ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Folded CORDIC Architectures using Hybrid-Radix on FPGA

Ch.Nishitha
M.Tech VLSI,
Department of ECE,
CMR Institute of Technology.

Mrs.S.Sri Bindhu
Assistant Professor,
Department of ECE,
CMR Institute of Technology.

Ganesh
Assistant Professor,
Department of ECE,
CMR Institute of Technology.

Abstract:

CORDIC algorithms have long been used in digital signal processing for calculating trigonometric, hyperbolic, logarithmic and other transcendental functions. The algorithm requires only shift and add operations and this simplicity encourages its implementation in hardware. Traditional CORDIC architectures have focused on radix-2 implementations because of their higher accuracy. However these architectures are slow, requiring a lot of iterations to converge to a given solution. Radix-4 and higher radix architectures have been proposed to speed up the process by reducing the number of iterations, but they suffer from poor accuracy. In this paper a hybrid-radix approach to CORDIC implementation is proposed. By using this approach the algorithm can be implemented with higher speed, lower power and lesser area utilization and at the same time a good accuracy can be achieved. Further the hybrid-radix architecture has been retimed resulting in an increase in the overall throughput which is particularly important in DSP applications.

Keywords:

CORDIC, DSP, Hybrid arithmetic, VLSI, Retiming, Unfolded structures, Folded Structures.

INTRODUCTION:

CORDIC (COordinate Rotation Digital Computer) [1] [2] is a simple shift and add algorithm that has been used in Digital Signal Processing (DSP) systems [3] for calculating various linear and transcendental functions. Conventional implementations of CORDIC have been software oriented. However, the development in the design of high speed Very Large Scale Integration (VLSI) architectures has provided the designers with significant impetus to map the algorithm into architecture [4]. This has enabled the designers to assess the performance in terms of some realistic parameters like throughput, area and power.

Typically, CORDIC algorithm has been implemented as a sequential structure [5]. This implementation, although area efficient has large iteration periods and is not, therefore, preferred for DSP applications. Sequential structures have been used for Application Specific Integrated Circuits (ASIC) implementations where area is of concern. However, with Field Programmable Gate Arrays (FPGAs) the underlying platform provides a huge logic capacity [6] [7] [8] [9] that can be utilized to develop architectures that are optimized for speed and power.

This has enabled designers to exploit the concurrencies [10] within the algorithm and develop unfolded architectures that map well on FPGAs [11] [12]. The unfolding process results in parallel structures where each processing element performs the same iteration.The unfolded CORDIC structures which are based on radix- 2 arithmetic give results with good accuracy.

However, these structures require a lot of iterative stages to converge to a given solution. The critical paths associated with these structures are thus very large, consequently resulting in slow structures [10]. By using radix-4 arithmetic the number of iterations is reduced [13] by half but the results have very poor accuracy. This paper implements a hybrid-radix structure where the accuracy of radix-2 structure is preserved while maintaining a suitable trade-off between speed, area and power.

Further the hybrid structure is analyzed for timing behavior and a retimed structure is developed which further increases the speed of the architecture. The rest of the paper is organized in this manner. Section II briefly discusses the CORDIC algorithm and its operating modes with special reference to radix-2 and radix-4 structures. Section III discusses the hybrid-radix CORDIC structure. Section IV discusses the use of retiming to speed up the structure. Section V provides the synthesis, implementation and simulation details. Conclusions are drawn in section VI and references are listed in the end.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

CORDIC ALGORITHM:

The CORDIC algorithm provides an iterative approach that involves the rotation of a vector in a linear, circular or hyperbolic coordinate system [1]. The choice of coordinate system depends on the function to be evaluated [4]. The vector rotations are performed using a series of specific incremental rotation angles. The rotation angles are restricted so that;

$$\tan\theta = \pm 2^{-i}$$

This ensures that the multiplication operation is decomposed into simple shift operation such that the algorithm involves only shift and add operations. The generalized equations for CORDIC algorithm when operated in circular coordinate system are:

$$x_{i+1} = x_i - \sigma_i y_i p^{-i}$$

$$y_{i+1} = y_i - \sigma_i x_i p^{-i}$$

$$z_{i+1} = z_i - \sigma_i \alpha_i$$

Where σ_i represents the direction of rotation in each iteration, ρ represents the radix of the number system and α_i gives the amount of shift in each iteration. The iteration process increases the length of the vector in each iteration. The magnitude of the rotating vector in (i+1)th iteration is given as:

$$M_{i+1} = M_i\sqrt{1+\tan^2\alpha}$$
$$= M_i\sqrt{1+\rho^{-2i}}$$

The increase in magnitude depends on the radix of number system and number of iterations and is represented by a scale factor K as:

$$K = \prod_{i=0}^{n-1}\sqrt{1+\rho^{-2i}}$$

CORDIC is operated in two modes: rotation mode and vectoring mode. In rotation mode the angle accumulator is initialized with some angle. The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator. Therefore, the direction of rotation in any iteration is determined using the sign of the residual angle obtained in the previous iteration. When operated in rotation mode the CORDIC algorithm can be used to calculate trigonometric functions. In vectoring mode the input vector is rotated through some angle so as to align the resultant vector with x-axis. The result of this operation is a rotation angle and the scaled magnitude of original vector. When operated in vectoring mode the CORDIC algorithm can be used to calculate the magnitude and the phase response.

CORDIC ARCHITECTURES:

CORDIC architectures are commonly categorized into sequential (folded) and combinational (unfolded) depend on hardware realization of iterative equations [5]. The folded architecture is obtained by the direct duplication of equations 15, 16, and 17. In time domain the sequential architecture has to be multiplexed so that all iterations are approved in a particular functional unit. The signal processing architectures delivers a means for operating area for speed [6]. Using a word sequential design the folded architecture is implemented the unabridged CORDIC core.

REDUNDANT ARITHMETIC:

Redundant Number Systems (RNS) offer an alternate form of computer arithmetic suited to numerically intensive applications. An important property of RNS is that it captures or prevents the carry propagation [8,9], creating parallel adders with constant delay, irrespective of the operand word-length. Thus low latency results are produced in an RNS format. Traditionally CORDIC implementations are based on ripple carry addition. These, however suffer from large internal carry propagation delays. Since the adder/subtractor unit forms a major component of CORDIC architecture, their performance will determine the overall performance of the CORDIC processor. To enhance the performance of CORDIC processors redundant arithmetic has been proposed. This arithmetic, due to its inherent carry-free property avoids the propagation of carry from the LSB to the MSB, resulting in faster operations. This section considers radix-2 hybrid and signed-digit additions and subtractions.

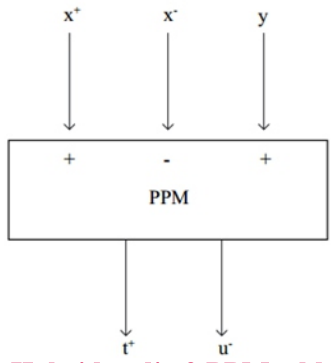


Fig: Hybrid radix 2 PPM adder

UNFOLDED PARALLEL ARCHITECTURE:

The CORDIC processor discussed above is iterative algorithm; it means processor needs to perform n iterations at the given data rate. This is an unfolded operation [8], it means always performs the same iteration at n times processing elements. It shows in Figure 3. This will result the value in two simplifications. First one is fixed shifts at shifters; by using wiring we can implement it. Another one based on lookup table. Here angle accumulator distributed the LUT values, as constants to each adder; this is chain process in accumulator angle. Instead of using storage space we are moving to constants, which are hardwired.

The whole CORDIC processor can be modified into grouping of subtraction-adder unit in interconnected manner. Now we do not require registers, building of this unrolled processor stringently combinatorial. Finally resulting circuit delay should be substantial; now processing time is decreased compare to iterative circuit. Mostly and particularly in FPGA, they do not make any sense of using combinatorial large circuit what we required. The design of unrolled processor can be easily pipelined [9] by inserting registers in-between the subtraction- adder unit block. Moreover FPGAs already contain registers in each logic cell, so this pipelined registers do not increase the hardware cost.

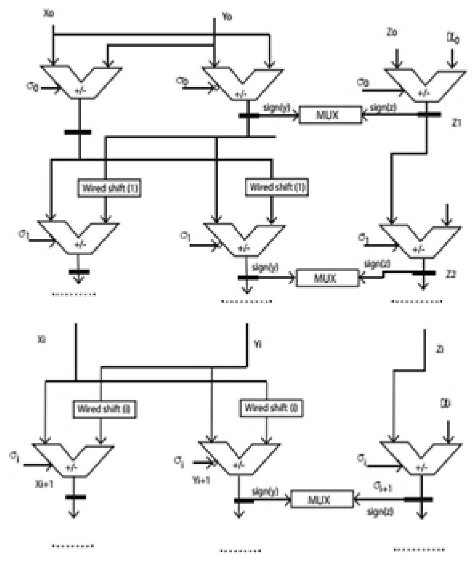


Fig: Unfolded parallel architecture.

SEQUENTIAL DESIGN-FOLDED ARCHITECTURE:

Folded word sequential design [7] is duplicated by using three difference equations in each. This is also called as iterative bit-parallel design, the hardware as shown in Figure 2. Each block contains a shift unit, subtraction-adder unit block, and one register used for output buffering. Initial values are given to register by the multiplexer for first level calculation. Here z-branch of MSB stored value determines the operation mode for the adder-subtraction unit. x and y outlet Signals passes to block of shift unit and finally subtracted or added to un-shifted signal value in reverse path.

The z branch mathematically mixing the registers values, lookup table passes these values, then each number of operations the address value is changed. After n operations output is passes again to register block before primary values are fed in again and these final value can be access as output. Normal FSM needs to control the addressing of the constant values and multiplexers. All the initial values are given hardwired in a word wide manner when it is implemented in FPGA. Both subtracted and adder component are passed out separately. Angle accumulator controls the multiplexer. Routing signals are required to find distinguish between subtraction and addition. For varying the shift distance value, shift operations are used. Shifters are not suitable for FPGA architectures because it desires several layers of logic. Due to numerous layers of logic cells, the resultant design structure will become slow.

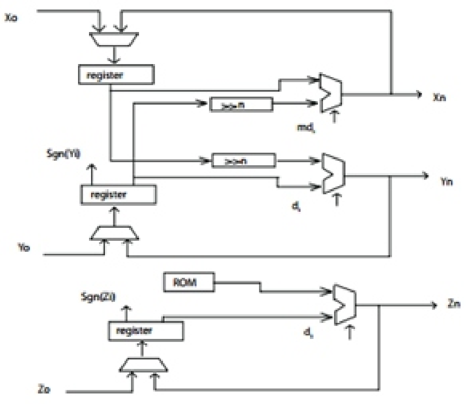


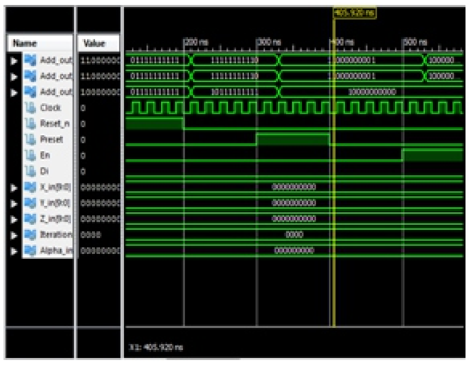
Fig: Sequential design.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

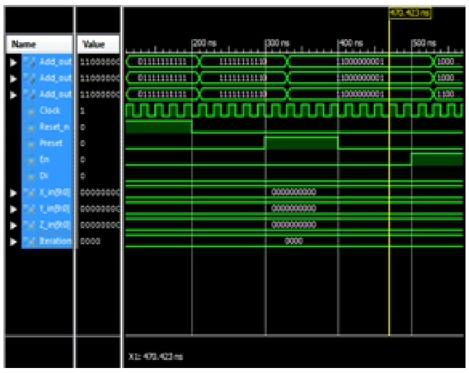
RESULTS

Simulation Results:

Unfolded Simulation Waveform:

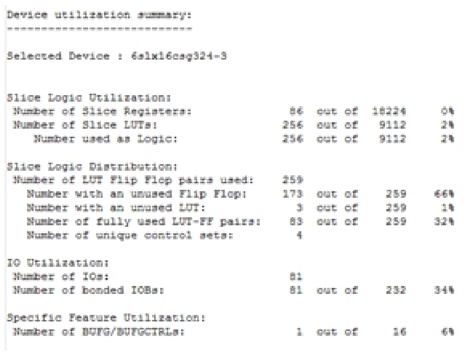


Folded CORDIC Waveform:



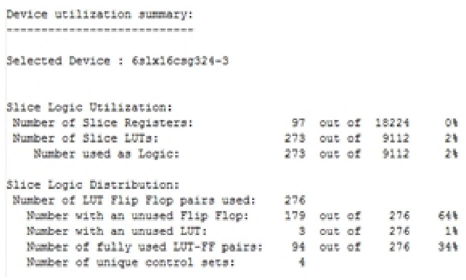
Synthesis Report:

Unfolded Area Report:

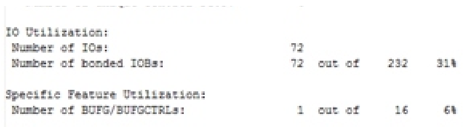


```
Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice Registers:                86  out of  18224     0%
 Number of Slice LUTs:                    256  out of   9112     2%
    Number used as Logic:                 256  out of   9112     2%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    259
   Number with an unused Flip Flop:     173  out of    259    66%
   Number with an unused LUT:             3  out of    259     1%
   Number of fully used LUT-FF pairs:    83  out of    259    32%
   Number of unique control sets:         4

IO Utilization:
 Number of IOs:                          81
 Number of bonded IOBs:                  81  out of    232    34%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                1  out of     16     6%
```

Folded CORDIC Area Report:

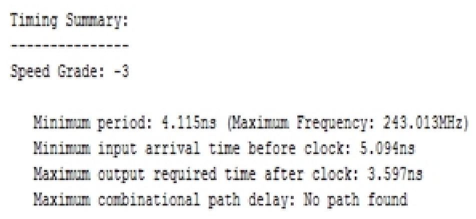


```
Device utilization summary:
---------------------------

Selected Device : 6slx16csg324-3


Slice Logic Utilization:
 Number of Slice Registers:                97  out of  18224     0%
 Number of Slice LUTs:                    273  out of   9112     2%
    Number used as Logic:                 273  out of   9112     2%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    276
   Number with an unused Flip Flop:     179  out of    276    64%
   Number with an unused LUT:             3  out of    276     1%
   Number of fully used LUT-FF pairs:    94  out of    276    34%
   Number of unique control sets:         4
```



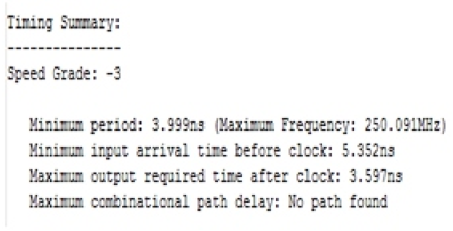
```
IO Utilization:
 Number of IOs:                          72
 Number of bonded IOBs:                  72  out of    232    31%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                1  out of     16     6%
```

Unfolded CORDIC Timing Report:



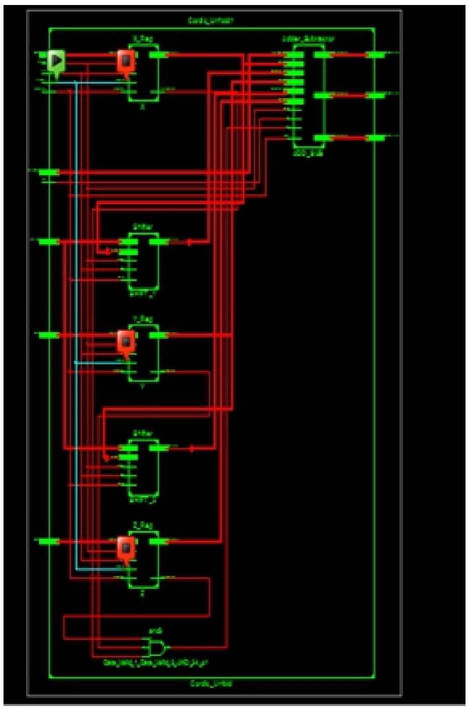
```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 4.115ns (Maximum Frequency: 243.013MHz)
   Minimum input arrival time before clock: 5.094ns
   Maximum output required time after clock: 3.597ns
   Maximum combinational path delay: No path found
```

Folded CORDIC Timing Report:

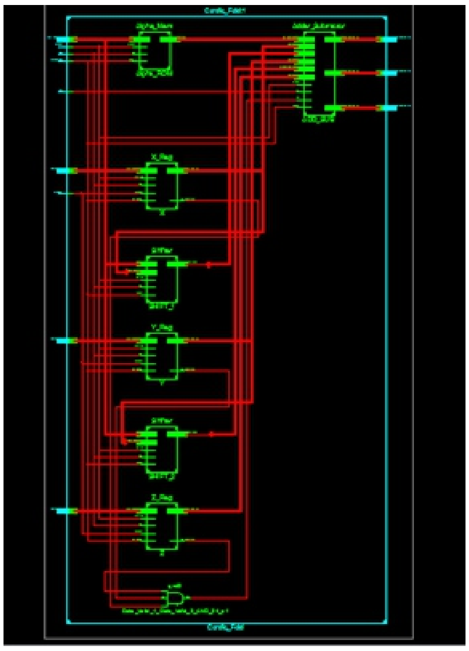


```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 3.999ns (Maximum Frequency: 250.091MHz)
   Minimum input arrival time before clock: 5.352ns
   Maximum output required time after clock: 3.597ns
   Maximum combinational path delay: No path found
```

Unfolded CORDIC RTL Schematic:



Folded CORDIC RTL Schematic:



CONCLUSION:

Here, in this paper the design and implementation of both Folded and Unfolded CORDIC structures using hybrid-radix arithmetic is done. The implemented structures were compared in terms of area, speed, power and accuracy of the simulated results.CORDIC is a powerful algorithm, and a popular algorithm of choice when it comes to various Digital Signal Processing applications. Implementation of a CORDIC-based processor on FPGA gives us a powerful mechanism of implementing complex computations on a platform that provides a lot of resources and flexibility at a relatively lesser cost.

Further, since the algorithm is simple and efficient the design and VLSI implementation of a CORDIC based processor is easily achievable.In this project a CORDIC module is designed and simulated using Xilinx ISE using Verilog as a synthesis tool. The output of the CORDIC core is analyzed and verified on the test-bench. The device utilization summary showed that minimum resources were consumed.

REFERENCES:

[1] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Trans. Electronic Computing, vol. EC-8, pp 330 – 334, 1959.

[2] J.S. Walther, “A unified algorithm for elementary functions,” Proc. Spring. Joint Comp. Conf., vol. 38, pp. 379-385, 1971.

[3] Y. Hu, “CORDIC-based VLSI architectures for digital signal processing,” IEEE Signal Proc. Mag., pp. 16-35, 1992.

[4] B. Lakshmi and. A. S. Dhar, “CORDIC Architectures: A Survey,” Hindawi Publishing Corporation vol. 2010, Article ID 794891.

[5] R.Andraka, “A survey of CORDIC algorithms for FPGA based computers,” FPGA ’98, in ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp 191-200, 1998.

[6] R. Woods, J. McAllister, G. Lightbody and Y. Yi, FPGA-based Implementation of Signal Processing Systems, Wiley, 2008.

[7] K.D.Underwood and K.S.Hemmert, “Closing the Gap: CPU and FPGA Trends in Sustainable Floating-Point BLAS Performance,” presented at International Symposium on Field-Programmable Custom Computing Machines, California, USA, 2004.

[8] L.Zhuo and V.K.Prasanna, “Sparse Matrix-Vector Multiplication on FPGAs,” presented at International Symposium on Field Programmable Gate Arrays (FPGA), Monterey, CA, 2005.

[9] Y.Meng, A.P.Brown, R.A.Iltis, T.Sherwood, H.Lee, and R.Kastner, “MP Core: Algorithm and Design Techniques for Efficient Channel Estimation in Wireless Applications,” presented at Design Automation Conference (DAC), Anaheim, CA, 2005.