merging and cannot be extended to a higher number of conventional CORDIC iterations since the induced error becomes unacceptable [5]. Parallel realization of CORDIC iterations to handle the first bottleneck by direct unfolding of micro-rotation is possible, but that would result in increase in computational complexity and the advantage of simplicity of CORDIC algorithm gets degraded [6]. Although no popular architectures are known to us for fully parallel implementation of CORDIC, different forms of pipelined implementation of CORDIC have however been proposed for improving the computational throughput [7].To handle latency bottlenecks, various architectures have been developed and reported in this review. Most of the well-known architectures could be grouped under bit parallel iterative CORDIC, bit parallel unrolled CORDIC, bit serial iterative CORDIC and pipelined CORDIC architecture which we discuss briefly in the following subsections.

A. Bit Parallel Iterative CORDIC Architecture

The vector Rotation CORDIC structure is represented by the schematics in Fig. 3. Each branch consists of an adder-subtractor combination, a shift unit and a register for buffering the output. At the beginning of a calculation initial values are fed into the register by the multiplexer where the MSB of the stored value in the z-branch determines the operation mode for the adder-subtractor. Signals in the x and y branch pass the shift units and are then added to or subtracted from the unshifted signal in the opposite path. The z branch arithmetically combines the registers values with the values taken from a lookup table (LUT) whose address is changed accordingly to the number of iteration. For n iterations the output is mapped back to the registers before initial values are fed in again and the final sine value can be accessed at the output. A simple finite-state machine is needed to control the multiplexers, the shift distance and the addressing of the constant values.

When implemented in an FPGA the initial values for the vector coordinates as well as the constant values in the LUT can be hardwired in a word wide manner. The adder and the subtractor component are carried out separately and a multiplexer controlled by the sign of the angle accumulator distinguishes between addition and subtraction by routing the signals as required. The shift operations as implemented change the shift distance with the number of iterations but those require a high fan in and reduce the maximum speed for the application. In addition the output rate is also limited by the fact that operations are performed iteratively and therefore the maximum output rate equals 1/n times the clock rate.

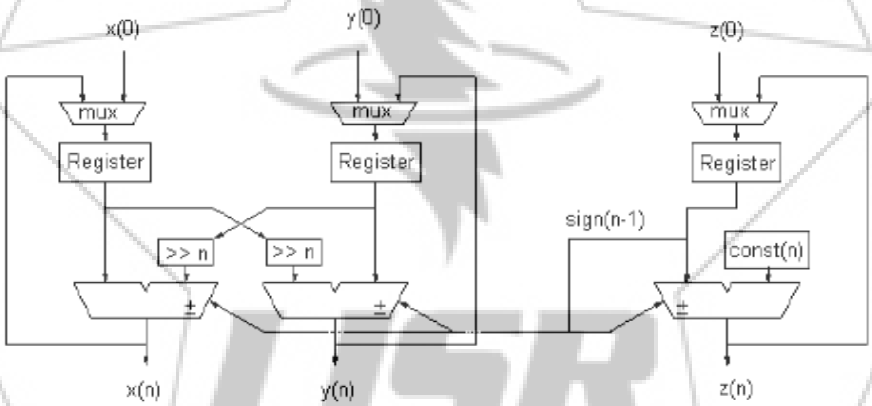


Figure 3: Iterative CORDIC

B. Bit Parallel Unrolled CORDIC Architecture

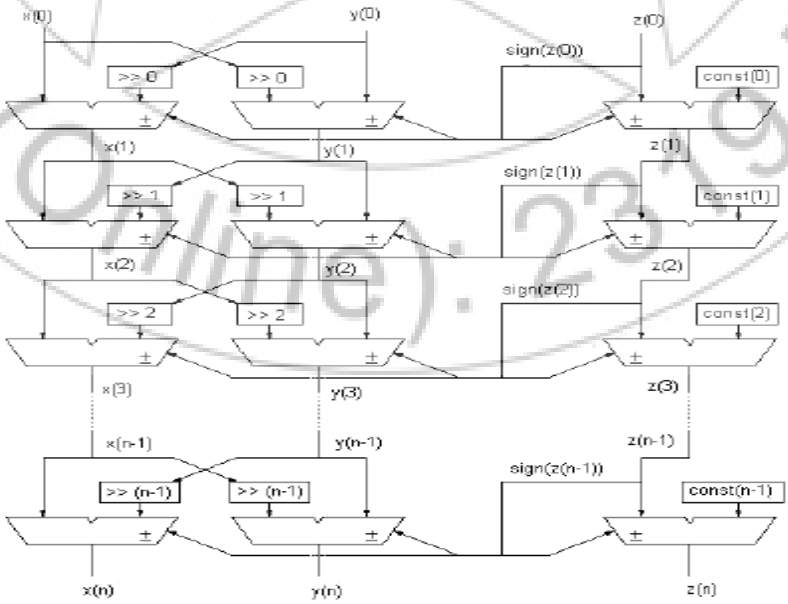


Figure 4: Unrolled CORDIC

Licensed Under Creative Commons Attribution CC BY

Instead of buffering the output of one iteration and using the same resources again, one could simply cascade the iterative CORDIC, which means rebuilding the basic CORDIC structure for each iteration. Consequently, the output of one stage is the input of the next one, as shown in Fig. 4, and in the face of separate stages two simplifications become possible. First, the shift operations for each step can be performed by wiring the connections between stages appropriately. Second, there is no need for changing constant values and those can therefore be hardwired as well. The purely unrolled design only consists of combinatorial components and computes one sine value per clock cycle. Input values find their path through the architecture on their own and do not need to be controlled. As we know, the area in FPGAs can be measured in CLBs, each of which consist of two lookup tables as well as storage cells with additional control components. For the purely combinatorial design the CLB's function generators perform the add and shift operations and no storage cells are used. This means registers could be inserted easily without significantly increasing the area. Pipelining ads some latency, of course, but the application needs to output values at 48 kHz and the latency for 14 iterations equals 312.5 which are known to be imperceptible. However, inserting registers between stages would also reduce the maximum path delays and correspondingly a higher maximum speed can be achieved

C. Bit Serial Iterative CORDIC Architecture

Both, the unrolled and the iterative bit-parallel designs, show disadvantages in terms of complexity and path delays going along with the large number of cross connections between single stages. To reduce this complexity one could change the design into a completely bit-serial iterative architecture. Bit-serial means only one bit is processed at a time and hence the cross connections become one bit-wide data paths. Clearly, the throughput becomes a function of In spite of this the output rate can be almost as high as achieved with the unrolled design. The reason is the structural simplicity of a bit-serial design and the correspondingly high clock rate achievable. Fig. 5 shows the basic architecture of the bit serial CORDIC processor.

Cock rate/Number of iterations *word length

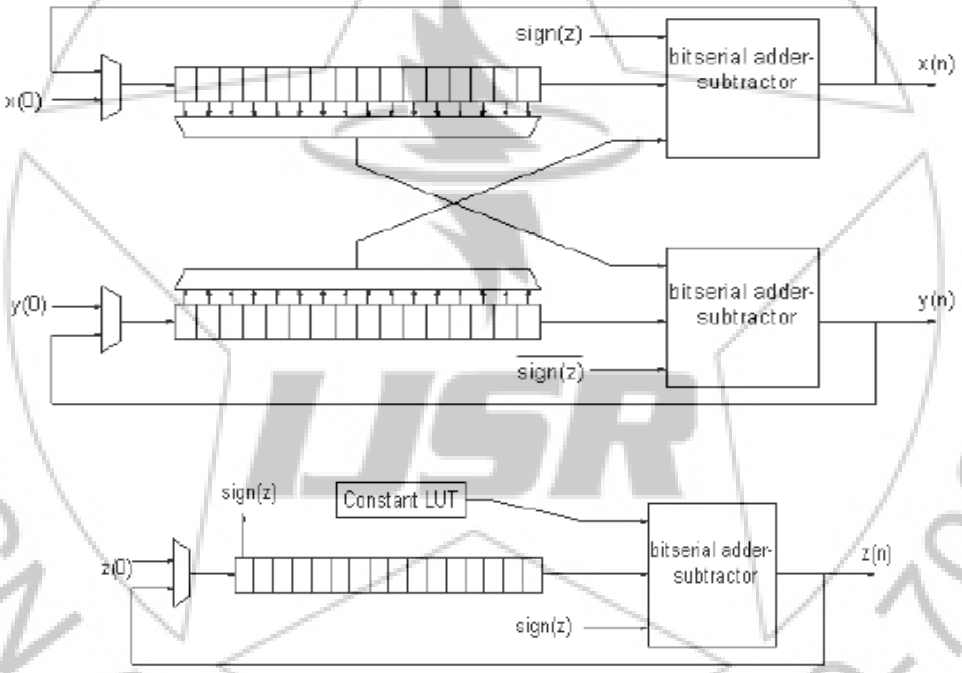


Figure 5: Bit-serial CORDIC

D. Pipelined CORDIC Architecture

Since the CORDIC iterations are identical, it is very much convenient to map them into pipelined architectures. The main emphasis in efficient pipelined implementation lies with the minimization of the critical path. The earliest pipelined architecture that we find was suggested in 1984. Pipelined CORDIC circuits have been used thereafter for high-throughput implementation of sinusoidal wave generation, fixed and adaptive filters, discrete orthogonal transforms and other signal processing applications [8].

3. MUX Based Cordic

The scheme for reducing the area of the CORDIC using multiplexer is proposed for the ASIC implementation in paper [3]. This is adopted for the FPGA based implementation in this paper. The area is reduced by removing some of the stages of Fig.1. The first stage outputof original unrolled CORDIC architecture is equal to xi, therefore we can directly write the output of first stage as

$$y1 = xi \text{----------} 9$$

$$x1 = xi \text{----------} 10$$

If the first stage output is positive, then

$$Y2=Y1-X1/2=Xi/2 \text{----------} 11$$

$$X2=X1+Y1/2=3*Xi/2 \text{----------} 12$$

The vector coordinates corresponding to negative output is

Licensed Under Creative Commons Attribution CC BY

Y2=Y1+X1/2=3*Xi/2----------13
X2=X1-Y1/2=Xi/2----------14

The output of the second stage is fixed. So we can implement the second stage using two Muxes and choosing select line as the MSB bit of the previous angle accumulator output. Fig.6 shows the circuit of second stage using Muxes with MSB select line.

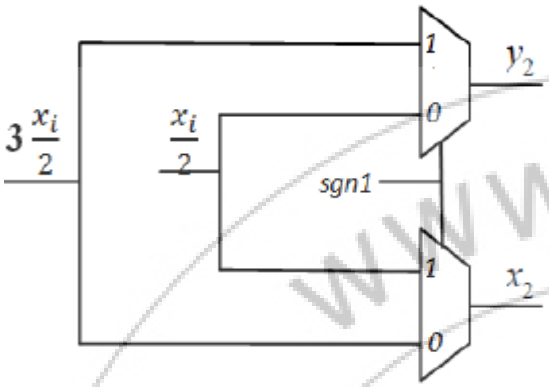
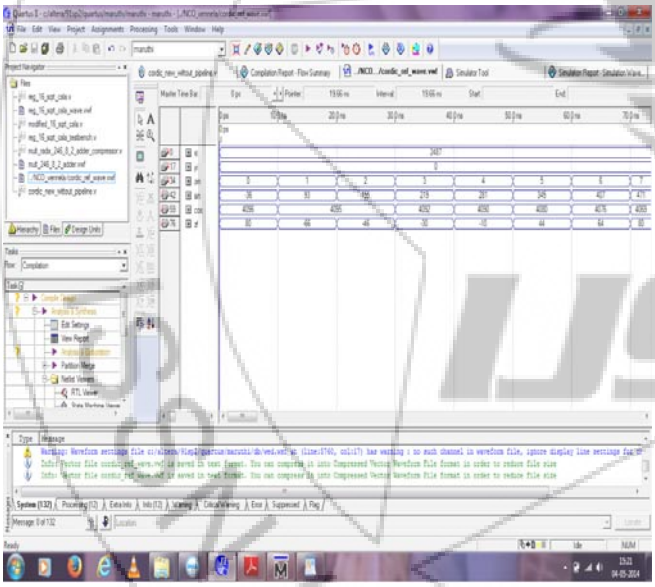


Figure 6

To reduce the area, we replace the third stage with Muxes. Since the third stage output also depends only on xi, we can express the outputs as



4. Conclusions

The design proposed in this paper for unrolled CORDIC is based on eliminating some of the adder stages by introduction of Muxes for area reduction. By implementation on Xilinx and Altera FPGAs, it is verified that the area reduction is achieved in these devices also in addition to the implementation on ASICs as reported in the literature.

5. Future Scope

Future scope of this paper can be done into the benefits that can be derived from using the consider unit in a neural networks. Practical neural network will help in understanding how large a network could be supported and which implementation best suited for networks of varying size.

References

[1] J.E. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic computer, vol. EC-8, pp. 330-334, 1959.
[2] J. Walther, "a unified algorithm for elementary functions," proc. Spring joint comp. con & vol.38, pp.379-385, 1971.
[3] Peter Nilsson, "complexity reduction in unrolled CORDIC architectures "Electronics, circuits, and systems,2009.ICECS 2009, pp.868-871.
[4] Vankka, J.; Kosunen, M.; Hubach, J.; Halonen, K.; , "A CORDICbased multicarrier QAM modulator," Global Telecommunications Conference,1999.GLOBECOM'99,vol.1A,no.,pp. 173-177vol.1a,1999 .
[5] Chen, A.; McDanell, R.; Boytim, M.; Pogue, R.;, "Modified CORDIC demodulator implementation for digital IF-sampled receiver," Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE , vol.2, no., pp.1450-1454 vol.2, 14-16 Nov 1995.
[6] Deprettere, E.; Dewilde, P.; Udo, R.;, "Pipelined cordic architectures for fast VLSI filtering and array processing," Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84. , vol.9, no., pp. 250- 253, Mar 1984.
[7] D. S. Cochran, "Algorithms and accuracy in the HP-35," Hewlett-Packard Journal, pp. 1–11, June 1972.
[8] S. Wang, V. Piuri, and J. E. E. Swartzlander, "Hybrid CORDIC algorithms,"IEEE Transactions on Computers, volume 46, no. 11, pp. 1202–1207, November1997.
[9] S. Wang and E. E. Swartzlander, "Merged CORDIC algorithm," in IEEE International Symposium on Circuits Systems (ISCAS'95),1995, volume 3, pp.1988–1991.
[10] B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," Microelectronics Journal, volume 33, Pp.77–89, 2002.
[11] D. E. Metafas and C. E. Goutis, "A floating point pipeline CORDIC processor with extended operation set," in IEEE International Symposium on Circuits and Systems, ISCAS'91, June 1991, volume 5, pp. 3066–3069.

Author Profile

M. Madhava Rao received B.Tech. degree in Electronics and Communications Engineering from JNTU University, Hyderabad, in 2006, Currently doing M. Tech. in ML Engineering college, from JNTUK University, Kakinada.

G. Suresh received B.E. degree in Electronics and Communications Engineering from JNTU, received M.TECH. from JNTUH and currently he working as Professor & HOD, Dept of ECE in ML. Engineering College, S. KONDA, Prakasam (DT)

Licensed Under Creative Commons Attribution CC BY