occupied by the hardware, the prime constituent of the real estate- memory - is not present in this design. This implies that a smaller area is sufficient to accommodate the architecture compared to the conventional Flat CORDIC architecture, which would have required a 256×24-bit ROM for the same specification. From the experimental results it can be seen that the number of logic cells used is 290 operating at a frequency of 3GHz. The FPGA device that is used to build the chip is EP20K100QC208-1, which comes under the family of APEX20K. A comparison of two Flat CORDIC pre-computation architectures is presented in Table 2. It can be seen that the earlier architecture uses 20 times as much as space as compared to the proposed method. This saving in space is compensated to the number of clock cycles used. The proposed architecture uses 9 clock cycles to compute the values compared to 4 clock cycles used by the earlier architecture. But since uni-directional CORDIC is here, the number of iterations required to compute any value is reduced significantly, thereby compensating for the 5 extra clock cycles used for pre-computation.

Table 1. Simulation results for 24-bit input angle

Input Angle	35.0
Input Angle (in radians)	0.610865238
Binary Equivalent of the angle (24 bits)	100111000110000110101010
Value of the first 8 bits	0.609375
Remaining 17 bit value	0.001490238
First 8 sign bit	10100101
Remainder after the first 8 iteration	0.001842437
Total Remaining Angle	0.003332675
Signed digits for the last 16 bits	1101101001101011

Table 2. Comparison of VLSI properties of two architectures

	Proposed architecture	Architecture in [11]
Number of Logic cells	290	6200
Number of clock cycles	9	4

5. Conclusion

A novel algorithm using only one direction of vector rotations to implement the Flat CORDIC architecture has been presented. The unidirectional CORDIC algorithm and the development of the pre-computation of the signed digits for Flat-CORDIC has been described. It has been observed that the proposed technique is efficient with respect to the speed (by reducing the number of iterations) and also the VLSI area (by eliminating the memory) requirement in the implementation of the Flat-CORDIC hardware.

References

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

[2] J.S. Walther, "A unified algorithm for elementary functions," *Proc. Joint Computer Conf.*, pp. 379-385, 1971.

[3] Y.H. Hu, " CORDIC-based vlsi architecture for digital signal processing." *IEEE Trans. Computers*, vol.42, no.1, pp. 99-102, Jan. 1993.

[4] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm," *IEEE Trans. Computers*, vol.46, no.8, pp. 855-870, Aug. 1997.

[5] H. Dawid and H. Meyr, "The differential CORDIC algorithm: constant scale factor redundant implementation without correcting iterations." *IEEE Trans. Computers*, vol. 45, no. 3, pp. 307-318, Mar. 1996.

[6] M. Kuhlmann and K. K. Parhi, "A high-speed CORDIC algorithm and architecture for dsp applications," *IEEE Workshop on Signal Processing Systems (SiPS) Design and Implementation*, Oct. 1999.

[7] Yu Hen Hu and Homer H.M. Chern, "A novel implementation of CORDIC algorithm using backward angle recording (BAR)," *IEEE Trans. Computers*, vol. 45, no. 12, pp. 1370-1378, Dec. 1996.

[8] S. Wang, V. Piluri and E. E. Swartzlander, "Hybrid CORDIC algorithms," *IEEE Trans. Computers*, vol. 46, no. 11, pp. 1202-1207, Nov. 1997.

[9] R. Andraka, "A survey of CORDIC algorithms for FPGAs," *Proceedings of the 1998 ACM/SIGDA sixth international Symposium on Field Programmable Gate Arrays, FPGA '98*, pp. 191-200, Feb. 1998.

[10] L.H. Sibul and A.L. Fogelsanger, "Application of coordinate rotation and function generation," *IEEE Int. Symp. Circuits and Systems*, pp. 821-824, 1984.

[11] B. Gisuthan, T. Srikanthan and K. V. Asari, "Design of an efficient digital architecture for the pre-synthesis of direction of micro-rotations in Flat-CORDIC", *Proc. Second International Conference on Information, Communications & Signal Processing - ICICS'99*, no. 2b2-1, no. 261, pp. 01-05, Dec. 1999.