



Design of Coordinate Rotation Digital Computer (Cordic) for Angle Rotation

Pradeep. Kalakoti
Assistant Professor
BITS, NSPT, T.G. India

Ram Prasad. Mudide
Assistant Professor
T.G. India

Abstract: Rotation of vectors from fixed and known angles has wide applications in many fields. We are presenting optimization schemes and CORDIC circuits for fixed and known rotations with different levels of accuracy. In order to reduce the area- and time-complexities, we proposing a hardwired pre-shifting scheme in barrel-shifters of the proposed circuits. Two dedicated CORDIC cells are proposed for the fixed-angle rotations. In one of those cells, micro-rotations and scaling are interleaved, and in the other they are implemented in two separate stages. Pipelined schemes are suggested further for cascading dedicated single-rotation units and bi-rotation CORDIC units for high-throughput and reduced latency implementations. We have obtained the optimized set of micro-rotations for fixed and known angles. We have synthesized the proposed CORDIC cells by Synopsys Design Compiler using TSMC 90-nm library, and shown that the proposed designs offer higher throughput, less latency and less area-delay product than the reference CORDIC design for fixed and known angles of rotation. We find similar results of synthesis for different Xilinx field-programmable gate-array platforms.

Key words: CORDIC, Angle rotation, Micro rotations, Bi-rotation, FPGA.

I. INTRODUCTION

The CORDIC algorithm[2] was first introduced by Jack E. Volder in the year 1959 for the computation of mathematical functions[3], and Logarithms. He stated that it is a highly efficient, low-complexity, and robust technique to compute the elementary functions. The CORDIC works in two different modes, they are Rotation Mode and Vector Mode. In Rotation Mode the algorithm rotates a vector with a given angle, and in Vector Mode the angle between a given vector and the x-axis is calculated. In this project to find trigonometric functions CORDIC works in rotation mode only. Volder's algorithm is derived from the general equations for a vector rotation. If a vector V with coordinates (x, y) is rotated through an angle Φ then a new vector V' can be obtained with coordinates (x', y') where x' and y' can be obtained using x, y and Φ

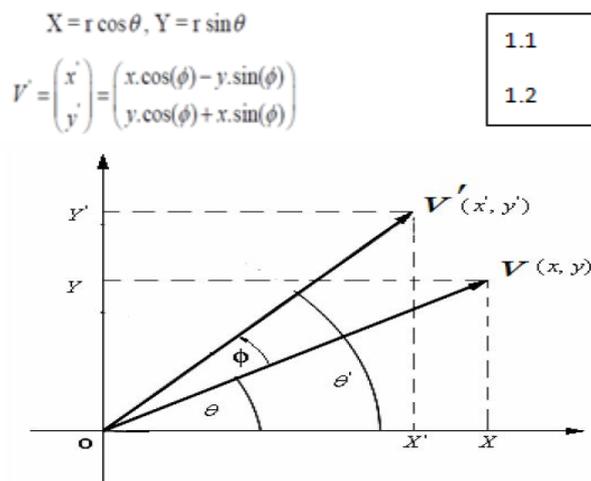


Fig: Rotation of a vector V by the angle Φ

As shown in the above figure, a vector $V(x, y)$ can be resolved in two parts along the x -axis and y -axis as $r \cos \Phi$ and $r \sin \Phi$ respectively. below Figure illustrates the rotation of a vector.

$$V = \begin{pmatrix} x \\ y \end{pmatrix} \text{ by the angle } \phi .$$

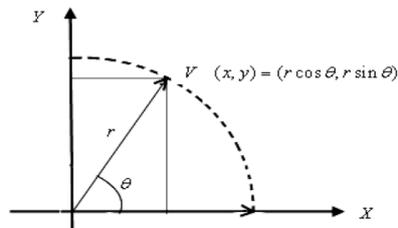


Fig: Vector V with magnitude r and phase Φ

II. CORDIC HARDWARE [1]

A straight forward hardware implementation for CORDIC arithmetic is shown below. It requires three registers for x, y and z, a look up table to store the values of $\alpha_i = \tan^{-1} 2^{-i}$, and two shifter to supply the terms $2^{-i}x$ and $2^{-i}y$ to the adder/subtractor units. The d_i factor (-1 and 1) is accommodated by selecting the (shift) operand or its complement.

A single adder and one shifter can be shared by three computations if a reduction in speed by a factor of 3 is acceptable. Where high speed is not required and minimizing the hardware cost is important (as in calculator), the adder can be bit serial. Then with k bit operands, $O(k^2)$ clock cycle would be required to complete the k CORDIC iterations. This is acceptable for hand handled calculators, since even a delay of tens of thousands of clock cycles constitutes a small fraction of a second and thus is hardly noticeable to a human user. The calculations in each iteration step are performed by shift, addition and subtraction, and recall of a prepared constant. Since the rotation is not a pure rotation but a rotation-extension, the number of rotations for each angle should be a constant independent of the operand so that the scale factor becomes a constant.

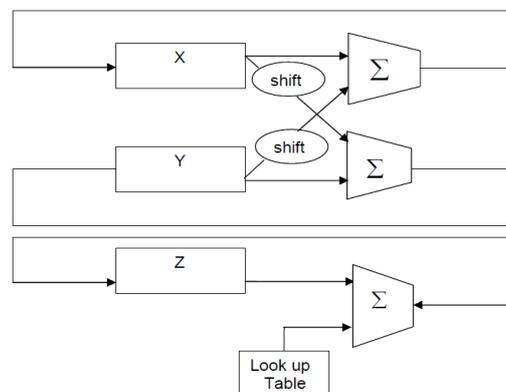


Fig Hardware elements needed for the CORDIC method.

Hardware implementation

It is thus possible to remove the first x and y adders in the unrolled chain without any additional hardware cost. The input angle α_i is between 0 and $\pi/2 > 1.75$, which means that at least two fractional bits are used. In addition, to be able to keep the same number representation throughout the chain, also negative numbers have to be handled, requiring an extra sign/integer bit.

Also in the x and y streams two integer bits, including the sign bit are needed. The sign bit is needed, since both x and y values take negative values (close to angles 0 and $\pi/2$ respectively). The need for extra bit is not obvious at first, but due to optimization, it is possible to overflow and end up at values $x, y > 1$ in certain stages of CORDIC [4]. To safely handle this extra integer bit is needed.

Original Bit-Serial Architecture

The figure shows architecture with 4 stages and a word length of 8-bits. As shown, the three paths the angular and two vector paths are different but highly dependent data paths.

To perform a subtraction the control bit of the adder, connected to the A/S unit is set to '1'. This inverts the second operand and sets the input sign bit to '1'. The control bit is the sign bit of the previous stage angle operation. Thus angle calculation of the previous stage must be completed before the addition or subtraction can begin.

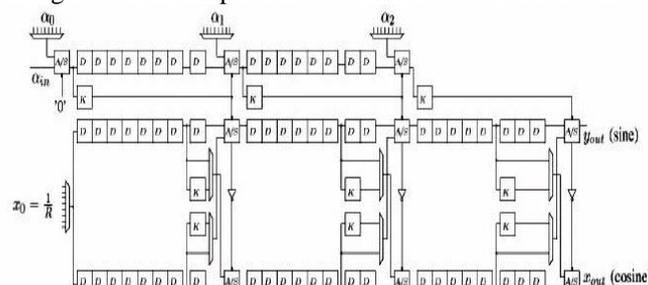


Fig: Bit-serial architecture; 4 stages, 8-bit words.

Improved Bit-Serial Architecture

Removing the registers changes the timing of the circuit. In order for this to work, the angular path calculations must begin earlier than the corresponding vector path calculations. Also, to not increase the latency or reduce the throughput of the design, the vector path calculations should start so that the final add/subtract operation, starts exactly when the final angular operation, starts exactly when the final angular operation is finished.

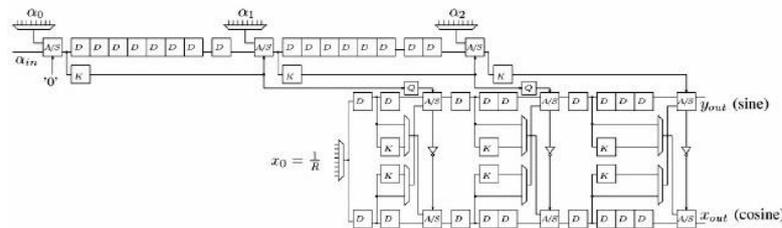


Fig: Improved Bit-Serial Architecture; 4 stages, 8-bit words.

Control Unit A bit-serial implementation has an inherent need for a control unit. It is important to reset a bit-serial A/S unit before standing a word operation. This is because the bit-serial units need to keep a state, i.e., the carry bit. That means that the controller somehow must implement a counter, counting up to the data, to ensure that a new reset signal is emitted before each new word operation is initiated.

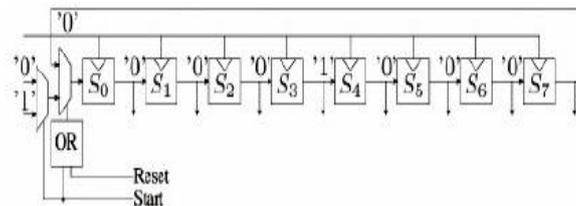
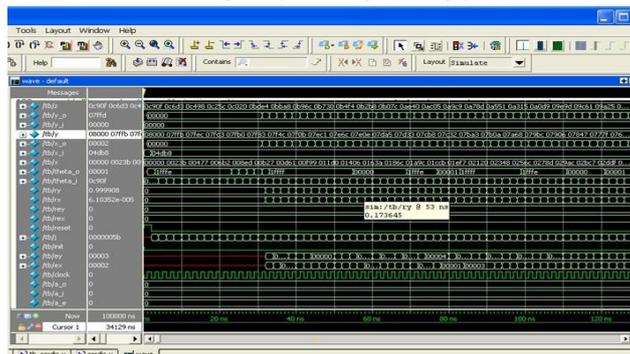
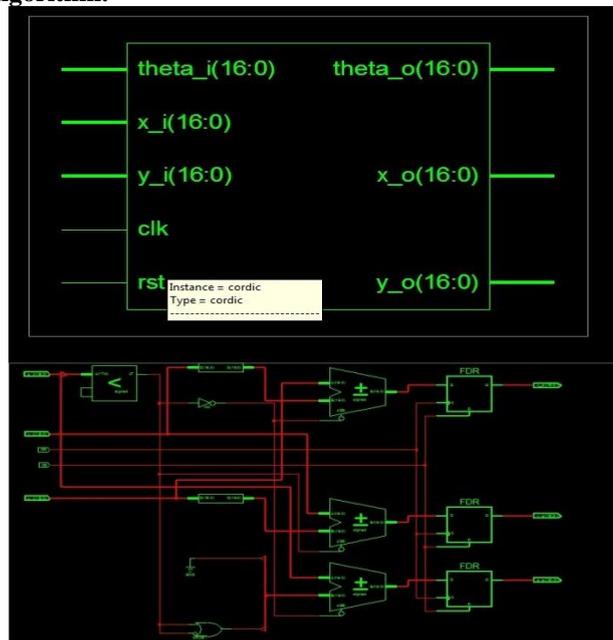


Fig: Control unit of CORDIC architecture

III. SIMULATION RESULT



RTL schematic of CORDIC algorithm:



IV. CONCLUSION

In this paper, a new and improved bit-serial CORDIC architecture is presented. It is shown how it is possible to reduce the number of registers by calculating the angular path prior to the vector paths in the CORDIC. Some extra registers and extra logic are needed to store the sign bits of the angular calculations. However, this is substantially less than the area reduction in the vector paths. The improved architecture is 20 % smaller and consumes 26 % less power.

V. FUTURE SCOPE

In Future the CORDIC cell with interleaved scaling involves ~4% more area giving more throughput above 43% with less latency 30% for known and fixed rotations. Bi-rotation is also to be implemented which improves above factors more by doubling and improving high-throughput and low latency implementation.

REFERENCES

- [1] Johan Lofgren and Peter Nilsson, Dept. of Electrical and information technology, Lund University, Bit Serial CORDIC: Architecture and Implementation Improvements, Sweden
- [2] Volder J. E., .The CORDIC trigonometric computing technique., IRE Trans. Electronic Computing, Volume EC-8, pp 330 - 334, 1959.
- [3] Qian M., .Application of CORDIC Algorithm to Neural Networks VLSI Design., IMACS Multiconference on .Computational Engineering in Systems Applications (CESA)., Beijing, China, October 4-6, 2006.
- [4] Lin C. H. and Wu A. Y., .Mixed-Scaling-Rotation CORDIC (MSR-CORDIC) Algorithm and Architecture for High-Performance Vector Rotational DSP Applications., Volume 52, pp 2385- 2398, November 2005
- [5] Walther J.S.A., .Unified algorithm for elementary functions., Spring Joint Computer Conference, pp 379 - 385, Atlantic city, 1971.