J Sign Process Syst (2013) 70:39–48
DOI 10.1007/s11265-012-0657-7

Pipelined VLSI Architecture using CORDIC for Transform Domain Equalizer

Ayan Banerjee · Anindya Sundar Dhar

Received: 26 June 2009 / Revised: 20 March 2010 / Accepted: 17 January 2012 / Published online: 17 February 2012
© Springer Science+Business Media, LLC 2012

Abstract In this paper, a pipelined architecture using CORDIC for realization of transform domain equalizer is presented. Transform domain equalizer has much faster convergence than its time domain counterpart for practical hardware realization having nonzero adaptation delay. Here running DFT is employed as the transform, and CORDIC is used for realization of running DFT. Pipelining is applied throughout the architecture, thus limiting the critical path delay to the propagation delay of a single 16 bit adder for 16 bit arithmetic. For N tap equalizer, primary clock speed is N times of the sample clock speed, so that on arrival of each sample, the computation of whole transform and weight update is possible. In the proposed architecture, hardware complexity is reduced by fully utilizing the pipeline without using parallel structures. The adaptation delay is only 2 sample clock periods resulting in fast convergence. The proposed architecture is suitable for VLSI implementation with primary clock speed limited by the binary adder propagation delay which could be as low as 2 ns in the present state-of-the-art technology.

Keywords CORDIC · Running DFT · Transform domain equalizer · VLSI architecture

A. Banerjee (✉) · A. S. Dhar
Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721302, India
e-mail: ayan@telecom.becs.ac.in

A. Banerjee
e-mail: ayanb12@gmail.com

A. S. Dhar
e-mail: asd@ece.iitkgp.ernet.in

1 Introduction

In a communication system, an adaptive equalizer is employed to minimize the intersymbol interference by adaptively trying to mimic the inverse characteristics of that of communication channel. Least mean square (LMS) algorithm has been among the most popular ones to be utilized for the process of adaptive equalization [1]. The speed of convergence for the LMS algorithm depends on various factors such as input data statistics (e.g. eigenspread of the covariance) and stepsize parameter etc. It has been established [2] that, in a given practical situation, transform domain LMS (TXLMS) converges faster than the conventional counterpart. Here delayed LMS algorithm should be used, as delayless LMS cannot be physically realized due to inherent delay in hardware resources. However, computational cost increases considerably in TXLMS due to its requirement for evaluating associated transform. Thus it becomes important to realize efficient architecture that can implement TXLMS based adaptive equalizer, which could be used in a functional unit in a practical communication system.

This paper describes an architecture for TXLMS based adaptive equalizer where the transform concerned is employed through running discrete Fourier transform (DFT) [3] which operates on both real and imaginary data. The proposed architecture employs Co-ordinate Rotation Digital Computer (CORDIC) [4] unit in a pipelined configuration, as the primary functional block for computing running DFT, thereby saving the usage of conventional bulky multipliers. However, in the weight updating section, pipelined multipliers have been employed. Here running DFT acts as a sliding window, making it possible to update the weights in the transform domain on arrival of each input sample. In the proposed architecture, hardware economy is achieved by exploiting the pipelining

technique to the maximum extent without using any parallel structure. Excepting the size of the RAM, hardware requirement for the proposed architecture is invariant to the equalizer tap length.

2 Theoretical Groundwork

For a transmitted signal $s(t)$, in a communication system, the received signal $r(t)$ is given by

$$r(t) = s(t) \otimes h(t) \tag{1}$$

where $h(t)$ is the channel impulse response and $\otimes$ denotes the convolution operation. Now, if the received signal is passed through an equalizer, its output $y(t)$ is given by

$$y(t) = r(t) \otimes q(t) \tag{2}$$

where $q(t)$ is the impulse response of the equalizer. The responsibility of the equalizer is to adopt $q(t)$ in such a way that the originally transmitted signal $s(t)$ can be recovered from $y(t)$ by a simple decision making process. This is possible in a digital communication system, since the transmitted signal can have discrete values representing symbols from a specified constellation having finite number of points in it. The equalizer output can be quantized to map to the correct (as transmitted) constellation point provided that the error is contained such that the decision boundary remains distinct.

A conventional N-tap time domain equalizer uses N number of weights to adaptively attain the required impulse response, as closely as possible, within the scope of the adaptation algorithm. Let $V(n)=[v_0(n)\ v_1(n)\ \ldots\ldots\ v_{N-1}(n)]$ be the values of the N weights at discrete time index n and $R(n)=[r(n)\ r(n-1)\ \ldots\ldots\ r(n-(N-1))]$ be the present and past received signals appearing at the same time index n, at the so called tapped delay line which is nothing but a FIFO like register array.

The output of the time domain equalizer is given by

$$y(n) = V(n)R^T(n) \tag{3}$$

The error in estimating the actually transmitted signal $s(n)$ is given by

$$e(n) = s(n) - y(n) \tag{4}$$

In the LMS algorithm, the error $e(n)$ is utilized to update the weights as per the following expression:

$$V(n+1) = V(n) + \mu R^*(n)e(n) \tag{5}$$

where μ is the stepsize parameter that governs the speed of convergence and * denotes complex conjugate. Choosing the value of μ beyond certain limit may lead to instability and divergence.

In the transform domain equalizer [5] of length N, as shown in Fig. 1, a suitable orthogonal transformation is applied to the input vector $R(n)$ made up of a block of N input samples such that the transformed output vector (having N elements) $U(n)$ is given by

$$U(n) = R(n)F(n) \tag{6}$$

where $F(n)$ is a unitary matrix of dimension $N\times N$ representing the chosen transform.

The transformed output will now be combined by a set of N weights, say $W(n)=[w_0(n)\ w_1(n)\ \ldots\ldots\ w_{N-1}(n)]$ to form the final output as follows:

$$y(n) = W(n)U^T(n) \tag{7}$$

The error in estimating the transmitted signal $s(n)$, in this case also, is given by Eq. 4. The weights are updated by following an approach similar to Eq. 5:

$$W(n+1) = W(n) + \mu' U^*(n)e(n) \tag{8}$$

where μ' is the stepsize parameter for transform domain LMS algorithm.

In a practical hardware realization of the equalizer, while resorting to pipelined architecture for supporting high data rate, a delay is inadvertently introduced in updating the weights with respect to the input signal. Under such cases, to incorporate the effect of this delay

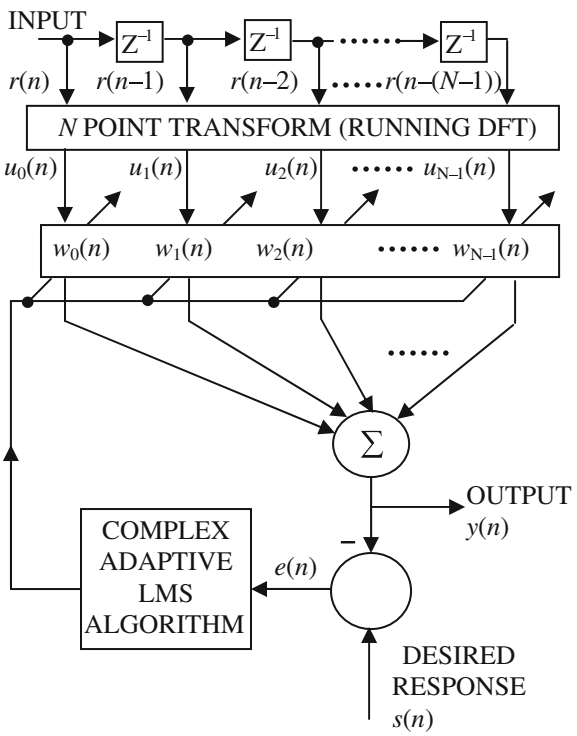


Figure 1 Transform domain equalizer block diagram.

D, the governing equations are getting slightly modified [6] as follows:

$$\begin{aligned} y(n-D) &= V(n-D)R^T(n-D) \\ e(n-D) &= s(n-D) - y(n-D) \\ V(n+1) &= V(n) + \mu R^*(n-D)e(n-D) \end{aligned} \tag{9}$$

This delay causes an offset between the calculated weights and the samples on which the weights act to produce the output, thereby resulting in slow convergence compared to the case where there is no computational delay. However, this does not pose a serious problem in achieving the final objective of equalizing the channel, provided that the signal statistics do not change by a large extent within the time span of such weight updating delay.

In this circumstance, transform domain equalization helps in accelerating the convergence process. After accommodating the delay, the set of governing equations for transform domain equalization becomes:

$$y(n-D) = W(n-D)U^T(n-D) \tag{10a}$$

$$e(n-D) = s(n-D) - y(n-D) \tag{10b}$$

$$W(n+1) = W(n) + \mu' U^*(n-D)e(n-D) \tag{10c}$$

Running DFT is chosen for our implementation of the transform domain equalizer and the former is computed with the help of CORDIC arithmetic module.

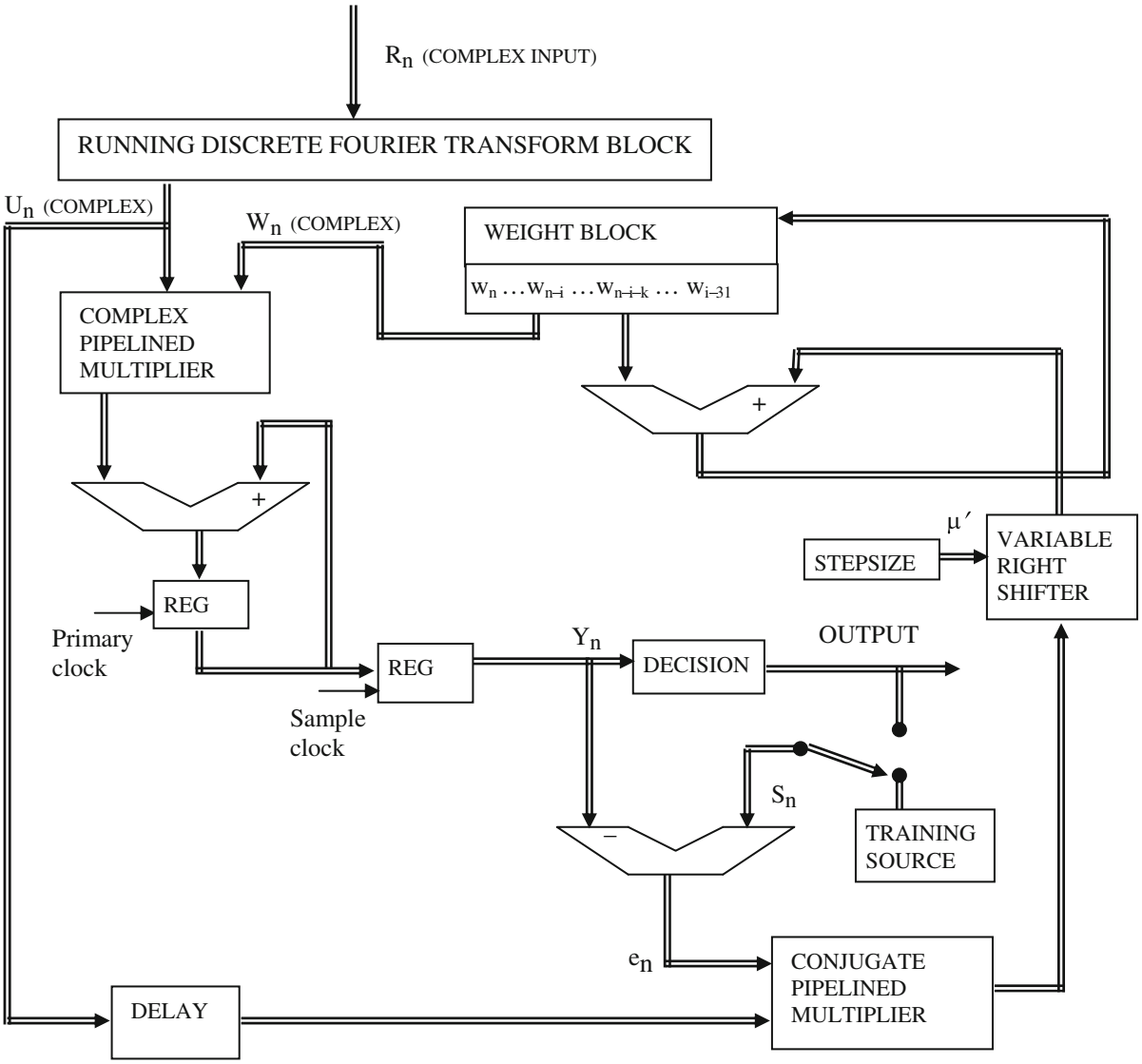


Figure 2 Transform domain equalizer architecture.

In the running DFT for N point transform length, $U_i(k)$, the k-th frequency component at i-th time instant, can be updated as follows:

$$U_{i+1}(k) = [U_i(k) - R(i-N+1) + R(i+1)]e^{2\pi jk/N} \tag{11}$$

where $R(i+1)$ is the $(i+1)$-th data sample. Multiplication by the factor $e^{2\pi jk/N}$, i.e., complex rotation is carried out by CORDIC operation.

In the CORDIC technique, the plane rotation through target angle α is done by decomposing it into several elementary angles and rotating through each of these angles as follows:

$$\alpha = \sum_{i=0}^{M-1} \sigma_i \theta_i \text{ where } \theta_i = \tan^{-1}\left(2^{-i}\right) \tag{12}$$

with M being the wordlength and $\sigma_i = +1$ or -1, deciding the direction of rotation. Plane rotation is achieved by pipelined operations as follows:

$$\begin{aligned} X_{i+1} &= X_i - \sigma_i Y_i 2^{-i} \\ Y_{i+1} &= \sigma_i X_i 2^{-i} + Y_i \\ Z_{i+1} &= Z_i - \sigma_i \theta_i \text{ with } Z_0 = \alpha, \sigma_i = \text{Sign}(Z_i) \end{aligned} \tag{13}$$

If operations in M number of pipelined stages are carried out as in Eq. 13, then the end results are scaled version of actual results (X_{act}, Y_{act}) as follows:

$$\begin{aligned} X_M &= K_M X_{act} \\ Y_M &= K_M Y_{act} \text{ where } K_M = 1/\prod_{i=1}^{M-1} \cos\theta_i \end{aligned} \tag{14}$$

3 Architecture Design

The proposed architecture for the transform domain equalizer is shown in Fig. 2. Here tap length N equals 32 and 16 bit internal arithmetic (M) is chosen. On arrival of each complex input (R_n), the running DFT block computes N frequency components as outputs. The N complex outputs of the running DFT block are fed successively to one input of complex pipelined multiplier, the other input is coming from the weight block. In the complex multiplier, N complex weight values are multiplied with corresponding transformed values from running DFT block. The N complex outputs from multiplier are accumulated to generate output as in Eq. 10a.

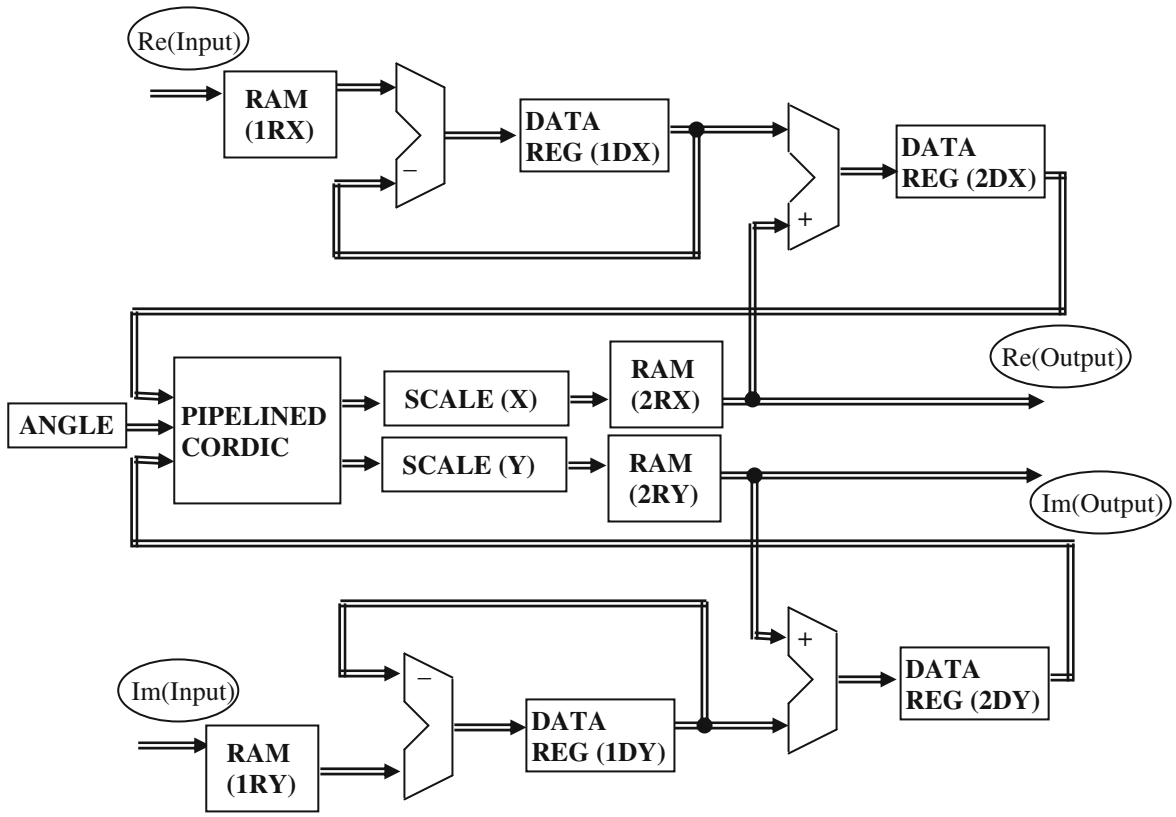


Figure 3 Running DFT architecture.

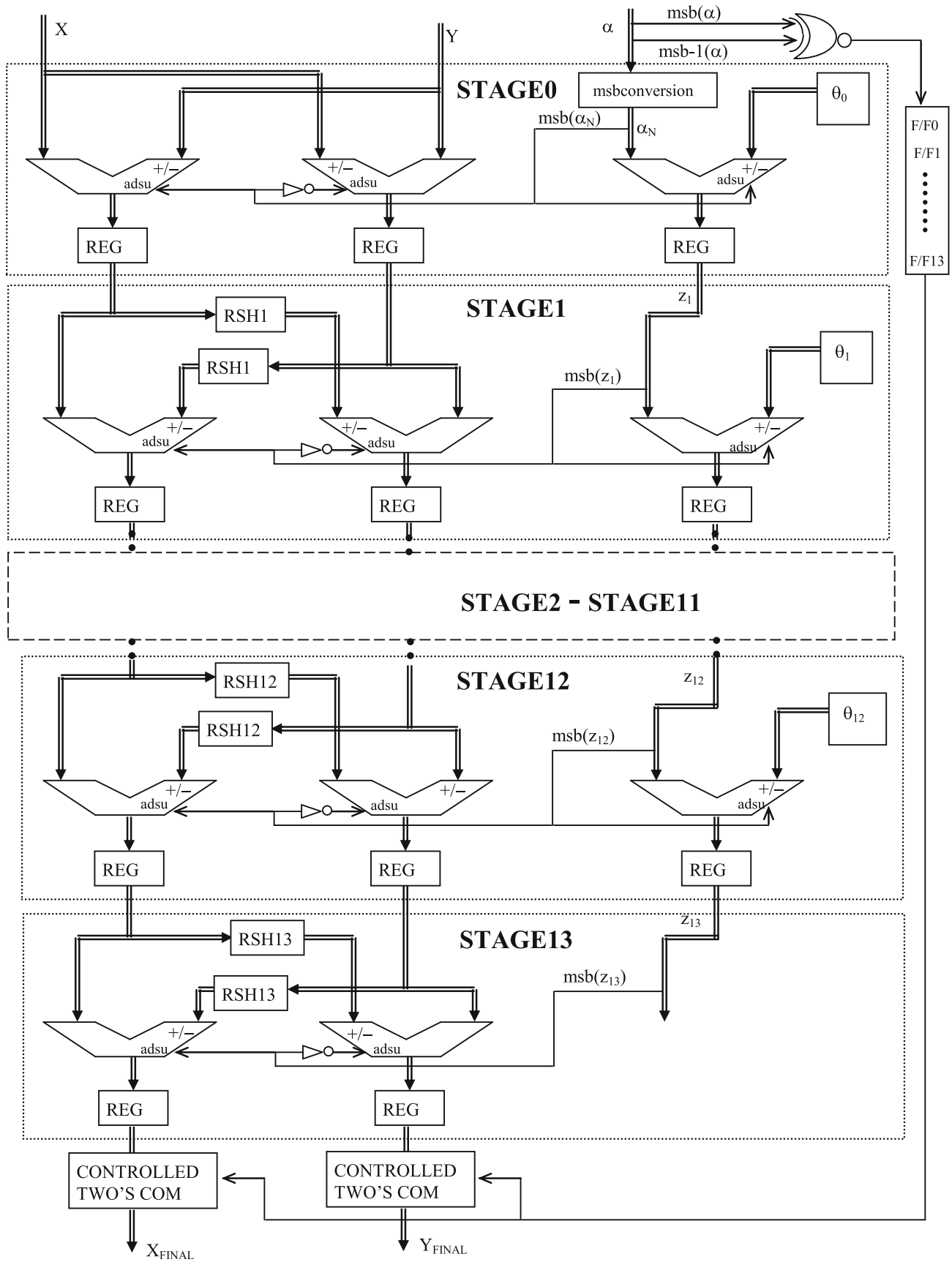


Figure 4 Pipelined CORDIC architecture.

The output is then subtracted from the desired response to generate the error signal as in Eq. 10b. The weight block is a dual port RAM along with two address generators, where initially first set of weight values is written. When weight from a particular location, say $(n-i)$ location, is read out for multiplication to generate $y(n)$, at the same time weight updating operation is going on at first by reading weight from $(n-i-k)$ location, keeping an offset from the other read operation, and updated weight after operation as per Eq. 10c is written back to the same $(n-i-k)$ location. Thus two weights are simultaneously accessed from two different address locations, with a fixed offset between them. The error signal is fed to another pipelined multiplier where complex conjugate operation is done. The other input to the pipelined multiplier is running DFT output after passing through a proper synchronization delay. The output of the conjugate multiplier is multiplied by stepsize μ' through a variable right shifter as μ' can be expressed here in the form 2^{-i}. Then it is added with previous weight, thus performing operation as per Eq. 10c to generate updated weight. The decision block is generating final output by taking decision as per structure of constellation points. Here the primary clock speed is $f_{max}=1/T_{delay(adder)}$ and the operations such as pipelined multiplications, accumulation operations to get $y(n)$, weight updating, running DFT internal operations are by primary clock and for N transform length the maximum sampling clock speed is f_{max}/N. The adaptation delay corresponds here to a maximum of two sampling clocks.

The main processing unit in this architecture is the running DFT block of transform length N that equals 32. Here the updating operation for a particular frequency component, as shown in Eq. 11, can be achieved through only an addition, a subtraction and a complex rotation amounting to $2\pi k/N$ for the k-th frequency component. Now the complex rotation is performed with the help of pipelined CORDIC module whose functionality is described in Section 3.A. The architecture of running DFT, shown in Fig. 3, processes real and imaginary data in two similar sections each comprising of two RAM blocks, adders, subtractors, scale blocks, two data registers, and pipelined CORDIC block, which is

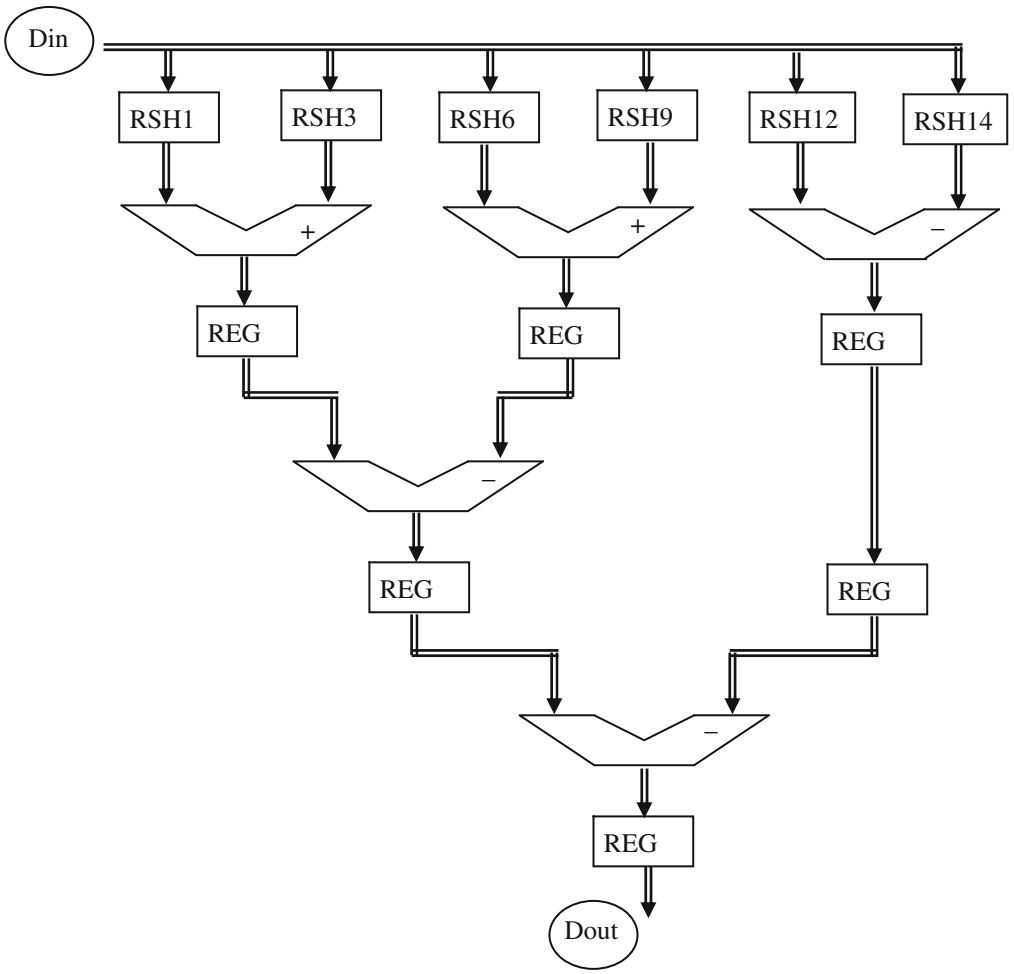


Figure 5 Scale block architecture.

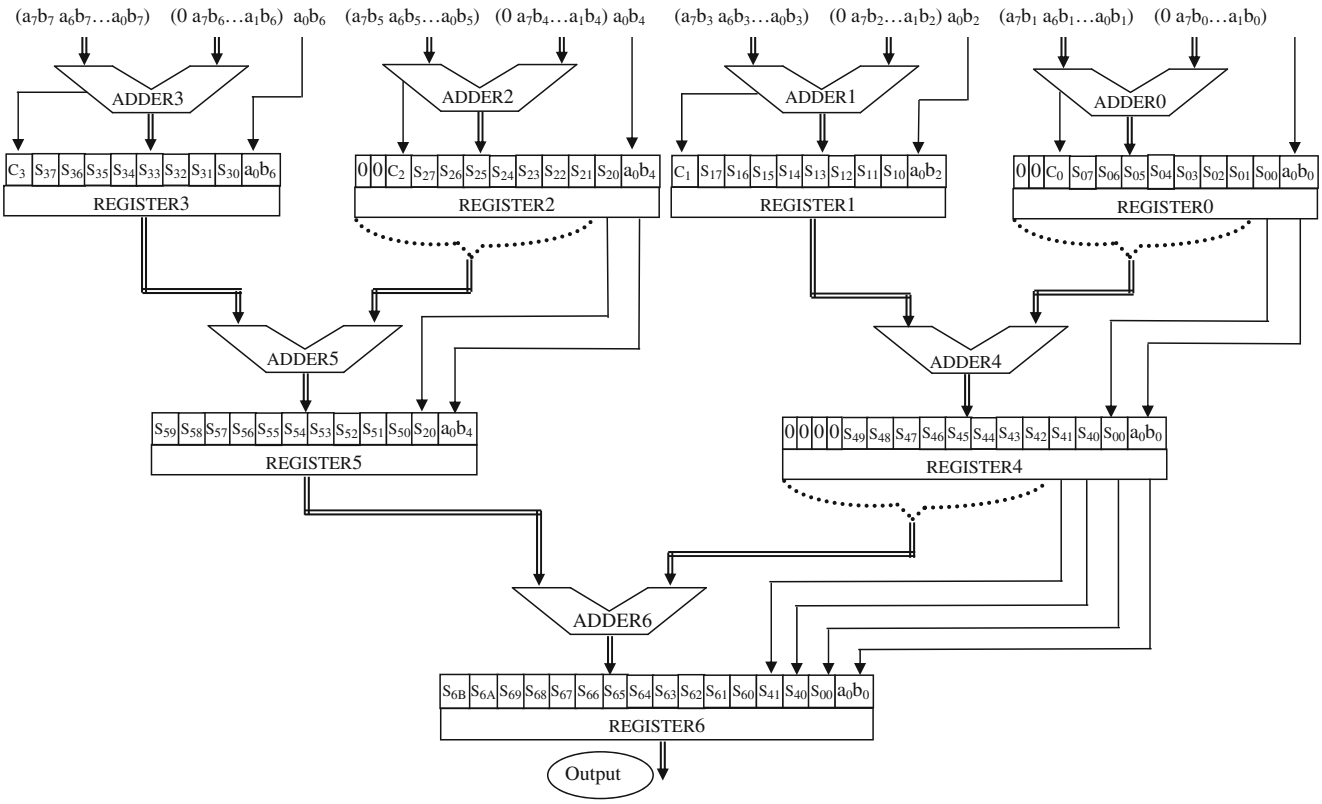


Figure 6 Pipelined multiplier architecture (8 bit unsigned binary).

common for both the sections. Here the primary clock speed is N times the sample clock speed so that on arrival of each sample, all N frequency components can be computed. The input RAM blocks (1RX and 1RY) are single port RAM, which can store N words of 16-bit length. The data is written in the input RAM and initially all the flip-flops and counters are cleared to zero. The general scheme for subtraction of (i–N)-th data from i-th data is as follows. Since the RAM can store N data, so the address of writing i-th data is the same where previously (i–N)-th data is written. For a given address, first the previously stored data is latched in data register (1DX(Y)) then the new data is written in the same location in the RAM. The latched data is subtracted from the new data by subtractor to obtain ($R(i+1)-R(i-N+1)$). For each input arrived at sample clock rate (fixed i), N different rotation angles (different k values) are fed to the CORDIC block successively at primary clock rate so that rotation corresponding to all N angles is achieved and N updated frequency components (real and imaginary) are available at the output in successive clock cycles after the latency period of the CORDIC processor. After the rotation operation, CORDIC outputs (real & imaginary) are fed to the scale blocks (X and Y) for multiplying it by a factor $\prod_{i=0}^{15} Cos\,\theta i = 0.607252935$, to get the actual value of the

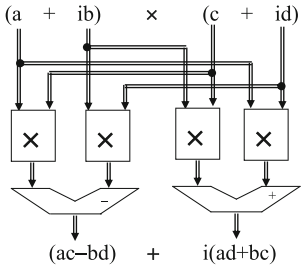


Figure 7 Complex multiplier architecture.

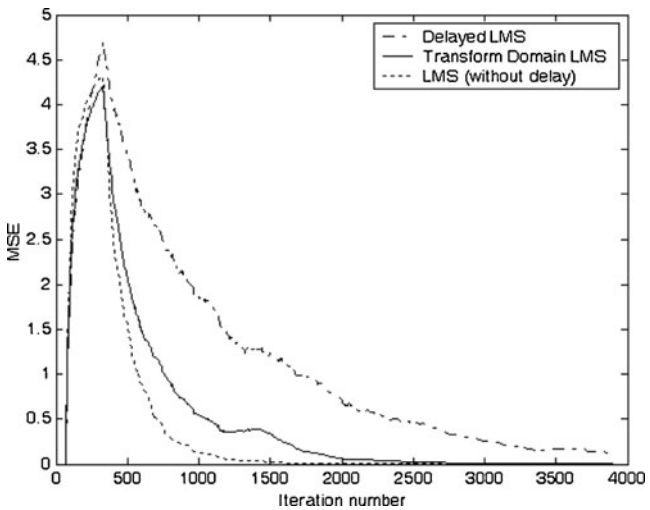


Figure 8 MSE convergence rate for different types of equalizer.

Table 1 Hardware utilization of proposed architecture.

N tap equalizer	16 bit adder	16 bit register	RAM (16 bit)	No. of blocks
CORDIC	43	43	–	1
Scale	76	78	–	2
Complex multiplier	18	18	–	2
Weight block	1	1	*N* word	1
Data RAM	–	–	*N* word	5

rotated output. After the scaling operation is done, the outputs are written in the dual port RAM blocks (2RX and 2RY). The data written corresponding to a particular frequency is to be read out again after the arrival of the next input data during the computation of the corresponding frequency component. This is done by keeping an offset between read and write addresses of the dual port RAM to cover the latency of the scale and CORDIC block. RAM block outputs are fed to adder, the other input of which is the new subtracted data ($R(i+1) - R(i-N+1)$). This subtracted data remaining same as one of the input of the adder, whereas the other inputs are the N frequency components corresponding to the previous input. The outputs of adder are fed back to the CORDIC block for the next set of rotation operations. The clock in the input RAM is the sample clock, which is obtained by a division of N of the primary clock, which is used in the other portion of the running DFT block.

3.1 Pipelined CORDIC Architecture

The CORDIC operation is done here through a pipelined structure, as shown in Fig. 4. There are three inputs at a time, real data (X), imaginary data (Y) and target angle (α). The total rotation is achieved here by decomposing the target angle α in elementary angles $\tan^{-1}(2^{-i})$, i starting from 0 to 15, and also in clockwise or anticlockwise direction for a particular i, as required. But as per normalized angle representation scheme [7], value of $\tan^{-1}(2^{-13})$ is equal to weight of LSB only, so rotation angle is decomposed up to this value. For rotation of each of these 14 angles, we have to perform three addition/subtraction operations along with right shifting (in two cases) as per Eq. 13. Since the amount of shift is fixed for a particular stage, it can be realized by bus cross connection only, without deploying additional hardware. The operation proceeds through stages in a pipelined structure, each stage performing a defined amount of rotation by taking data set from its previous stage, and new data set is entering in very first stage in each clock cycle. However, at the time of loading the target angle in the very first stage, the corresponding register copies MSB−1 bit as the MSB instead of the original value of MSB which has got an weightage equals to $-\pi$. This is done so as to keep the target angle α within the range $[-\pi/2, \pi/2]$. If the target angle is beyond that limit, i.e. if it lies either in the second or in the third quadrant, the sign changing of the output variables X and Y are to be carried out to incorporate the reflections about the axes, as described in [7]. The controlled sign change for the two outputs are done through controlled two's complementers. The pipe fills in 15 clock cycles, which defines its latency and then we get one set of outputs (X and Y) in every clock cycle, which defines its throughput. The maximum combinatorial delay in this CORDIC block for 16 bit internal arithmetic is delay of a 16 bit adder, which also defines its maximum clock speed as $f_{max}=1/T_{delay(adder)}$

3.2 Scale Block Architecture

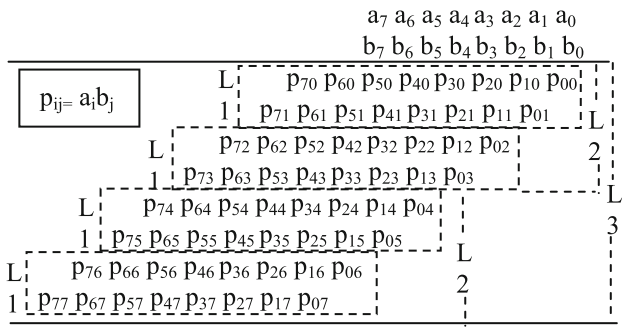
The scale block is designed for a fixed amount of multiplication of the input and it is also implemented in a pipelined structure. Let us consider the structure of the scale block, shown in Fig. 5, which multiplies the input by the factor 0.607252935. So here the relation between the input and the output is as follows: dout=din×0.607252935=din.($2^{-1}+2^{-3}-2^{-6}-2^{-9}-2^{-12}+2^{-14}$), assuming 16 bit accuracy. Here the addition/subtraction operation is done in three stages. In the first stage, input din is right shifted by appropriate amounts (through RSH blocks by bus cross connection only), and then necessary additions/subtractions are carried out in this stage and also in the second and third stage, to get final output. This scale block is a pipeline structure and the operation proceeds here stage by stage in successive clock cycles. So the latency of the scale block is 3 clock cycles and throughput is one output per clock cycle. The maximum combinatorial delay in this scale block for 16 bit internal arithmetic is delay of a 16 bit adder which also defines its maximum clock speed as $f_{max}=1/T_{delay(adder)}$.

Table 2 Comparison between various architectures.

N tap equalizer	DFT_DLMS [9]	FD LMS [10]	Proposed
Adaptation delay	$(2N+1)T_{sample}$	$(N-1)T_{sample}$	$2T_{sample}=2NT_{primary}$
Critical period	T_{mult}	$T_{mult}+T_{add}$	T_{add}
No. of multipliers	$5N-2$	$16N$	8
No. of adders	$5N+1$	–	58

3.3 Pipelined Multiplier Architecture

There are two multipliers used in this design, one is complex and the other is complex conjugate multiplier, each input is 16-bit signed binary. The basic module is 8-bit×8-bit unsigned binary multiplier, which is shown in Fig. 6. There are four addition operations done in parallel in level1 (L1) by taking the AND-ed inputs (a_ib_j) as shown below:



Four 8-bit adders ADDER0 to ADDER3 are used in level1 pipelining, whose outputs are latched in REGISTER0 to REGISTER3. Here S_{ij} is the j-th sum bit of i-th adder and C_i is the carry output of i-th adder. In level2 (L2) pipelining, two parallel addition operations are done by taking required portions of output from level1 by two 10-bit adders ADDER4 and ADDER5 and latched in REGISTER4 and REGISTER5. In each of ADDER4 and ADDER5, the output is limited to 10 sum-bit since for one of the inputs, MSB and MSB−1 are always zero and thus no final carry bit is generated. In level3 (L3) pipelining, one addition operation is done by taking required portions of output from level2 by one 12-bit adder, ADDER6 and latched in REGISTER6. In ADDER6 also, output is limited to 12-bit and no final carry is generated. Final 16-bit output is taken from REGISTER6. This basic module is used to realize 16×16 bit signed binary multiplier. In complex multiplier, used in the equalizer, complex numbers (a+jb) and (c+jd) are multiplied to generate output (ac−bd)+j(ad+bc) using four multipliers, one adder and one subtractor as shown in Fig. 7. In realization of complex conjugate multiplier, role of adder and subtractor is interchanged. The latency for this basic module is 3 primary clock cycles and throughput is one per clock cycle and critical path delay is limited to the propagation delay of a 16-bit adder/subtractor.

4 Results and Discussion

The novelty of the proposed architecture lies in utilizing running DFT for transform domain equalization. The running DFT architecture processes both real and imaginary data at its input. This architecture is fully pipelined with throughput of one set of real and imaginary data per clock cycle and on arrival of each new input, it computes all 32 frequency components for 32 point running DFT operation in 32 clock cycles. The running DFT architecture, reported in [8], processes only real data at its input and throughput is 20 clock cycles per output and computes only 1 frequency component for single carrier and 8 frequency components for multi carrier systems. The adaptation delay for the transform domain equalizer is 2 sample clock cycles in the architecture realized here. The MSE convergence rate for time domain delayed LMS and transform domain LMS taking an adaptation delay of 2 clock cycles are compared with LMS (without delay), as shown in Fig. 8, having stepsize values 0.00012, 0.00001 and 0.0005 respectively for 64 point QAM input having P and Q values {−7, −5, −3, −1, 1, 3, 5, 7}. The stepsize values are so chosen that the convergence and stability are guaranteed satisfying the constraints as described in [5]. Since the hardware realization for delayless LMS equalizer is not feasible in practice, the transform domain equalizer has got much faster convergence rate than its time domain counterpart. The critical path delay in all the blocks used in the equalizer architecture is limited to the propagation delay of a 16 bit binary adder/subtractor for 16 bit internal arithmetic. The sample clock speed is limited by $f_{primary}/32$ for 32 tap equalizer, and $f_{primary}/N$ for N tap equalizer and the adaptation delay is 2 sample clock cycles for N tap equalizer and thus same in terms of sample clock for any N. The hardware utilization of various blocks in terms of 16 bit adder, data register and RAM bit is shown in Table 1. Thus with increasing tap length only RAM size is increasing, other hardware requirements remaining same. The comparison between various architectures is shown in Table 2. The proposed architecture requires less hardware resource, than as in [9, 10], without sacrificing convergence rate. Proposed architecture is suitable for implementation in FPGA and ASIC, and critical path delay may be as low as 2 ns as reported in [11].

References

1. Qureshi, S. U. H. (1985). Adaptive equalization. *Proceedings of the IEEE, 73*(9), 1349–1387.
2. Narayan, S. S., & Peterson, A. M. (1981). Frequency domain LMS algorithm. *Proceedings of the IEEE, 69*(1), 124–126.
3. Liu, J.-C., & Lin, T.-P. (1988). Running DHT and real-time DHT analyzer. *Electronics Letters, 24*(12), 762–763.
4. Haviland, G., & Tuszynski, A. (1980). A CORDIC arithmetic processor chip. *IEEE Transactions on Computers, C-29*(2), 68–79.
5. Narayan, S. S., Peterson, A. M., & Narasimha, M. J. (1983). Transform domain LMS algorithm. *IEEE Transactions on Acoustics, Speech, & Signal Processing, ASSP-31*(3), 609–615.

6. Long, G., Ling, F., & Proakis, J. (1989). The LMS algorithm with delayed coefficient adaptation. *IEEE Transactions on Acoustics, Speech, & Signal Processing, 37*(9), 1397–1405.
7. 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.
8. Banerjee, A., & Dhar, A. S. (2005). Novel architecture for QAM modulator-demodulator and its generalization to multi-carrier modulation. *Microprocessors and Microsystems, 29*(7), 351–357.
9. Santha, K. R., & Vaidehi, V. (2004). *Design of synchronous and asynchronous architectures for DFT based adaptive equalizer* (pp. 383–389). Proc. IEEE Conf. SoutheastCon, Greensboro, NC, USA, 26–29 Mar.
10. Glentis, G., & Georgoulakis, K. (2006). Pipelined architectures for the frequency domain linear equalizer. *International Journal of Applied Mathematics and Computer Science, 16*(4), 525–535.
11. Yu, Z., Yu, M. -L., & Willson, A. N. Jr. (2001). *Signal representation guided synthesis using carry-save adders for synchronous data-path circuits* (pp. 456–461). Proc. 38th Design Automation Conf., Las Vegas, NV, USA, 18–22 June.

Ayan Banerjee received his Bachelor of Engineering Degree in Electronics and Telecommunication Engineering from Bengal Engineering College, Shibpur, Howrah, India in 1994. He completed M.Tech in Electronics and Electrical Communication Engineering with specialization in Integrated Circuits and Systems Engineering from Indian Institute of Technology, Kharagpur, India in 1999. He worked as Research Engineer in Centre for Development of Telematics, Delhi, India. He is presently working as Assistant Professor in Electronics & Telecommunication Engineering Department, Bengal Engineering and Science University, Shibpur, Howrah, India and also pursuing his Ph. D. at Electronics and Electrical Communication Engineering Department, Indian Institute of Technology, Kharagpur, India. His research interests include CORDIC based DSP architectures, VLSI Architectures for Communication Systems and Biomedical Engineering.

Anindya Sundar Dhar received his Bachelor of Engineering Degree in Electronics and Telecommunication Engineering from Bengal Engineering College, Shibpur, Howrah, India in 1987. He received his M.Tech degree in Electronics and Electrical Communication Engineering with specialization in Integrated Circuits and Systems Engineering from Indian Institute of Technology, Kharagpur, India in 1989. He received his Ph.D. degree in 1994 from the same Institute, where he is presently serving as Associate Professor in the Department of Electronics and Electrical Communication Engineering. His research interests include VLSI for Communication, High speed DSP architectures.