J Sign Process Syst (2016) 84:225–236
DOI 10.1007/s11265-015-1047-8

Design of Complex Non-Linear Adaptive Equalizer in Mitigating Severe Intersymbol Interferences

A. Mandal[1] **· R. Mishra**[1]

Received: 9 April 2015 / Revised: 21 July 2015 / Accepted: 15 September 2015 / Published online: 30 September 2015

Abstract In this paper, we propose an efficient architectural design for Adaptive Decision Feedback Equalizer (ADFE) for high-speed reliable data transmission through wired as well as wireless communication channels to minimize Inter Symbol Interference (ISI). High hardware cost of the system makes an urge to redesign it in order to minimize overall circuit complexity and cost. Realization of CORDIC based pipelined ADFE architecture using reformulated LMS algorithm has not only reduced multipliers in its feed forward path, the flexibility and modularity of the components makes it possible for easy implementation also. The realization results area efficient, high speed and throughput with good convergence. The efficacy of the proposed equalizer is corroborated with MATLAB simulations for mitigation of severe channel distortion arise due to ISI in high speed communication systems.

Keywords Inter symbol interference · Adaptive decision feedback equalizer · CORDIC · LMS algorithm · Pipelined multiplier

1 Introduction

In digital communication, high operating speed of adaptive equalizer is being demanded to cater high speed reliable data transmission over wired or wireless channel. But the transmitted data are often distorted due to non-linearity of channel response. The amount of distortion also varies with respect to time-varying nature of the operating channel. Due to the deviation of channel frequency response, both tails of the transmitted pulse will interfere with the neighboring pulses creating an *Inter Symbol Interference* (ISI) which may lead to erroneous decision by escalating the probability of error in detection process. To overcome this problem an Adaptive Decision Feedback Equalizer (ADFE) at the receiver with a transfer function which is essentially inverse of the channel transfer function is capable to eliminate ISI.The performance of non-linear ADFE is better on a channel with spectral nulls in their frequency response characteristics than the linear equalizer which introduces significant amount of additive noise present in the received signal by introducing a large gain [1, 2].

In this paper, a pipelined CORDIC architecture has been used as a main processing element of the design. Since last few decades, a tremendous inclination among the researchers toward the implementation of CORDIC algorithm [3] in various applications of digital signal processing arena had been witnessed [4, 5]. The modularity, compatibility, pipelinability, numerical stability and efficiency in generation of trigonometric and hyperbolic function have made the CORDIC algorithm suitable for implementation of fast Fourier transform (FFT), discrete Fourier transform (DFT), adaptive lattice filter, and many more applications [6–8]. In this paper, an adaptive transversal filter has been designed with a pipelined CORDIC unit as a core processing element in the equalizer for filtering and filters weight updating. Unlike the conventional LMS algorithm, the rotational angles rather than the tap weights are updated directly. The most popular and highly robust Least Mean Squared (LMS) algorithm [9] has been efficiently mapped to obtain trigonometric form of LMS algorithm which facilitates in incorporating CORDIC processing element to

✉ A. Mandal
amritkar2k@gmail.com

R. Mishra
rmishra@gbu.ac.in

1 School of Information and Communication Technology, Gautam Buddha University, Yamuna Expressway, Greater Noida Uttar Pradesh-201308, India

participate in filtering as well as weight updating operations of the Feed Forward Filter (FFF) of the ADFE. Due to trivial bit length, *decision feedback loop* (DFL) has been simplified with general purpose multiplier to reduce complexity of the Feedback Filter (FBF). The proposed architecture uses CORDIC blocks instead of multipliers in the FFF and hence is more efficient in terms of internal numerical errors and power consumption.

2 Related Work

Adaptive Decision Feedback Equalizer (ADFE) is widely used equalization technique for radar, antenna beam forming, digital communication systems, wireless video telephony, magnetic storage and many more applications [10, 11]. Wide variants of ADFE are available in the literature [12–14]. However, pipeline design of ADFE is known to be a difficult task for high speed applications. It is well known fact that the clock rate is limited by DFL. The quantizer and adaptive feedback loop in ADFE makes it difficult for design of pipelined architecture. Therefore, previously published works mostly have used parallelism technique for its design [15, 16]. These architectures are inherited by incorrect initialization which leads to degraded performances along with huge hardware overhead as it transforms serial algorithm into equivalent pipelined algorithm. Shanbag et al. [17] and Yang et al. [18] has tried to reduce hardware overhead although it suffers from performance degradation in terms of output signal-to-noise ratio (SNR) and the rate of convergence.

VLSI implementation of ADFE as well as its performance in high-speed communication systems has become an ardent requirement. Reformulated trigonometric LMS (TLMS) algorithm based transversal filter using CORDIC processing unit has been proposed in [19]. Later on transform domain equalization technique using CORDIC was proposed [20]. In this paper, we have used two transversal filters namely feed forward filter (FFF) and feedback filter (FBF). In FFF CORDIC element has been used whereas, in FBF trivial multipliers have been used. Throughout the architecture, pipeline design technique has been used to reduce hardware complexity and to obtain high throughput for high speed applications. The modular design of multiplier has been incorporated to avoid place and route problem during physical layout. The combination of modified booth encoding (MBE) techniques [21, 22] along with partial product reduction schemes [23–25] have facilitated in reduction of overall latency of the design to a single adder which has been implemented with carry look-ahead adder (CLA). This kind of design not only facilitates in easy implementation on FPGA device, but also reduces complexity significantly as compared to multiplier and accumulator (MAC) based design. It has been observed that during digital conversion of input analog signals with limited number of bits of ADC, sampling at Nyquist rate disturbs the amplitude and phase response curve of equalizer. Again, finite word-length, round-off noise or overflow in MAC based transversal filter creates huge instability within the system. The effects of quantization noise continuously gets accumulated due to its feedback path resulting in severe instability. The CORDIC processing element has been used by optimizing the quantization error [26] and to facilitate easy implementation of the equalizer for digital signal processing application. The simulation results for convergence behavior under various step sizes and its ability to generate inverse transfer function in severe channel mismatched situation have been adequately analyzed.

The rest of the paper is organized as follows. In the next section, we review the simple LMS algorithm and subsequently mapped into trigonometric form. In section 4, CORDIC algorithm has been revisited and corresponding pipelined architecture is implemented. Section 5 deals with the design of pipelined Adaptive Decision Feedback Equalizer along with explanation of associated components used with the design. Performance analysis of proposed design and discussion part of the paper is accommodated section 6 and finally, conclusions are presented in section 7.

3 Trigonometric Mapping of LMS Algorithm

The LMS algorithm is one of the simplest as well as robust adaptive algorithms available in literature. We would like to reformulate LMS in trigonometric form for our design. Let the FIR filter is linear discrete time filter $x(n)$ and $d(n)$ are input and desired output sequence of the FIR filter (Fig. 1).

Let $\mathbf{w}$ and $\mathbf{x}(n)$ are filter coefficient and input vector of N-tap filter,

$$\mathbf{w} = [w_0, w_1, ..., w_{N-1}]^T \text{ and} \tag{1}$$

$$\mathbf{x}(n) = [x(n), x(n-1), ..., x(n-N+1)]^T$$

The output signal of the adaptive filter, $y(n)$, can be computed as the inner product of $\mathbf{w}(n)$ and $\mathbf{x}(n)$.

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n) \tag{2}$$

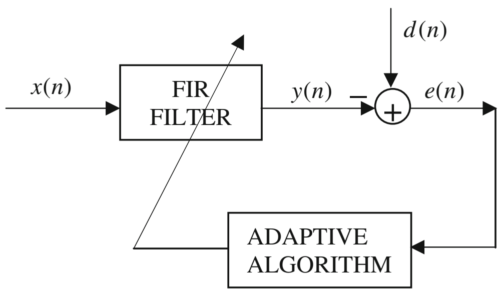


Figure 1 Adaptive LMS algorithm to update FIR filter weights.

The error signal $e(n)$ is expressed as the difference between the desired response $d(n)$ and the filter output $y(n)$, i.e.,

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n) \tag{3}$$

The weight vector is updated iteratively in such manner that the mean-square error (MSE) is minimized.

$$\varepsilon \equiv E\left[e^2(n)\right]. \tag{4}$$

$$= E\left[d^2(n)\right] - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \tag{5a}$$

$$= \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \tag{5b}$$

where $\mathbf{R} \equiv E[\mathbf{x}(n)\mathbf{x}^T(n)]$ is the input auto-correlation matrix and $\mathbf{p} \equiv E[\mathbf{x}(n)d(n)]$ is the cross-correlation vector.

Solving the Eq. (5b), the gradient first term is zero and overall $\nabla_w\varepsilon$ becomes

$$\nabla_w\varepsilon = -2\mathbf{p} + 2\mathbf{R}\mathbf{w} \tag{6}$$

Solving for $\nabla_w\varepsilon = 0$, the optimal weights of a FIR filter can be derived as

$$\mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{p}. \tag{7}$$

The straightforward application of $\mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{p}$ to calculate the optimum filter weight ($\mathbf{w}_{opt}$) is not of practical use. The computer burden corresponding to the matrix inversion can be evaluated from the number of complex multiplications, which is proportional to N^3. Hence, it is not compatible with the constraint of real-time operations. To avoid these complexities, steepest descent search method can be used for the design. The expression for steepest descent search method can be written for n-th index as

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{\mu}{2} \times \nabla_w\varepsilon \Big|_{w=w(n)} \tag{8}$$

Putting the value of **R** and **p** in Eq. (8), we get the simplest LMS algorithm in the following form:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\,\mathbf{x}(n)e(n) \tag{9}$$

The pipelined implementation of the above mentioned conventional LMS algorithm is difficult as it does not support pipelined implementation due to its recursive behavior. That is why delayed LMS (DLMS) algorithm has been used for easy implementation. If L numbers of pipelined stages are used in the design, the error signal $e(n)$ becomes available after adaptation delay for architecture of L cycles. The error in DLMS algorithm corresponds to $(\text{n-L})_{\text{th}}$ iteration for updating current filter weight in place of recent most error. Therefore, the iterative equation of delayed LMS algorithm can be expressed as:

$$\mathrm{w}(n+1) = \mathrm{w}(n) + \mu\mathrm{x}(n-L)e(n-L) \tag{10}$$

Since our proposed architecture uses CORDIC as a main processing element for weight updating, the iterative Eq. (9) is required modification in terms of angle. Let the tap weight w_k be

$$w_k = A_k\sin\theta_k \tag{11}$$

w_k satisfies, $-A_k \le w_k \le +A_k$, and maps uniquely to a θ_k in the interval $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$. The function $\sin\theta_k$ is a monotonically increasing continuous function of θ_k, as it lies within $[-\frac{\pi}{2} \le \theta_k \le +\frac{\pi}{2}]$, i.e. $\frac{\partial\varepsilon}{\partial\theta_k}$ has the same sign as that of $\frac{\partial\varepsilon}{\partial w_k}$ everywhere within the hypercube resulting in local minima or maxima less MSE. Using the Eq. (11), we can show the variation of error w.r.t angle variation as follows:

$$\frac{\partial\varepsilon}{\partial\theta_k} = \frac{\partial\varepsilon}{\partial w_k}A_k\cos\theta_k \tag{12}$$

Or

$$\nabla_\theta\varepsilon = \Delta.\nabla_w\varepsilon \tag{13}$$

Here Δ is $N\times N$ diagonal matrix whose k-th term can be given by:

$$\Delta_{k.k} = A_k\cos\theta_k, k = 0, 1, \ldots\ldots\ldots, N-1$$

Using the Eq. (6) and (13), we get the following

$$\boldsymbol{\theta}(i+1) = \boldsymbol{\theta}(i) - \frac{\mu}{2}\nabla_\theta\varepsilon\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(i)} \text{and} \nabla_\theta\varepsilon = -2\Delta(\mathbf{p} - \mathbf{R}\mathbf{w}) \tag{14}$$

With replacing **p** and **R** by $\mathbf{x}(n)d(n)$ and $\mathbf{x}(n)\mathbf{x}^T(n)$ in Eq. (14), we get the trigonometric form of LMS algorithm as

$$\begin{aligned} \boldsymbol{\theta}(n+1) &= \boldsymbol{\theta}(n) + \mu\boldsymbol{\Delta}(n)\mathbf{x}(n)e(n) \\ e(n) &= d(n) - \sum_{k=0}^{N-1}\sin\theta_k(n)x(n-k) \end{aligned} \tag{15}$$

But the input as well as desired signals to real communication channels is complex in nature and the filter tap weights are also complex. Therefore, we can write filter tap as:

$$w_k = w_{kr} + jw_{ki} \tag{16}$$

Where $w_{kr} = \sin\theta_{kr}$ and $w_{ki} = \sin\theta_{ki}$.

Again, Eq. (12) can be modified to

$$\frac{\partial\varepsilon}{\partial\theta_{kr}} = \frac{\partial\varepsilon}{\partial w_{kr}}\cos\theta_{kr} \text{ and } \frac{\partial\varepsilon}{\partial\theta_{ki}} = \frac{\partial\varepsilon}{\partial w_{ki}}\cos\theta_{ki} \tag{17}$$

The complex trigonometric form of LMS algorithm with real and imaginary form can be shown separately as angle update equation,

$$\boldsymbol{\theta}_r(i+1) = \boldsymbol{\theta}_r(i) + \mu.\Delta_r.\left[\mathbf{x}(n)e^*(n)\right]_r \tag{18}$$

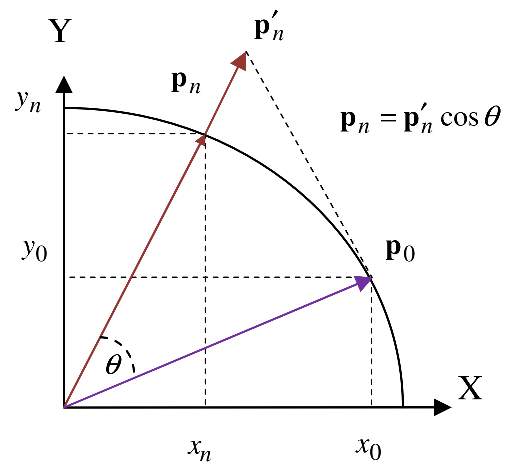


Figure 2 Rotation of a vector.

$$\theta_i(i+1) = \theta_i(i) + \mu.\Delta_i.\left[\mathbf{x}(n)e^*(n)\right]_i \tag{19}$$

where $e^*(n) = d(n) - \sum_{k=0}^{N-1} (\sin\theta_{kr} - j\sin\theta_{ki}) x(n-k)$.

The above form of LMS algorithm can be realized using CORDIC processing element. The sinusoidal term generated by CORDIC can be utilized for filtering as well as updating filter coefficients in the equalizer.

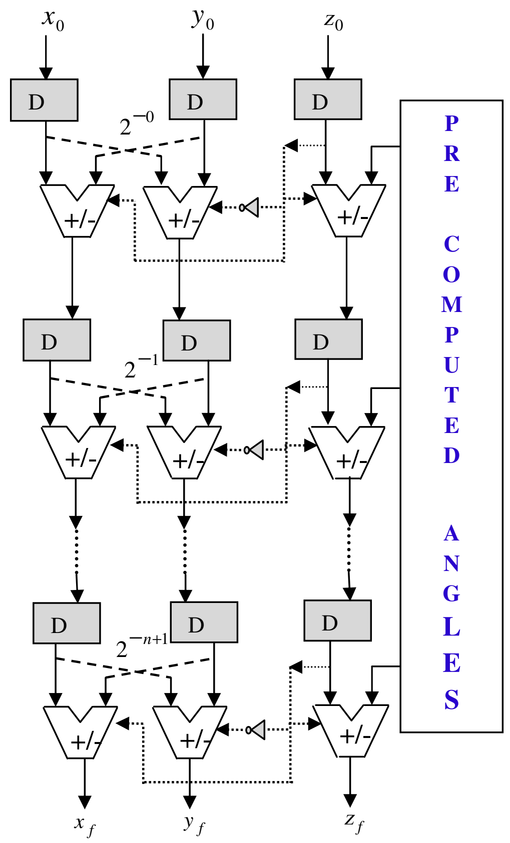


Figure 3 The pipelined CORDIC.

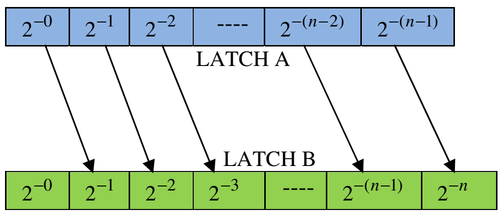


Figure 4 Oblique bus connection for shift operation.

4 CORDIC Processing Element

4.1 CORDIC Algorithm:

Rotation of a vector $\mathbf{p}_0=[x_0\, y_0]$ in 2-dimensional space (shown in Fig. 2) through an angle θ with n number of sequence of elementary rotations to get a final vector $\mathbf{p}_n=[x_n\ y_n]$ can be accomplished by using a corresponding matrix product $\mathbf{p}_n=\mathbf{R}_{rot}.\mathbf{p}_0$, where $\mathbf{R}_{rot}$ is represented as rotation matrix:

$$\mathbf{R}_{rot} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{20}$$

The rotation matrix $\mathbf{R}_{rot}$ in (20) can be written after factoring out the cosine term as:

$$\begin{aligned}\mathbf{R}_{rot} &= [\cos\theta] \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \\ &= \left[\left(1+\tan^2\theta\right)^{-1/2}\right] \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} = K\mathbf{R}_c\end{aligned} \tag{21}$$

where, $K=[(1+\tan^2\theta)^{-1/2}]$ is called scale-factor and $\mathbf{R}_c = \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix}$ is a pseudo-rotation matrix. During these elementary operations, the vector $\mathbf{p}_0=[x_0\ y_0]$ changes its magnitude only by amount of scale-factor, $K=[(1+\tan^2\theta)^{-1/2}]$, to obtain pseudo rotated vector

$$\mathbf{p}' = \mathbf{R}_c\mathbf{p}_0. \tag{22}$$

The CORDIC algorithm decomposes the desired rotation angle into the weighted sum of a set of consecutive micro

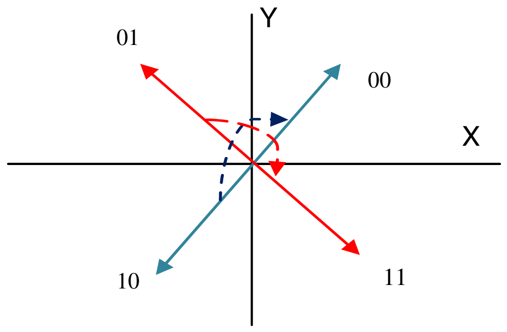


Figure 5 Quadrant transformation.

Table 1 Minimum clock period required by different CORDIC architectures [29].

Conventional CORDIC	$T_A+T_{FF}+5T_{MX}$
Fixed rotation CORDIC	$T_A+T_{FF}+5T_{MX}$
Interleaved scaling	$T_A+T_{FF}+5T_{MX}$
Separated scaling	$T_A+T_{FF}+4T_{MX}$
Bi-rotation cascade	$T_A+T_{FF}+2T_{MX}$
Proposed design	T_A

T_A, T_{FF} and T_{MX} stand for addition time, Flip-Flop delay and delays in 2:1 MUX respectively.

rotation and these micro rotations are based on sequence of elementary angles, α_i=arctan(2^{-i}).Therefore, tanα_i=2^{-i} can be easily implemented on hardware by right shifting through i. CORDIC achieves a desired rotational angle θ by performing a sequence of micro rotations through angle α_i which can be represented as:

$$\theta = \sum_{i=0}^{n-1} \delta_i \alpha_i \tag{23}$$

The rotation matrix at *i-th* iteration for a specific angle α_i is given by:

$$\mathbf{R}_{rot}(i) = K_i \begin{bmatrix} 1 & -\delta_i 2^{-i} \\ \delta_i 2^{-i} & 1 \end{bmatrix} \tag{24}$$

where scale-factor, $K = \overset{n-1}{\underset{i=0}{\Pi}} K_i = \overset{n-1}{\underset{i=0}{\Pi}} 1/\sqrt{\left(1+2^{-2i}\right)}$ and pseudo-rotation matrix

$$\mathbf{R}_c(i) = \begin{bmatrix} 1 & -\delta_i 2^{-i} \\ \delta_i 2^{-i} & 1 \end{bmatrix} \tag{25}$$

Since the scale-factor decreases monotonically irrespective of direction of micro rotation, it finally converges to ≅1.6467605. Therefore, it would be better to scale the final output by K instead of scaling during each micro rotation to maintain correct phase at the output. The scaling free CORDIC iterations are given by:

$$\mathbf{p}'_{i+1} = \mathbf{R}_c(i)\mathbf{p}_i \tag{26}$$

For specific angle α_i, the iterative equations can be shown as,

$$\begin{aligned} x_{i+1} &= x_i - \delta_i . 2^{-i} . y_i \\ y_{i+1} &= y_i + \delta_i . 2^{-i} . x_i \\ w_{i+1} &= w_i - \delta_i . \alpha_i \end{aligned} \tag{27}$$

4.2 Implementation of Pipelined CORDIC:

In Pipelined CORDIC architecture, a number of rotational modules are incorporated and each module is responsible for one elementary rotation. The modules are cascaded through intermediate latches (Fig. 3). Every stage within the pipelined CORDIC architecture, only adder/subtractor is used. The shift operations are hardwired using permanent oblique bus connections to perform multiplications by 2^{-i} as shown in Fig. 4. The pre-computed values of *i*-th iteration angle α_i required at each module can be generated through required sequence of binary data within V_{cc} and G_{nd} using trivial hardware arrangements. The delay is adjusted by using proper bit-length in the shift register. In the design of CORDIC module, adders and subtracters contribute the critical path. Since carry propagation delay in full adders is the source of delay, Carry Save Adder (CSA) [27, 28] was the obvious choice for the design. The use of these adders reduces the stage delay significantly. With the pipelining architecture, the propagation delay of the multiplier is the total delay of a single adder. So ultimately the throughput of the architecture is increased significantly as the throughput is given by: "*1*/(*delay due to a single adder*)". If an iterative implementation of the CORDIC is used, the processor would take several clock cycles to give output for a given input. But in the pipelined architecture, each pipeline stage takes exactly one clock cycle to pass one output.

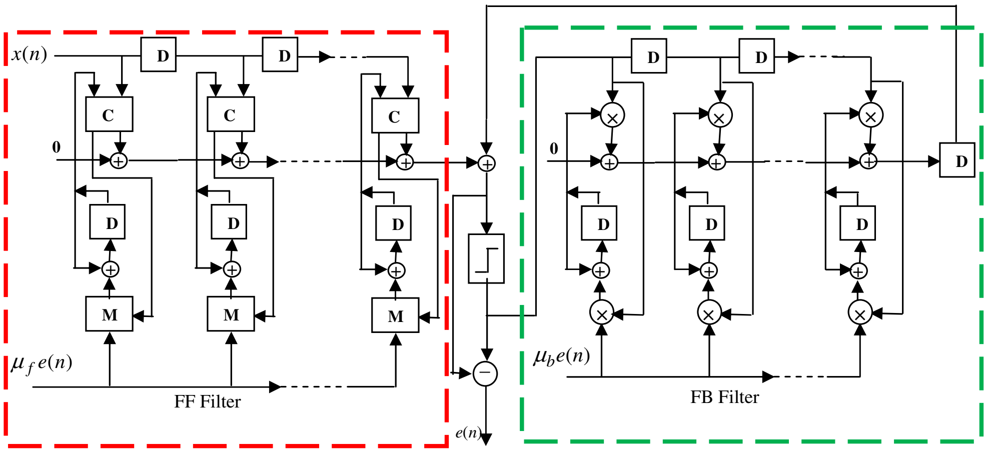
It is very important in our design to keep the rotation angles within $-\pi/2 \le \theta_i \le +\pi/2$ for utilization of CORDIC in design of equalizer in updating weights and filtering purpose utilizing

Table 2 Comparison with architectures based on synthesis result [29].

Designs	Area (sq. um)	Clock (ns)	TPT (per µs)	Latency (ns)	ACT (ns)	ADP
Conventional CORDIC	2987	2.52	36.07	27.72	27.72	82,799
Fixed rotation CORDIC	2471	2.49	40.16	24.90	24.90	61,527
Interleaved scaling	2674	2.56	55.8	17.92	17.92	47,918
Separated scaling	4074	2.29	109.17	16.03	9.16	37,317
Bi-rotation cascade	5819	2.21	226.24	15.47	4.42	25,719
Proposed design	3142	1.98	505.05	12.31	1.98	38,678

TPT stands for throughput, *ACT* stands for average computation time and *ADP* stand for area-delay product.

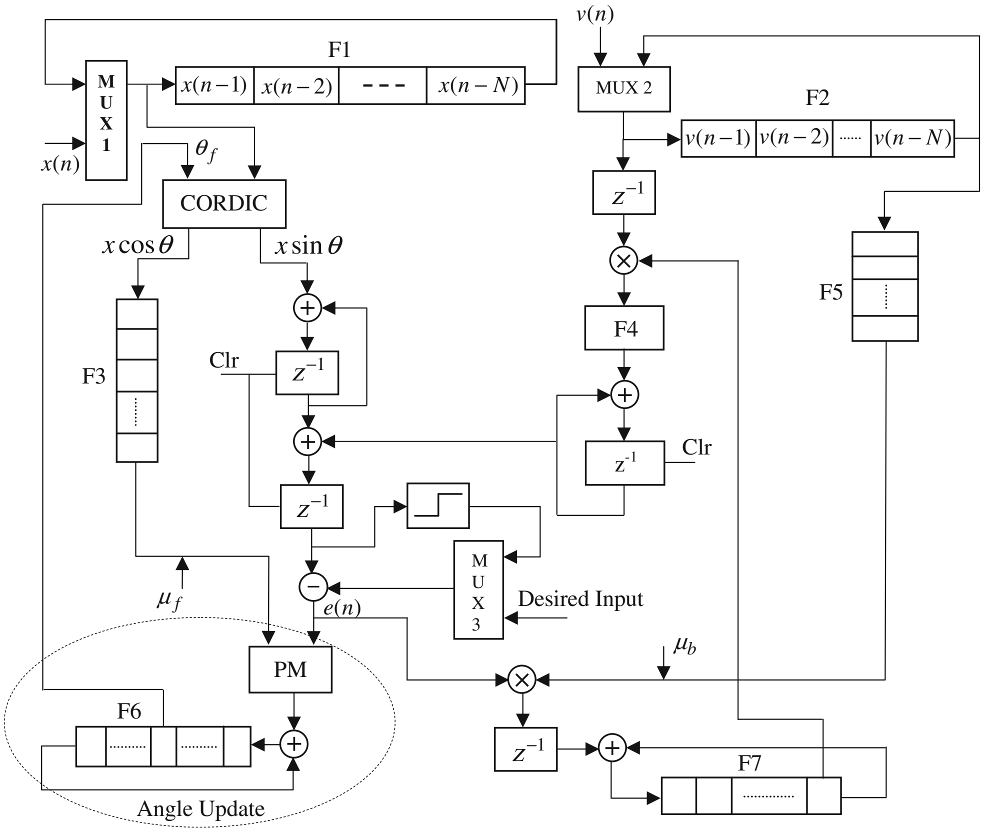
Figure 6 Block diagram of adaptive decision feedback equalizer with CORDIC as a central element in FFF.



sine/cosine data. The quadrants can be transformed by using first two MSB bits. If required rotation angle falls either in second quadrant ($\pi/2 \leq \theta_i \leq +\pi$) or in third quadrant ($-\pi \leq \theta_i \leq -\pi/2$), addition/subtraction of π with the θ_i will facilitate to transform to fall within fourth or first quadrant respectively through complementing corresponding MSBs. The transformation of angles in different quadrant has been illustrated in Fig. 5.

We now discuss the hardware and time complexities of the proposed design and our design can be compared with various types of CORDIC architectures including the basic one which is discussed in [30, Fig. 2]. Unlike the proposed design, the conventional and few other designs available in the literatures uses adders, resisters, barrel shifters and MUX. It has been seen that shifting operations using barrel-shifter for maximum of S shifts for word-length L can be implemented by $\lceil \log_2(S+1 \rceil$ stages of 2:1 MUX. Few designs have used ROM memory to store arctan angles as well as control signals for CORDIC operations. Therefore, the hardware complexity of the design also increases with increase in required word-length which leads to maximum number of shift operations for the design. We have compared our design with the other published work [29] in respect of clock period, throughput, latency and area which have been shown in Table 1 and Table 2 using TSMC 90-nm library.

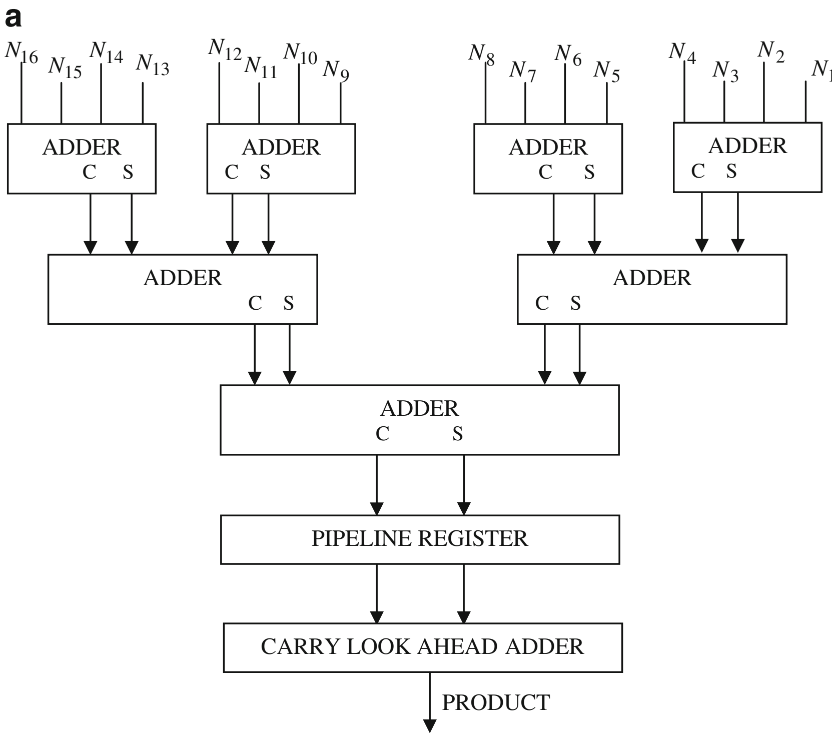
Figure 7 Internal pipelined architecture for adaptive decision feedback equalizer.



5 Architecture of ADFE

The adaptive decision feedback equalizer consists of feed forward filter (FFF) and feedback filter (FBF) and equalized signal is generated from the sum of the outputs of both the filter. The feed forward filter (FFF) in our design is a linear transversal filter which is implemented with a finite duration impulse response (FIR) filter with adjustable coefficients. The decisions made on the equalizer signal are fedback via a second transversal filter as shown in Fig. 6. If the detected symbols are stationary, then the ISI contributed by these symbols can be cancelled exactly by subtracting past symbol values with appropriate weight from the equalizer output [30]. But channel response may not be same always. Therefore, the forward and feedback filter should adjust its coefficients simultaneously [31] to counter the variation in channel response by minimizing the mean square error (MSE). At initial operation, the coefficients of the equalizer are needed to be adjusted as per the channel response and therefore, a known sequence of symbols named as training sequence is transmitted through the desired channel for this purpose. The design uses pipelined CORDIC unit as a main processing element. In circular rotation mode, CORDIC unit generates sinθ and cosθ which are the main ingredients for filtering and weight updating respectively in the proposed design as derived in Eq. (18, 19) [32].

The pipelined technique has been adopted throughout the architecture to implement delayed version of trigonometric form of LMS algorithm in FFF and simple delayed LMS (DLMS) for FBF. The reason behind that is the inputs to the FBF are the transmitted symbols whose bit lengths are very small and hardware required for multiplication is trivial. But it is not the case for FFF filter. Adaptation delay in FFF naturally more as compared to FBF. The μ_f and μ_b are the step sizes used



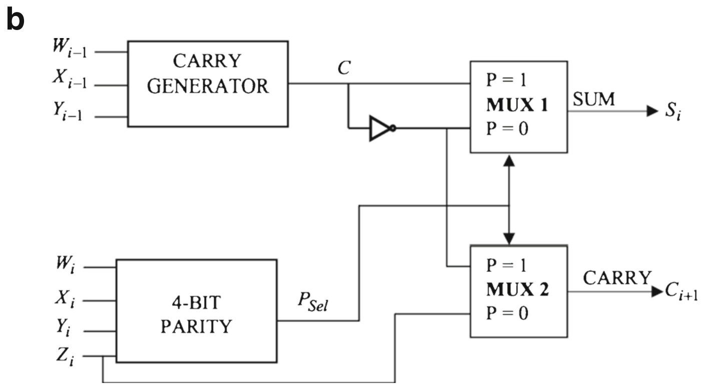


Figure 8 **a** Schematic of pipelined multiplier circuit. **b** Schematic of expended architecture of 4-to-2 adder

Table 3 Device utilization by multiplier.

Number of Slices:	45	out of	2448	1 %
Number of Slice Flip Flops:	39	out of	4896	0 %
Number of 4 input LUTs:	80	out of	4896	1 %
Number used as logic:	63			
Number used as Shift registers:	17			
Number of IOs:	36			
Number of bonded IOBs:	36	out of	158	22 %
Number of GCLKs:	1	out of	24	4 %

to compute the weight update term of FFF and FBF respectively to synchronize the error output. The filtering process in ADFE has been explained briefly with the help of architecture shown in Fig. 7. The FF and FB filter uses equal number of taps (N). The FIFO F_1, pipelined CORDIC processing unit and accumulator unit constitute the FF filter. On the other hand, FB filter consist of FIFO F_2, multiplier and FIFO F_4. Synchronization is essential within FF and FB filter and therefore, FIFO F_1 and F_2 has been employed for proper sequencing of input data to FF and FB filter. N number of input samples are circulated through the FIFO F_1 and F_2 at primary clock cycles. The new input samples are introduced after every (N+1) primary clock cycles by flipping the multiplexer MUX_1 and MUX_2. The FIFO F_3 and F_5 has been employed for proper sequencing of intermediate results required for angle and weight updating respectively. MUX_3 is being used to add desired signal for training of the equalizer. The architecture gives its output at every (N+1) primary clock periods. The primary clock is responsible for loading of data into different latches and the maximum frequency of primary clock, f_{PMax}, is totally dependent on propagation delay of the critical path of the design. In our design, the critical path is equal to propagation delay of single adder. The real part of input signals to the basic filtering block has been considered and processing elements are used in the filtering blocks engaged in corresponding operations only. $x(n)$ and $v(n)$ are the input signals for FF and FB filter respectively. Whereas, $w(n)$ and θ are the weight and angle updating signals are being used in the process. The computed result of present samples being processed are stored in *FIFO* F_1 and in *FIFO* F_2 of respective FF and FB filters before arrival of next sample on which similar process is to be carried out. To achieve high speed operation of the design, the computations on input signals are carried out simultaneously in FF and FB sections sothat the final outcome can be retrieved at every system clock cycle. Hence, on subsequent clock cycles, filter output come out and repeated processes continue.

The pipelined multiplier we have used in the design is Modified Booth encoding (MBE) algorithm based design which has been widely used in implementation due to its efficiency in reduction of partial products by half. The presented multiplication operations are performed using three steps [33]. All the partial products are generated in first step. For this purpose, popular MBE scheme has been employed and has not been discussed in detail in this design. The second step deals with reduction of all partial products in parallel structure using 4-to-2 adder. The third step uses fast adder named Carry Look-Ahead (CLA) to add the two numbers obtained from second stage to generate final product. The reduction of partial products using 4-to-2 adder has been shown in Fig. 8 (a). There are four 4-to-2 adder have been used to accommodate 16 partial product generated from the input coming from CORDIC as well as error generator circuits. Each 4-to-2 adder receives four inputs and gives two outputs in terms of sum and carry. The adder cell consists of carry generator which receives three inputs from previous adder cell and a parity generator which generates a control signal for the two MUXes to be selected. The expended architecture of the adder has been shown in Fig. 8 (b).

The multiplier has been synthesized on Xilinx 13.4 using target device spartan 3E xc3s250e-5-pq208. The resource utilization summary has been shown in Table 3.

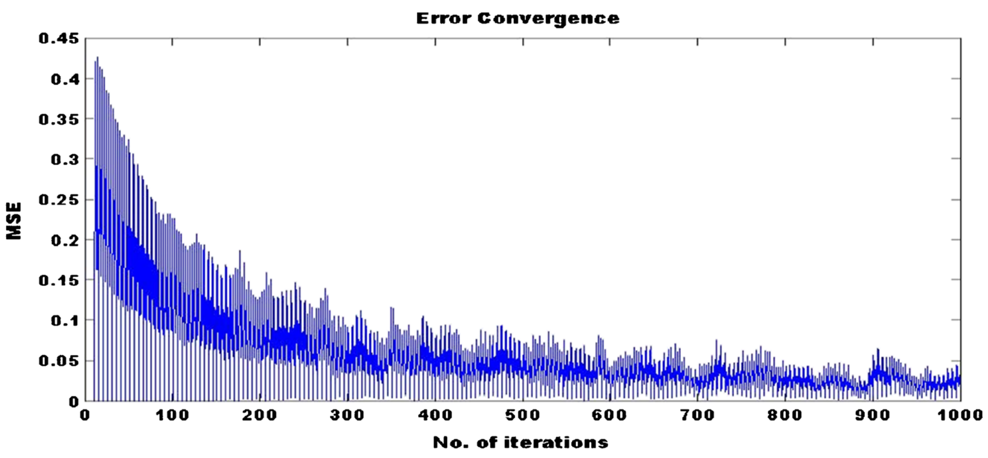


Figure 9 Learning Curve of the design at μ = 0.003.

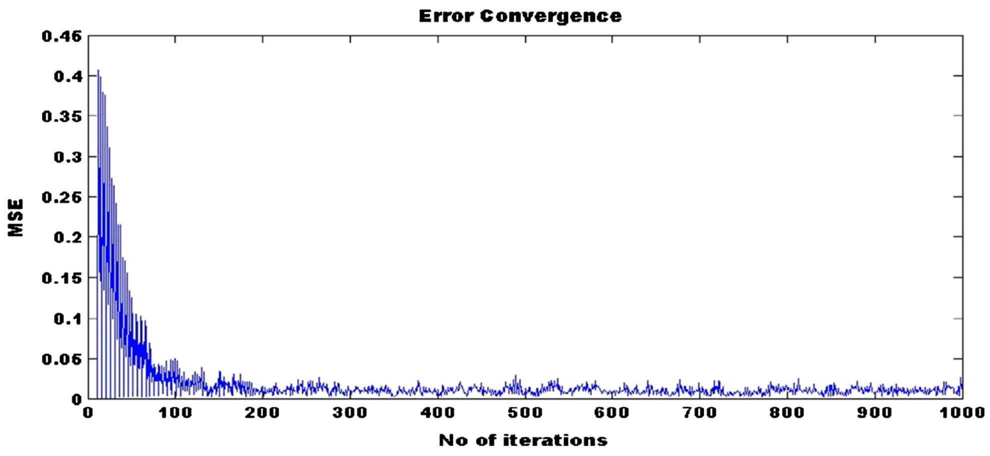


Figure 10 Convergence curve of the design at $\mu = 0.012$.

6 Performance Analysis and Discussion

It is well known that pipelined CORDIC itself has got good convergence property which has been efficiently used in this application. Exact convergence analysis of proposed architecture using trigonometric LMS algorithm is not possible due to presence of nonlinearity of the filter as well as angle update equation. Using the iteration error generated in the equalizer, approximate convergence studies have been carried out. The channel system function has been considered for simulation is $H(z)$,

$$H(z) = 0.315 + z^{-1} - 0.315z^{-2} \tag{28}$$

The input sequence to the channel is considered to be white Gaussian with corresponding unity variance. The channel

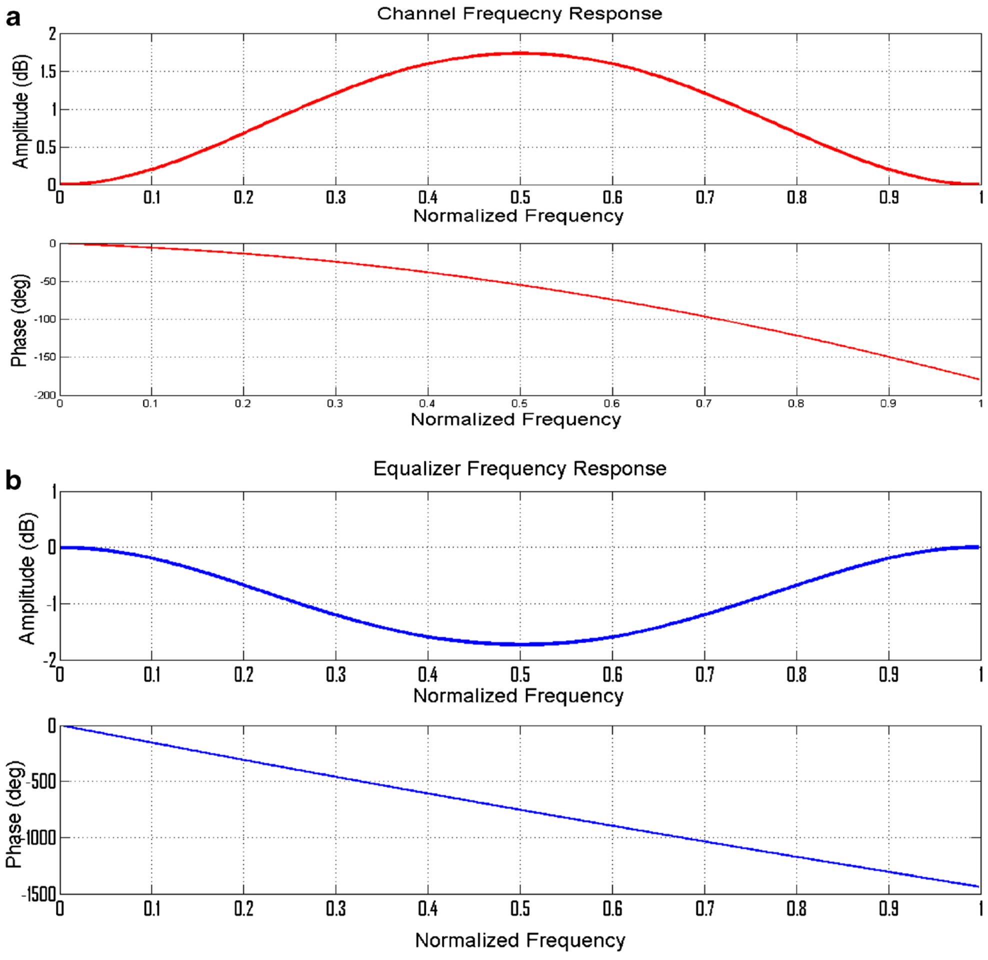


Figure 11 **a** The channel frequency response which is affected with the interferences. **b** The equalizer frequency response which is just inverse of that channel frequency response.

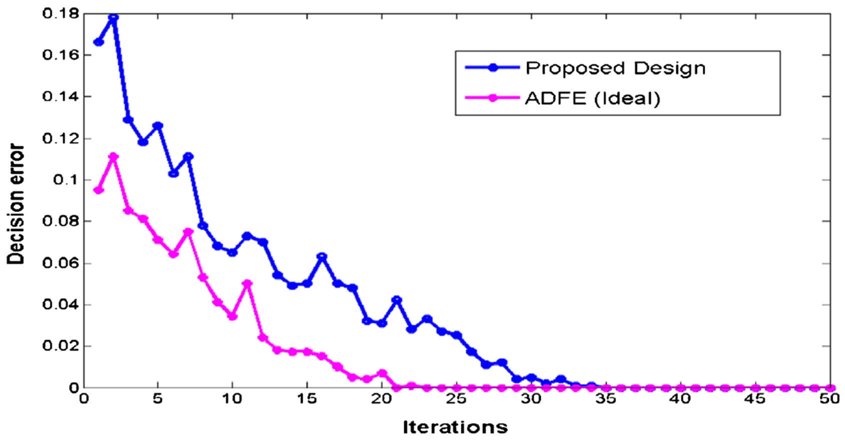


Figure 12 Comparative analysis of decision error Vs number of iterations for proposed and ideal ADFE.

noise is modeled as additive Gaussian noise process. The equalizer delay is set at 10. 11-tap adaptive filter with centre placed at 6-th tap has been taken for simulation studies. The tap weights are initialized to zero. The step size (μ) which is the guiding force for fast or slow conversion of the system is appropriately taken care. If $\mu \leq 0.002$ the convergent speed of the algorithm will be very slow and too large μ can cause instability. The simulation for convergence performance of ADFE consists of MATLAB simulated outcome in terms of learning curves of the design at $\mu=0.003$ and $\mu=0.012$ has been shown in Fig. 9 and Fig. 10 respectively.

The proposed design has been tested using randomly generated signal as an input for the equalizer which is to be used in mismatched channel. The equalization of mismatched channel has been shown in Fig. 11 (a) whereas, Figure 11(b) shows the equalizer response which is essentially inverse of the given channel response. Again MATLAB simulation of the proposed design and the corresponding ideal ADFE has been simulated and compared considering the decision error plots against number of iterations. The Fig. 12 shows that the proposed architecture gives good convergence result. The proposed design starts converging at iteration number 29 and finally it is fully converged at iteration 33. The ideal ADFE converges at iteration number 21.

The pipelined design of ADFE faces difficulty while implementing on the hardware due to its huge resource utilization. Though hardware utilization of pipelined architecture available in the literature is limited, we have tried to compare our design with the available pipelined design [17, 18] in the Table 4.

The proposed architecture has been coded in Verilog HDL and synthesized it using Cadence design tool. TSMC 90-nm CMOS library have been used for synthesis of the design. The area consumed by the hardware is 384,376 sq.um. Again, timing analysis shows that the delay produced by the design is 4.715 ns which lead to the operating speed of 212.09 MHz. The power consumptions of the proposed hardware have been tested with the same technology file at different operating frequencies and the results are shown in Table 5.

7 Conclusion

This paper presents realization of CORDIC based adaptive feedback equalizer using reformulated LMS algorithm in mitigating severe Inter Symbol Interference (ISI) in wire as well as wireless communication systems. The use of pipelined CORDIC computational architecture replaces few multipliers and makes pipelined implementation of adaptive DFE realizable on the chip. For better convergence, numbers of micro-

Table 4 Hardware complexity in different types of ADFE architectures [17, 18].

N-TAP ADFE	PIPEADFE1	PIPEADFE2	PCFADFE	PROPOSED
Multiplier in FFF	$2N_f$	$2N_f+N$	$2N_f+2N$	N_f
Multiplier in FBF	$2N_b$	$2N_b$	$2N_b$	$2N_b$
Total Adder	$2N_f+2N_b$	$2N_f+N+2N_b$	$2N_f+2N+2N_b$	N_f+2N_b+2

N_f and N_b stand for number in FF and FB filter

Table 5 Hardware power consumptions at different frequencies tested on TSMC 90.

Frequency	Power Consumed (nW)		
	Leakage Power	Dynamic Power	Total Power
50 MHz	1,852,613.070	26,249,672.709	28,102,285.779
100 MHZ	1,857,369.859	46,591,452.357	48,448,822.216
200 MHz	1,952,317.949	89,486,968.515	91,439,286.464
400 MHz	1,993,230.489	173,062,987.284	175,056,217.773
500 MHz	2,043,333.903	209,780,756.057	211,824,089.959

rotations and internal word-length has been optimized at the micro-level that not only increases the speed of operation, the decision error also minimized greatly. Again, optimization in quantization noise in CORDIC which is the main processing element in feed forward filter reduces instability in the performance of design as the system never get the quantization noise to accumulate in the feedback system. Therefore, proposed design achieves high degree of computational accuracy facilitating better application in channel equalization where ISI is severe. The convergence result shows the suitability of the design in real-time applications.

References

1. Farhang-Boroujeny, B. (2013). *Adaptive filters: theory and applications*. John Wiley & Sons.
2. R. Mishra, A. Mandal, (2012). Coordinate rotation algorithm based non-linear Adaptive Decision Feedback Equalizer, In *IEEE 2012 9th International Multi-Conference on Systems, Signals and Devices* (*SSD*), pp. 1–5
3. Volder, J. E. (1959). The CORDIC trigonometric computing technique. *IRE Transactions on Electronic Computers*, *EC-8*, 330–334.
4. Hu, Y. H. (1992). CORDIC-based VLSI architectures for digital signal processing. *IEEE Signal Processing Magazine*, *9*(3), 16–35.
5. Meher, P. K., et al. (2009). 50 years of CORDIC: algorithms, architectures, and applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *56*(9), 1893–1907.
6. Banerjee, A., Dhar, A. S., & Banerjee, S. (2001). FPGA realization of a CORDIC based FFT processor for biomedical signal processing. *Microprocessors and Microsystems*, *25*(3), 131–142.
7. Hu, Y. H., & Liao, H. E. (1992). CALF: a CORDIC adaptive lattice filter. *IEEE Transactions on Signal Processing*, *40*(4), 990–993.
8. Angarita, F., Canet, M. J., Sansaloni, T., Perez-Pascual, A., & Valls, J. (2008). Efficient mapping of CORDIC algorithm for OFDM-based WLAN. *Journal of Signal Processing Systems*, *52*(2), 181–191.
9. S. S. Haykin, (2008). Adaptive filter theory, 4 ed., Pearson Education India.
10. Sung, W., Ahn, Y., & Hwang, E. (2005). VLSI implementation of an adaptive equalizer for ATSC digital TV receivers. *Journal of VLSI signal processing systems for signal, image and video technology*, *40*(3), 301–310.
11. Chen, S., Hanzo, L., & Livingstone, A. (2006). MBER space-time decision feedback equalization assisted multiuser detection for multiple antenna aided SDMA systems. *IEEE Transactions on Signal Processing*, *54*(8), 3090–3098.
12. Chakraborty, M., & Pervin, S. (2003). Pipelining the adaptive decision feedback equalizer with zero latency. *Signal Processing*, *83*(12), 2675–2681.
13. Shaik, R. A., & Chakraborty, M. (2013). A block floating point treatment to finite precision realization of the adaptive decision feedback equalizer. *Signal Processing*, *93*(5), 1162–1171.
14. Magarini, M., Barletta, L., & Spalvieri, A. (2012). Efficient computation of the feedback filter for the hybrid decision feedback equalizer in highly dispersive channels. *IEEE Transactions on Wireless Communications*, *11*(6), 2245–2253.
15. Raghunath, K. J., & Parhi, K. K. (1993). Parallel adaptive decision feedback equalizers. *IEEE Transactions on Signal Processing*, *41*(5), 1956–1961.
16. Gatherer, A., & Meng, T. H. (1993). A robust adaptive parallel DFE using extended LMS. *IEEE Transactions on Signal Processing*, *41*(2), 1000–1005.
17. Shanbhag, N. R., & Parhi, K. K. (1995). Pipelined adaptive DFE architectures using relaxed look-ahead. *IEEE Transactions on Signal Processing*, *43*(6), 1368–1385.
18. Yang, M. D., Wu, A. Y., & Lai, J. T. (2004). Fast convergent pipelined adaptive DFE architecture using post-cursor processing filter technique. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *51*(2), 57–60.
19. Chakraborty, M., Dhar, A. S., & Lee, M. H. (2005). A trigonometric formulation of the LMS algorithm for realisation of pipelined CORDIC. *IEEE Trans. Circuits and Systems*, *52*(9), 530–534.
20. Banerjee, A., & Dhar, A. S. (2013). Pipelined VLSI architecture using CORDIC for transform domain equalizer. *Journal of Signal Processing Systems*, *70*(1), 39–48.
21. Wu, A., Ng, C. K., & Tang, K. C. (1998). Modified booth pipelined multiplication. *Electronics Letters*, *34*(12), 1179–1180.
22. Waters, R. S., & Swartzlander, E. E. (2010). A reduced complexity wallace multiplier reduction. *IEEE Transactions on Computers*, *59*(8), 1134–1137.
23. Kornerup, P. (2005). Reviewing 4-to-2 adders for multi-operand addition. *Journal of VLSI signal processing systems for signal, image and video technology*, *40*(1), 143–152.
24. Galbi, D., et al. (1990). *U.S. Patent No. 4,901,270*. Washington, DC: U.S. Patent and Trademark Office.
25. Kuang, S.-R., Wang, J.-P., & Guo, C.-Y. (2009). Modified booth multipliers with a regular partial product array. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *56*(5), 404–408.
26. Hu, Y. (1992). The Quantization Effects of the CORDIC Algorithm. *IEEE Transactions on Signal Processing*, *40*(4), 834–844.
27. Biswas, D., & Maharatna, K. (2015). A CORDIC-Based Low-Power Statistical Feature Computation Engine for WSN Applications. *Circuits, Systems, and Signal Processing*, 1–18. doi: 10.1007/s00034-015-0041-5.
28. Noll, T. G. (1991). Carry-save architectures for high-speed digital signal processing. *Journal of VLSI signal processing systems for signal, image and video technology, 3*(1–2), 121–140.
29. Meher, P. K., & Park, S. Y. (2013). CORDIC designs for fixed angle of rotation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *21*(2), 217–228.
30. Lin, Y. C., Jou, S. J., & Shiue, M. T. (2012). High throughput concurrent lookahead adaptive decision feedback equalizer. *IET Circuits, Devices and Systems*, *6*(1), 52–62.
31. Sailer, T., & Tröster, G. (2003). An Efficient VLSI Architecture for Computing Decision Feedback Equalizer Coefficients from the Channel State Information. *Journal of VLSI signal processing systems for signal, image and video technology*, *35*(1), 91–103.
32. Chakraborty, M., Dhar, A. S., & Pervin, S. (2001). CORDIC realization of the transversal adaptive filter using a trigonometric LMS algorithm. *In proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001 (ICASSP'01),* (2), 1225–1228.
33. Mandal, A., Mishra, R., Kaushik, B. K., & Rizvi, N. Z. (2015). Design of LMS Adaptive Radar Detector for Non-homogeneous Interferences, *IETE Technical Review.* doi:10.1080/02564602.2015.1093436.

Amritakar Mandal received his Bachelor of Engineering degree in Electronics and Communication Engineering from The Institution of Engineers (India) in 2006. He completed M.Tech with specialization in VLSI from Shobhit University, India in 2009. He is currently working toward the Ph.D degree in Electronics and Communication Engineering in the School of Information and Communication Technology, Gautam Buddha University, India. He is an Associate Member of IE (India) and Student Member of IEEE. He has vast experience in Electronic Warfare and Surveillance Radar systems. His research interests include Radar Signal Processing, Adaptive Filtering and High Speed VLSI Design for Communication Systems.

Rajesh Mishra is currently working as Assistant Professor in School of Information and Communication Technology, Gautam Buddha University, Greater Noida, Delhi NCR (India). He received his BE (Electronics Eng.), M. Tech and Ph. D. degree (Reliability Eng.) from Reliability Engineering Centre, IIT Kharagpur (India) in year 2000, 2004, and 2009 respectively. He has research interest in the area of reliability engineering, layout design for capacitated networks, and network optimization. He has published papers in several international journals such as IJPE, IEEE, RESS, QTQM etc.