A High Speed 32-bit FPGA based CORDIC Architecture for Sine and Cosine Function Evaluation

Burhan Khurshid
Department of Computer Science Engineering,
NIT Srinagar, India

Roohie Naz Mir
Department of Computer Science Engineering,
NIT Srinagar, India

Hakim Najeeb-ud-din Shah
Department of Electronics and Communication Engineering,
NIT Srinagar India

1. ABSTRACT

Digital Signal Processing (DSP) algorithms always have a need for calculating certain linear, trigonometric, hyperbolic, logarithmic and other transcendental functions. CORDIC based algorithms have long been used in evaluating these functions. Traditional approaches have, however, been limited to software domain only. The simplicity of operation of CORDIC algorithm encourages its implementation in hardware. In this paper a novel CORDIC architecture for sine and cosine function evaluation has been proposed. The hardware integration is carried out using Field Programmable Gate Arrays (FPGAs).The proposed algorithm is based on modified carry save addition and incorporates bit-truncation. The structure offers extremely low latency and high operating frequencies, when pipelined. The novelty of the proposed architecture is that it offers a flat timing response for varying input word lengths. The structure has an inherent capability of supporting an additional internal pipeline within each stage, enabling the structure to operate at high frequencies, typically four times that of the normal CORDIC. The performance analysis is carried out by comparing the proposed architecture against existing non-redundant (basic) and redundant (modified) architectures.

General Terms

DSP algorithm, Reconfigurable computing

Keywords

Carry save Addition; CORDIC Algorithm; Digital Signal Processing; FPGA; Pipelining; Rotation Mode

2. INTRODUCTION

CORDIC (COordinate Rotation DIgital Computer) [1,2] is a shift-and-add algorithm that is widely used in Very Large Scale Integration (VLSI) DSP systems [3] due to its simple hardware and versatility. The algorithm is the best compromise between the look up table method (requires an enormous memory) and the polynomial approximation method, (slow to converge). Traditionally the hardware implementation has foccussed on sequential structures. However, these structures do not map well on FPGAs [4]. The major drawback of the conventional CORDIC algorithm was its relatively high latency and low throughput [5]. This was due to the sequential nature of the iteration process with carry propagate addition and variable shifting in every iteration [6]. Unfolded parallel implementations were proposed to overcome the above drawbacks. However, the carry propagate addition still remained a bottleneck for further latency improvements [7, 8]. To improve the latency of CORDIC architectures, high radix CORDIC and redundant signed arithmetic have been proposed and implemented. The use of redundant signed arithmetic prevents the propagation of carry thereby making it possible to create parallel adders with constant delay irrespective of the operand word-length [9]. Thus low latency results are produced in a redundant architecture resulting in high-performance designs.

Carry save arithmetic also leads to a form of redundant number representation resulting in low latency operations.This paper proposes a novel architecture that, besides reducing the latency of CORDIC algorithm stabilizes its timing response by providing a constant operating frequency for varying input word lengths.

The use of FPGAs as an implementing platform for various custom DSPs has increased since the beginning of this decade [10, 11]. However, earlier FPGAs were not able to implement the parallel CORDIC architecture due to the limited chip size and difficulty in routing the hard-wired shifters [4, 12, 13]. Consequently FPGAs were used to implement only the iterative CORDIC architectures [14, 15]. The implementation of parallel CORDIC architectures was, therefore restricted only to the programmable platforms.

However, the advancement in VLSI technology has resulted in high density FPGAs which provide an attractive platform for parallel CORDIC architectures [16, 17, 18, 19]. This paper emphasizes the implementation of parallel CORDIC architectures in FPGAs.

3. CORDIC ALGORITHM

The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shift and add operations. The algorithm, credited to Volder, is derived by rotating a vector (x, y) in a Cartesian plane by some angle, say ϕ.

For a single CORDIC micro-rotation the resulting equations are:

$$x_{i+1} = x_i - y_i d_i 2^{-i} \tag{1}$$

$$y_{i+1} = y_i + x_i d_i 2^{-i} \tag{2}$$

$$z_{i+1} = z_i - d_i \tan^{-1}\left(2^{-i}\right) \tag{3}$$

Where,

$$d_i = -1 \qquad if \;\; z_i < 0$$
$$=+1 \qquad Otherwise$$

Equations 1 and 2 are the coordinate equations, giving the coordinates for the next micro-rotation. Equation 3 is the angle equation, giving the amount of rotation for the next micro-rotation.

The CORDIC rotator is normally operated in one of two modes. In rotation mode, the angle accumulator is initialized with the desired rotation angle. The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator. The decision at each iteration is therefore based on the sign of the residual angle after each step.

After n iterations we get the following results:

$$x_n = A_n \left[x_0 \cos z_0 - y_0 \sin z_0 \right] \tag{4}$$

$$y_n = A_n \left[y_0 \cos z_0 - x_0 \sin z_0 \right] \tag{5}$$

$$z_n = 0 \tag{6}$$

Setting the *y* component of the input vector to zero reduces the rotation mode result as under:

$$x_n = A_n . x_0 \cos z_0 \tag{7}$$

$$y_n = A_n . x_0 \cos z_0 \tag{8}$$

By setting x_0 equal to $1/A_n$, the rotation produces the unscaled sine and cosine of the angle argument $z_{0.}$

4. CARRY SAVE ADDITION

Carry-save arithmetic leads to a form of redundant number representation. Due to its similarity with redundant arithmetic, most redundant architectures can be adapted to carry-save case. In carry save addition, we refrain from directly passing on the carry information until the very last step. This is done by taking three numbers, *x, y, z* and converting them into two numbers *c* and *s* such that:

$$x + y + z = c + s \tag{9}$$

Where, ***s*** represents the sum and ***c*** represents the carry.

The important consideration is that ***c*** and ***s*** can be computed independently, and furthermore, each c_i (and s_i) can also be computed independently from all of the other ***c***'s (and ***s***').

In order to add m different n-bit numbers together, the simple approach would be to repeat the procedure approximately m times over. This would require (m−2) carry save adder (CSA) blocks and a final ripple carry adder (RCA) block. Note that every time we pass through a CSA block, our number increases in size by one bit. Therefore, the number that goes to the RCA will be at most (n + m – 2) bits long. So the final RCA will have a gate delay of O(log (n + m)). Therefore the total gate delay is O(m + log (n + m)). Two points worth noting here are; firstly the major part of the delay in the structure is due to the final RCA stage; and, secondly after every CSA stage an additional bit gets added to the sum vector. The proposed architecture takes care of both these issues.

5. PROPOSED ARCHITECTURE

The proposed architecture is based on carry save addition. A carry save adder consists of a carry save stage and a final ripple carry stage that adds (or subtracts) the sum (or difference) and the carry (or borrow) bits [20]. The performance of the adder is limited by the ripple carry stage that accounts for the major portion of the delay in the circuit. Unfolded implementation of CORDIC structures is carried out by decomposing the entire core into several subsequent stages, with the output of one stage serving as input to the subsequent one. At each stage a sum (or difference) vector is obtained that is propagated to the next stage. The proposed architecture eliminates the ripple carry stage from the structure, such that instead of propagating only the final sum (or difference) vector we are actually propagating both the sum (or difference) and carry (or borrow) vectors. This takes care of the large delay introduced by the RCA block at each stage. However, this results in certain modifications in the overall algorithm which is depicted in the flow chart of figure 1. The block diagram of the proposed CORDIC follows and is shown in figure 3.

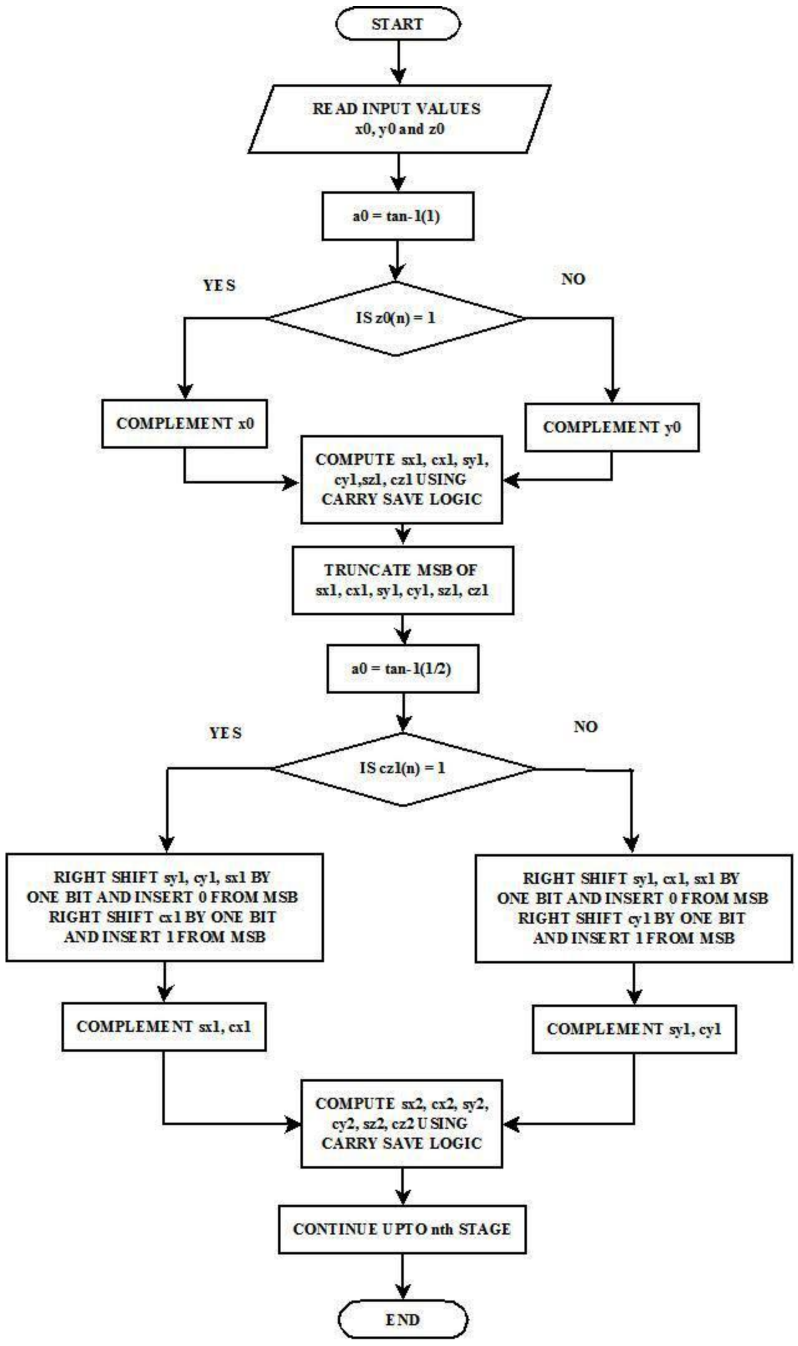
START
READ INPUT VALUES x0, y0 and z0
a0 = tan-1(1)
YES
NO
IS z0(n) = 1
COMPLEMENT x0
COMPLEMENT y0
COMPUTE sx1, cx1, sy1, cy1,sz1, cz1 USING CARRY SAVE LOGIC
TRUNCATE MSB OF sx1, cx1, sy1, cy1, sz1, cz1
a0 = tan-1(1/2)
YES
NO
IS cz1(n) = 1
RIGHT SHIFT sy1, cy1, sx1 BY ONE BIT AND INSERT 0 FROM MSB RIGHT SHIFT cx1 BY ONE BIT AND INSERT 1 FROM MSB
RIGHT SHIFT sy1, cx1, sx1 BY ONE BIT AND INSERT 0 FROM MSB RIGHT SHIFT cy1 BY ONE BIT AND INSERT 1 FROM MSB
COMPLEMENT sx1, cx1
COMPLEMENT sy1, cy1
COMPUTE sx2, cx2, sy2, cy2, sz2, cz2 USING CARRY SAVE LOGIC
CONTINUE UPTO nth STAGE
END

Fig 1: Flow chart for proposed CORDIC.

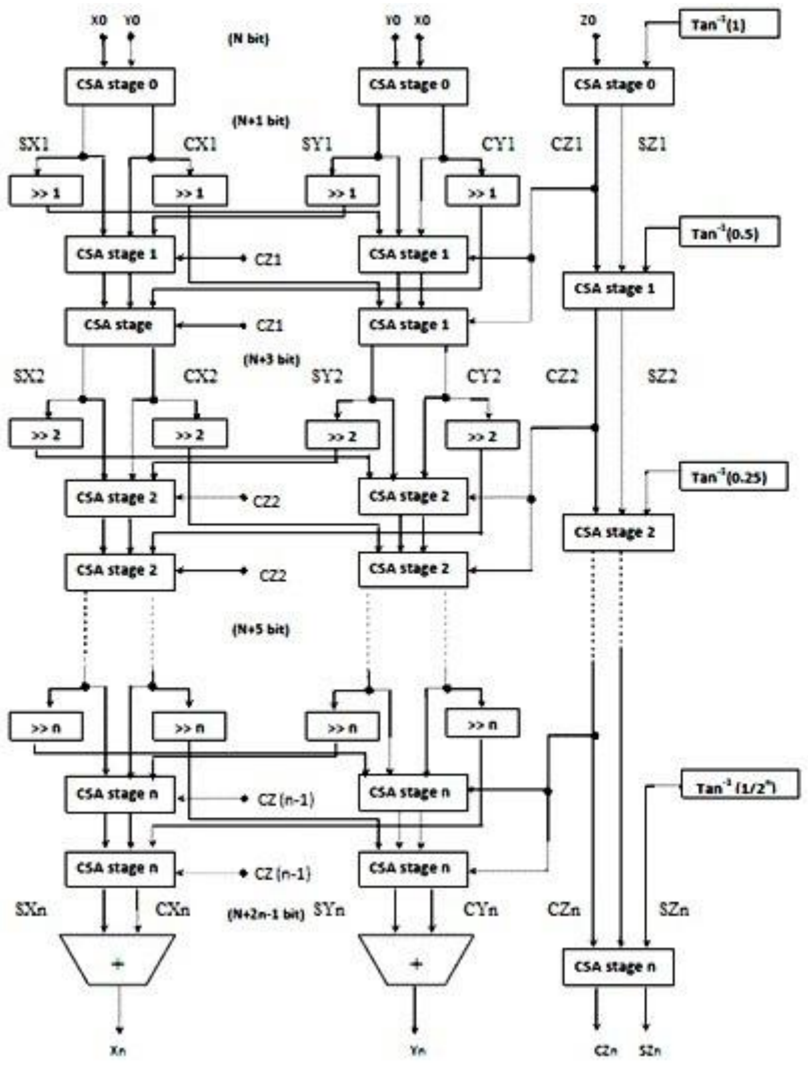

Fig 2: Proposed CORDIC logic diagram

As can be seen in figure 2 above, an additional redundant bit is added after every CSA stage, such that, if the initial word length is N bits, then after n stages the word length will be N+2n -1.This causes a high fan-in for the subsequent stages requiring several layers of logic, thus slowing the process. This has been overcome by truncating the most significant bit after every carry save addition/subtraction. The truncation of the most significant bit does not affect the precision of the result as the bit is redundant. The bit truncated CORDIC is shown in figure 3

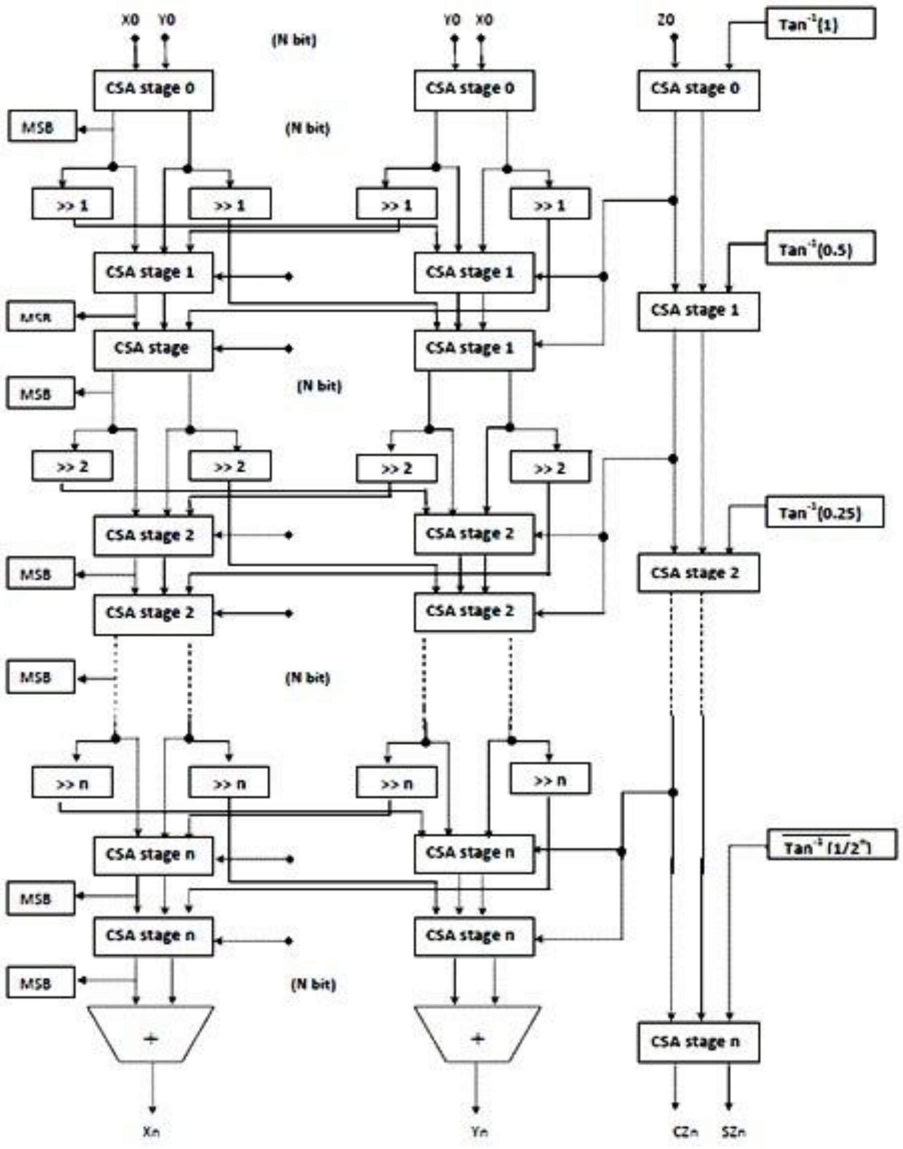

Fig 3: Bit truncated proposed CORDIC logic diagram

As seen in the above figure in each stage a redundant bit is generated. This MSB is eliminated in each stage, so that if the initial wordlength is N-bits the wordlength after n stages will remain N-bits.

6. SYNTHESIS AND SIMULATION

6.1 Methodology

The proposed CORDIC is implemented in seven stages for a word length of 32 bits. The initial design entry is done using VHDL. The design translation is carried out and the simulator database is then analyzed for different performance parameters and logical conclusions are drawn on the basis of different parameter values.

6.2 Implementation

The core is implemented with the following synthesis description:

Platform: FPGA
Family: Virtex5
Target device: XC5VLX30
Package: FF324
Speed grade: -3

Figure 4 shows the block view of the proposed CORDIC structure and figure 5 shows the RTL view of a single stage.

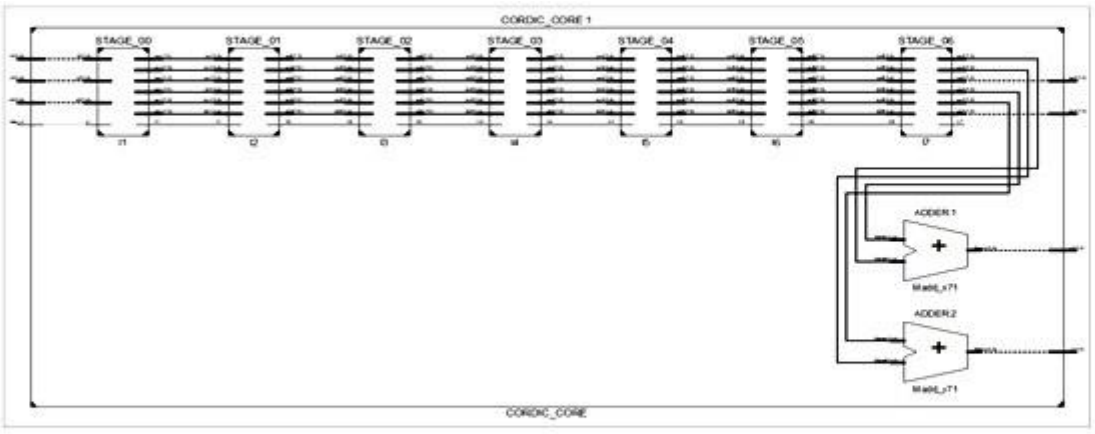

Fig 4: Block view of proposed CORDIC

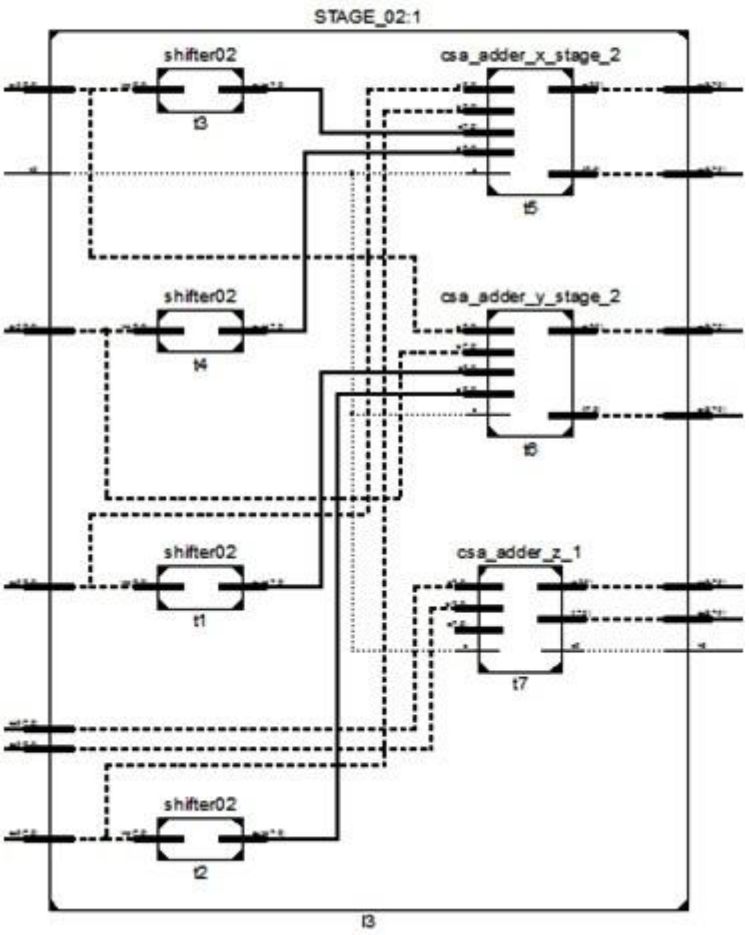

Fig 5: Single stage RTL view

The generated core has been simulated for sine and cosine functions by operating it in the rotation mode and results were calculated for different angles. The proposed CORDIC is pipelined by inserting registers in between the individual stages. This increases the maximum operating frequency resulting in higher throughputs. The structure provides an inherent capability of introducing an additional pipeline register within each stage. This further increases the throughput of the system. Figure 6 shows the proposed CORDIC with conventional and internal pipeline.

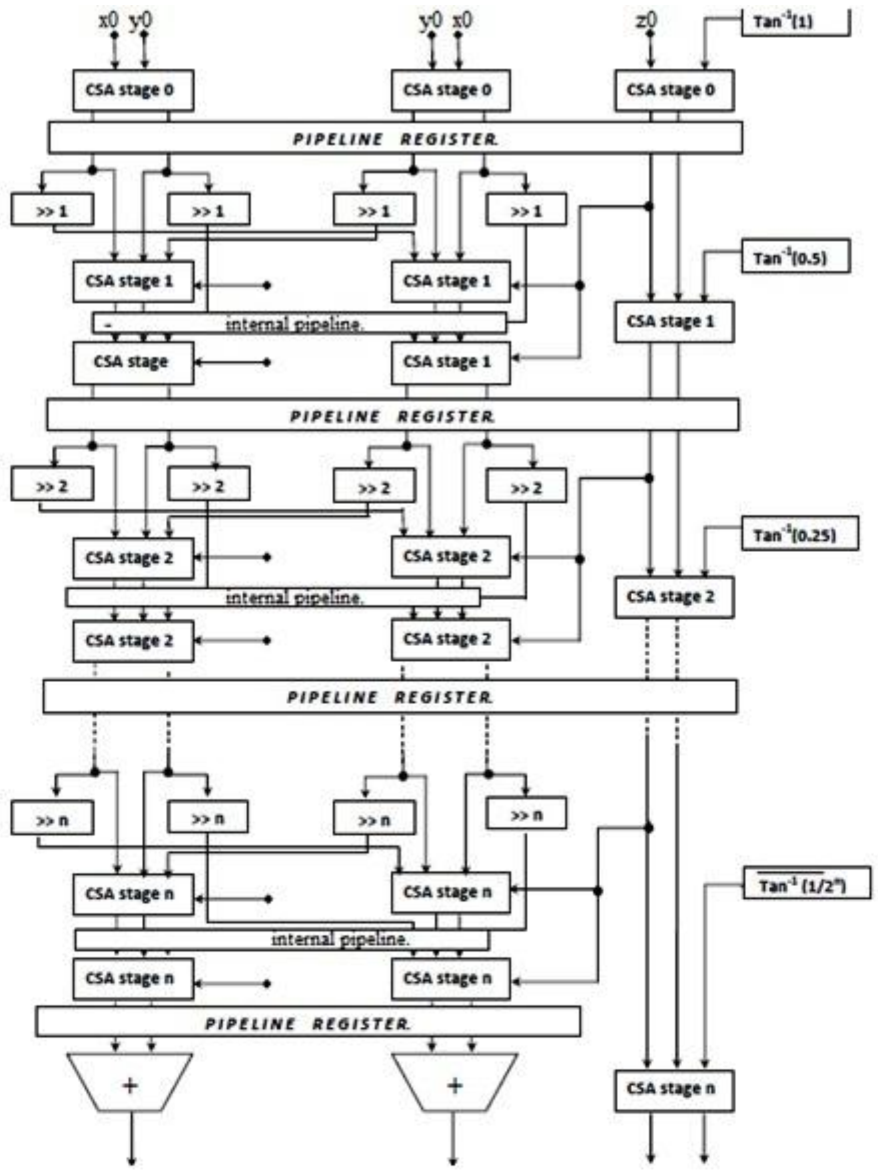

Fig 6: Proposed CORDIC with conventional aon internal pipelining

7. EXPERIMENTAL AND SIMULATION RESULTS

The proposed structure is analysed for different performance parameters. Table 1 provides latency and throughput comparisons of the non-redundant (basic), redundant signed (modified) and proposed CORDIC structures. All the structures are implemented with the same synthesis description.

Further analysis of the proposed CORDIC is carried out by varying the input word length from 16 bits to 128 bits. It has been observed that the proposed CORDIC offers a flat latency and throughput response. As mentioned earlier the proposed structure has an inherent capability of being pipelined within each stage. This increases the overall throughput as depicted in figure 7 and figure 8. Finally the latency and throughput variations of the proposed architecture are compared against the non-redundant (basic) and redundant signed (modified) CORDIC. The comparison results are graphically shown in figures 9 and 10.

Table 1. Latency and Throughput comparison for different 32-bit CORDIC structures

Parameter	CORDIC architectures		
	Basic	Modifed	Proposed
Logic delay (ns)	9.18	13.589	5.034
Route delay (ns)	25.002	6.448	9.237
Max. Combinational delay (ns)	34.182	20.037	14.271
Throughput. (MHz)	31.667	60.85	70.07
Throughput (pipelined). (MHz)	120.841	423.276	905.469

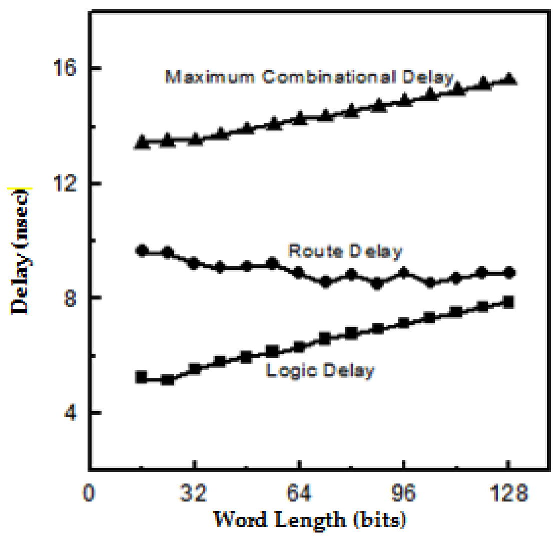

Fig 7: Delay variations in proposed CORDIC

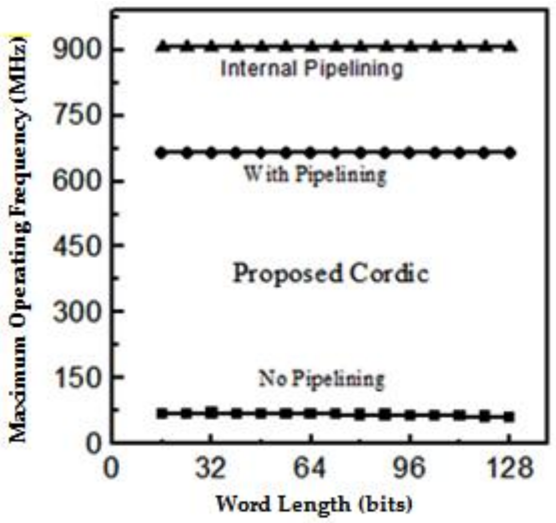

Fig 8: Throughput variations in proposed CORDIC

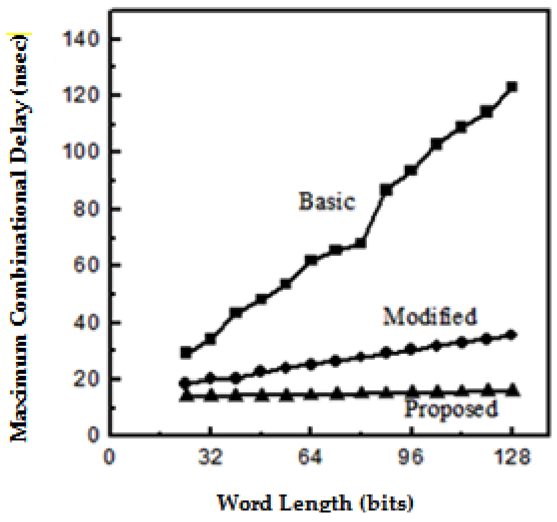

Fig 9: Worst case delay comparison for basic, modified and proposed CORDIC

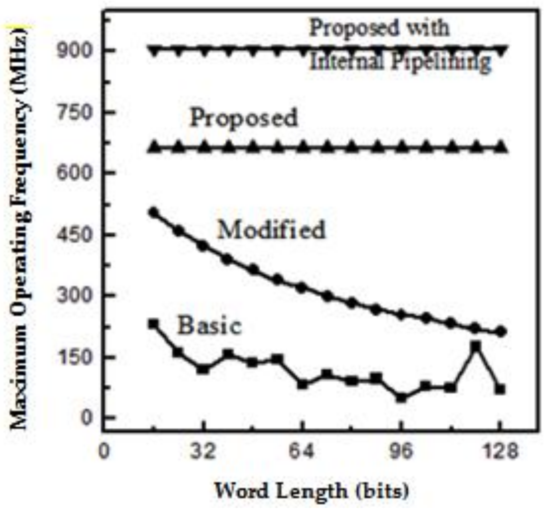

Fig 10: Throughput comparison for basic, modified and proposed CORDIC

Further analysis of proposed CORDIC is carried out by comparing the power consumption for 32 bit word length. Table 2 gives the power consumption of the three structures.

Table 2. Power dissipation for redundant, non-redundant and proposed CORDIC

Parameter	Power Dissipated (mW)		
	Basic	Modified	Proposed
Clock	15.53	10.36	32.17
Logic	9.23	6.55	22.00
Signals	10.59	5.79	18.55
IOs	196.69	110.73	200.76
Leakage	380.12	379.75	380.53
Dynamic	232.03	133.44	273.49
Total	612.15	512.58	654.02

From table 2 it is observed that the proposed structure dissipates more power than the non-redundant and redundant signed structures. This is due to the high operating frequency of the proposed CORDIC which demands high clocking rates and thus greater power dissipation. Apart from it, the proposed CORDIC requires more hardware resources resulting in larger on chip power dissipation. The device utilization for the three structures is summarized in table 3 for 32 bit word length.

Table 3. Device utilization for redundant. non-redundant and proposed CORDIC

Parameter	CORDIC Architectures		
	Basic	**Modified**	**Proposed**
No. of Registers	678	678	2079
No. of Look-Up-Table (LUTs)	1006	685	2091
No. used as logic	1006	685	2089
No. of occupied Slices	336	181	593
No. of LUT Flip Flop pairs used	1013	685	2143
No. of bonded Input-Output-Buffers (IOBs)	194	194	226

7.1 Discussions

It is observed that when timing response of the CORDIC structures is concerned, the proposed CORDIC is suitable for practical implementation. The critical path delays in case of the proposed CORDIC is much lesser and is almost constant for input word lengths varying from 16 to 128 bits. Critical path delays determine the clock period and thus the operating frequency of the core [21]. Owing to a constant critical path delay the operating frequency of the core remains constant irrespective of the input word length. When the core is pipelined it results in an increased operating frequency that remains constant for the entire range of input word lengths. In effect the core offers a flat timing response for the entire range of input word lengths, which is much higher than the basic CORDIC.

The flat timing response of the proposed CORDIC can be explained as under: the decomposition of a given function into sub functions and the routing of the interconnections between them yield a considerable uncertainty in the propagation delay from the input to the output of an implemented circuit. Since most combinational circuits are placed in a sequential environment, there is usually an interest in the worst case delay which is determined by adding up the maximum expected delays of the sub functions into which a given function is decomposed. CORDIC represents a typical example of such a structure wherein the entire core is implemented by decomposing it into subsequent stages (sub functions). There is, however, always an uncertainty in determining the worst case delay and thus the critical path of a given function from its sub functions. These uncertainties are due to the decomposition of a function into sub functions, implementation of these sub functions and the interconnections within an FPGA. The exact worst case delay can thus be known only after the implementation process has been completed including the decomposition into sub functions and interconnect routing.

The adder/subtractor used in case of the non-redundant CORDIC is based on ripple carry logic, wherein the carry has to propagate from the LSB to MSB. Thus, as the word length increases the uncertainties in the decomposition process and the implementation of the sub functions becomes large. This results in different critical paths for different stages and thus varying clock periods. The overall operating frequency of the core is thus determined by the slowest stage within a structure and is thus variable. In contrast to this, the adder logic in case of the proposed CORDIC is much simpler, requiring only bit-by-bit XOR, AND, and OR operations (which are carried out in parallel) so that the uncertainty in the decomposition and implementation process is almost negligible resulting in a flat timing response. Also the adder logic is simplified by eliminating the ripple carry stage in each carry save adder.

Another way of looking at it is that the conventional non-redundant and signed structures depend on the input word as a whole. As the word length increases the corresponding combinations also increase and the randomness or uncertainties associated with these combinations tend to get complex. This results in complex critical paths after implementation and thus random delays. In contrast to this, the addition operation in case of proposed CORDIC depends on individual bits and not on the word as a whole. Since a bit can have only two logic values (0 and 1) the corresponding combinations and the associated uncertainties are very simple and predictable. This results in simpler critical paths and thus faster structures.

8. CONCLUSION

This paper presented a novel approach for implementing parallel CORDIC algorithm. The implementation was focussed for FPGA platforms. The proposed architecture is based on carry save addition and has been shown to offer minimum latency and high operating frequency. More importantly the proposed structure offers a flat throughput response for varying word lengths, which is an improvement over the previous designs. This is, however achieved at the cost of more power dissipation and area usage. However, since the implementation is targeted for FPGA devices, area is not a major concern. The proposed architecture thus presents a high throughput solution that is demanded by present day high speed large word length DSP applications.

9. ACKNOWLEDGMENTS

This work has been carried out in smdp-ii vlsi laboratory of the electronics and communication engineering department, of national institute of technology srinagar, india. This smdp – ii vlsi project is funded by ministry of communication and information technology, government of india. Authors are grateful to the ministry for the facilities provided under the said project

10. REFERENCES

[1] Volder, J. E. The CORDIC trigonometric computing technique, in IRE Transactions Electronic Computing, Vol EC-8, pp 330 – 334, 1959.

[2] Walther, J. S. A unified algorithm for elementary functions, in Proceedings Spring Joint Computer Conference, Vol. 38, pp. 379-385, 1971.

[3] Hu, Y. H. CORDIC-based VLSI architectures for digital signal processing, in IEEE Signal Proceedings Magazine, pp. 16-35, 1992.

[4] Andraka, R. A survey of CORDIC algorithms for FPGA based computers, FPGA98, ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp 191-200, 1998.

[5] Hu, Y. H. Pipelined CORDIC architecture for the implementation of rotational based algorithm, in Proceedings of the International Symposium on VLSI

Technology, Systems and Applications, p. 259, May 1985.

[6] Erecegovac, M. D. and LANG, T. Digital Arithmetic, Elsevier, Amsterdam, the Netherlands, 2004.

[7] Antelo, E., Villalba, J., Bruguera, J. D. and Zapata, E. L. High performance rotation architectures based on the Radix-4 CORDIC algorithm, IEEE Transactions on Computers, Vol. 46, no. 8, pp. 855–870, 1997.

[8] Ercegovac, M. D., and Lang, T. Fast cosine/sine implementation using on-line CORIC, Proceedings of the 21st Asilomar Conference on Signals, Systems, and Computers, 1987.

[9] Parhi, K. K. VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, 1999.

[10] Graumann, P. J., Turner, L. E. Implementing Digital Signal Processing Algorithms using Pipelined Bit-Serial Arithmetic and Field Programmable Gate Arrays, First International ACM/SIGDA Workshop on Field Programmable Gate Arrays (FPGA'92), 1992.

[11] Isoaho, J., Pasanen, J., Vainio, O., and Tenhunen, H. DSP System Integration and Prototyping with FPGAs, Journal of VLSI Signal Processing, Vol. 6, pp. 155 – 172, 1993.

[12] Wahab, M., Puckey, D. FPGA-based DSP Systems, Eds. W.R. Moore and W. Luk, Abindon EE&CS books, 1994.

[13] Petersen, R. J. and Hutchings, B. An Assessment of the Suitability of FPGAbased Systems for Use in DSPs, in Lecture Notes in Computer Science, no. 975, pp.293-302, Springer-Verlag, Berlin, 1995.

[14] Meyer-Base, U., Meyer-Base, A. and Hilberg, W. Coordinate Rotation Digital Computer (CORDIC) Synthesis for FPGA, in 41h International Workshop on Field Programmable Logic and Applications (FPL'94), 7-9 September 1994, Prag, Czech Republic.

[15] Dick, C. Computing the Discrete Fourier Transform on FPGA Based Systolic Arrays, ACM/SIGDA Int. symposium on Field Programmable Gate Array, pp. 129-135, Feb. 1996.

[16] Meyer-Base, U., Meyer-Base, A., Mellott, J. and Taylor, F. A fast modified CORDIC- Implementation of radial basis neural networks, Joumal of VU1 Signal Processing, Vol. 20, pp. 211-218, 1998.

[17] Mayosky, M. A., Battaiotto, P. E. and Toccaceli, G. M. A CORDIC Architecture for Vector Control, in Proceedings. of the Int. Con$ on Signal Processing Applications & Technology, 1998.

[18] Deprettere, E., Dewilde, P. and Udo, R. Pipelined CORDIC Architecture for Fast VLSI Filtering and Array Processing, in Proceedings of ICASSP 84, pp. 41.A.6.1 – 6.4, 1984.

[19] Khurshid, B., Rather, G. M. and Hakim, N. Performance Comparison of Non-redundant and Redundant FPGA based Unfolded CORDIC Architectures, in International Journal of Electronics and Communication Technology, Vol. 3, issue 1 pp 85-89, Marcch 2012.

[20] Deschamps, J. P., Bioul, G. J. A. and Sutter, G. D. Synthesis of Arithmetic Circuits, Wiley, 2006.

[21] Bhakthavatchalu, R., Sinith, M. S., Jismi, K. and Nair, P. A Comparison of Pipelined Parallel and Iterative CORDIC Design on FPGA, in Proceedings 5th International Conference on Industrial and Information Systems, (ICIIS 2010), pp. 239 – 243, July 29 - August 01, 2010, India.