Circuits Syst Signal Process (2016) 35:603–618
DOI 10.1007/s00034-015-0076-7

An Efficient Distributed Arithmetic-Based Realization of the Decision Feedback Equalizer

M. Surya Prakash[1] · **Rafi Ahamed Shaik**[2] · **Sagar Koorapati**[3]

Received: 29 August 2014 / Revised: 4 May 2015 / Accepted: 5 May 2015 / Published online: 23 May 2015

Abstract A distributed arithmetic (DA)-based decision feedback equalizer architecture for IEEE 802.11b PHY scenarios is presented. As the transmission data rate increases, the hardware complexity of the decision feedback equalizer increases due to requirement for large number of taps in feed forward and feedback filters. DA, an efficient technique that uses memories for the computation of inner product of two vectors, has been used since DA-based realization of filters can lead to great computational savings. For higher-order filters, the memory-size requirement in DA would be high, and so ROM decomposition has been employed. The speed is further increased by employing digit-serial input operation. Two architectures have been presented, namely the direct-memory architecture and reduced-memory architecture where the later is derived using the former. A third architecture has also been presented where the offset-binary coding scheme is employed along with the ROM decomposition and digit-serial variants of DA. Synthesis results on Altera Cyclone III EP3C55F484C6 FPGA show that the proposed DA-based implementations are free of hardware multipliers and use less number of hardware resources compared to the multiply-and-accumulate-based implementation.

✉ Rafi Ahamed Shaik
rafiahamed@iitg.ernet.in

M. Surya Prakash
surya@iitg.ernet.in

Sagar Koorapati
koorapati.sagar@gmail.com

[1] Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, Assam, India

[2] IIT Guwahati, Room No. #304, G-Block, Academic Complex, Guwahati 781039, Assam, India

[3] Ineda Systems Pvt Ltd, Hyderabad 500 034, Telangana, India

Keywords Inter symbol interference (ISI) · Decision feedback equalizer (DFE) · Distributed arithmetic (DA) · Feed forward filter (FFF) · Feedback filter (FBF) · Field programmable gate array (FPGA)

1 Introduction

In telecommunications, intersymbol interference (ISI) occurs because of the multipath propagation of the transmitted signal and due to the band limited nature of channels. Equalizers [19,20] are employed at the receiver end in order to cancel the effect of multipath channels. Linear equalizers can counter the ISI by estimating the inverse of the discrete time model of the channel. However, noise power would theoretically become infinity at those frequencies where the channel transfer function becomes zero. Even, in the absence of spectral nulls, there would be a substantial enhancement of noise power in the vicinity of those frequencies which are greatly attenuated within the bandwidth of the transmitted signal. Decision feedback equalizers (DFE) [7] are widely used for effectively equalizing the channels that exhibit nulls in their frequency spectrum . Unlike the linear equalizers, DFEs do not estimate the inverse of the channel directly. A DFE basically consists of a feed forward filter (FFF), a feedback filter (FBF) and a decision device (quantizer). The FFF works directly on the received data and equalizes the anti-causal part of the channel transfer function. The residual ISI at the FFF output is then cancelled out by subtracting the FBF output from the FFF output. The FBF whose coefficients are carefully chosen operates on the decisions made on the past symbols. The DFE works basically on the assumption that there are no errors at the output of the decision device. As long as the decisions are correct, the DFE can equalize the channel effectively with low noise enhancement.

The problem with DFEs is that the sizes of FFF and FBF increase as the transmission data rate increases. This is because of the fact that as the transmission data rate increases, more and more symbols get overlapped, which demands for large number of taps for FFF and FBF. Hence, when implementing the DFEs, more number of multiply-and-accumulate (MAC) units are to be employed and operated in parallel in order to cope up with the transmission speed. But, due to the presence of large number of multipliers, the system would become complex and the real-time implementation becomes difficult. Distributed arithmetic [5,31], an efficient technique to compute the sum-of-product of two vectors, can be employed since it can realize vectors of any size without the presence of a hardware multiplier. Further, DA-based realization can lead to good throughput achievements since the computation speed depends on the bit-length of the input vectors unlike the MAC-based implementation where it depends on the length of the vectors. Most of the digital signal processing (DSP) algorithms such as convolution, correlation and fast transforms are principally the sum-of-product operations [15]. Hence, past work on DA-based realization mostly included finite and infinite impulse-response digital filters [2,3,28,29,34], transforms [8,9,22,33], rotation operations [4,27], adaptive digital filters [1,10,26,30]. Recent works based on distributed arithmetic once again include transforms and filtering [11–14,16–18,21,23,32]. In the most recent work [24,25], a block floating point realization (BFP) of DFE and adaptive DFE (ADFE) have been presented where the processing cost is marginally higher compared to the fixed point implementation. So far, DA technique has not been exploited for the efficient realization of the decision feedback equalizer. In this paper, we present a DA-based decision feedback equalizer architecture that consumes less hardware resources operating at high frequency compared to conventional (MAC-based) DFE implementations.

The paper is organized as follows: Section 2 presents the background of DA concept. Section 3 presents the proposed DA-based implementation of DFE where DA treatment to both FFF and FBF blocks are carried out separately. The analysis of hardware complexity and speed of MAC-based and DA-based schemes and their implementation details are discussed in Sect. 4, and the superiority of the proposed scheme over MAC-based scheme is also described here. In the entire paper, variables with the subscripts f and b represent the terms related to FFF and FBF, respectively.

2 Distributed Arithmetic Background

Distributed arithmetic is an efficient way of computing the inner product of two vectors in a fixed number of clock cycles irrespective of the length of vectors. DA works on the principle that the one-dimensional scalar convolution is equivalent to two-dimensional binary convolution [6]. In DA, the basic idea behind the computation of inner product is to store the partial products in memory and accessing and shift-accumulating them will compute the result. The computation of inner product of two vectors using DA is explained in the following paragraphs.
Consider the inner product of two vectors '**c**' and '**x**' ($i = 0, 1, \ldots, N - 1$) which may be given as

$$y = \sum_{i=0}^{N-1} c_i x_i \tag{1}$$

If every sample of x_i is written in fixed point $Q1.B - 1$ signed 2's-complement representation,

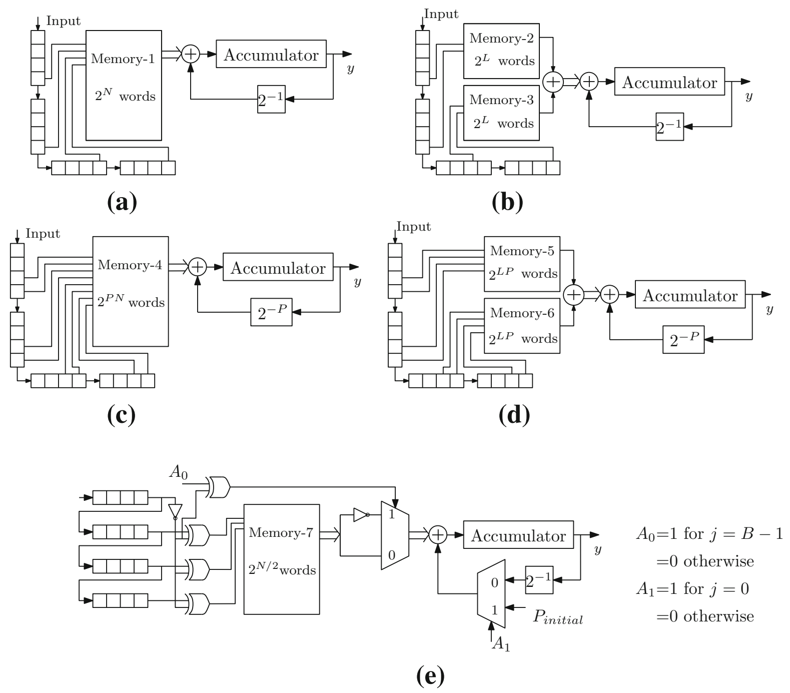
$$x_i = -b_{i,B-1} + \sum_{j=1}^{B-1} b_{i,B-1-j} 2^{-j} \tag{2}$$

where $b_{i,j} \in \{0, 1\}$ is the j-th bit in the binary representation of x_i . Now,

$$y = \sum_{j=0}^{B-1} \left[\sum_{i=0}^{N-1} c_i b_{i,j} \right] 2^j = \sum_{j=0}^{B-1} f\left(c_i, b_{i,j}\right) 2^j \tag{3}$$

where,

$$f\left(c_i, b_{i,j}\right) = \sum_{i=0}^{N-1} c_i b_{i,j} \tag{4}$$



Memory words						
Memory-1	Memory-2	Memory-3	Memory-4	Memory-5	Memory-6	Memory-7
0			0	0	0	$\frac{1}{2}[c_0 - c_1 - c_2 - c_3]$
c_3	0	0	c_3	c_3	c_1	
c_2			$2c_3$	$2c_3$	$2c_1$	$\frac{1}{2}[c_0 - c_1 - c_2 + c_3]$
$c_2 + c_3$			$3c_3$	$3c_3$	$3c_1$	
c_1			c_2	c_2	c_0	$\frac{1}{2}[c_0 - c_1 + c_2 - c_3]$
$c_1 + c_3$	c_3	c_1	$c_2 + c_3$	$c_2 + c_3$	$c_0 + c_1$	
$c_1 + c_2$			$c_2 + 2c_3$	$c_2 + 2c_3$	$c_0 + 2c_1$	$\frac{1}{2}[c_0 - c_1 + c_2 + c_3]$
$c_1 + c_2 + c_3$			$c_2 + 3c_3$	$c_2 + 3c_3$	$c_0 + 3c_1$	
c_0			⋮	$2c_2$	$2c_0$	$\frac{1}{2}[c_0 + c_1 - c_2 - c_3]$
$c_0 + c_3$			⋮	$2c_2 + c_3$	$2c_0 + c_1$	
$c_0 + c_2$	c_2	c_0	⋮	$2c_2 + 2c_3$	$2c_0 + 2c_1$	$\frac{1}{2}[c_0 + c_1 - c_2 + c_3]$
$c_0 + c_2 + c_3$			⋮	$2c_2 + 3c_3$	$2c_0 + 3c_1$	
$c_0 + c_1$			⋮	$3c_2$	$3c_0$	$\frac{1}{2}[c_0 + c_1 + c_2 - c_3]$
$c_0 + c_1 + c_3$	$c_2 + c_3$	$c_1 + c_0$	$3c_3 + 3c_2 + 3c_1$	$3c_2 + c_3$	$3c_0 + c_1$	
$c_0 + c_1 + c_2$			$+3c_0$	$3c_2 + 2c_3$	$3c_0 + 2c_1$	$\frac{1}{2}[c_0 + c_1 + c_2 + c_3]$
$c_0 + c_1 + c_2 + c_3$				$3c_2 + 3c_3$	$3c_0 + 3c_1$	

(f)

Fig. 1 Architecture for computing the inner product of two 4-length vectors using **a** DA. **b** DA ROM decomposition. **c** Digit-serial input scheme. **d** ROM decomposition and digit-serial input. **e** DA with OBC scheme. **f** Memory words stored inside the memories for different DA structures

$$y = -\sum_{i=0}^{N-1} c_i b_{i,B-1} + \sum_{j=1}^{B-1} \left\{ \sum_{i=0}^{N-1} c_i b_{i,B-1-j} \right\} 2^{-j} \tag{5}$$

Let

$$f_j = \sum_{i=0}^{N-1} c_i b_{i,B-1-j} \tag{6}$$

and

$$C_{B-1-j} = \begin{cases} -f_0 & j = 0 \\ f_j & j \neq 0 \end{cases} \tag{7}$$

Hence, (5) becomes

$$y = \sum_{j=0}^{B-1} C_{B-1-j} 2^{-j} \tag{8}$$

From (6) and (7), it can be observed that, taking j-th bit from each of x_i, the term C_{B-1-j} would take only one out of 2^N possible combinations which are nothing but the partial products of elements of c_i. Hence all these combinations can be stored in an LUT (usually a read-only memory (ROM)) whose address bits are formed by j-th bit of every x_i. Then the output y can be computed by shifting the partial products taken from the memory for every j-th ($j = 0, 1, \ldots, B-1$) set of bits of x_i's and accumulating all of them using shifting operation. As there are B bits in all x_i's counting from LSB to MSB, the system would take a total of B clock cycles to compute the inner product. Figure 1a shows the DA structure for computation of the inner product as per (8) with $B = 4$. Here, the address bits are indexed by j and the memory word is indexed by i .

As there are 2^N partial products, the size of the LUT would be 2^N, and so if N is large, the LUT size requirement would be high. DA provides the flexibility for the usage of multiple small-sized LUTs if N is split as $N = M \times L$ in which case (6) becomes

$$f_j = \sum_{i=0}^{ML-1} c_i b_{i,B-1-j} \tag{9}$$

$$= \sum_{i=0}^{L-1} c_i b_{i,B-1-j} + \sum_{i=K}^{2L-1} c_i b_{i,B-1-j} + \cdots + \sum_{i=(M-1)K}^{ML-1} c_i b_{i,B-1-j} \tag{10}$$

$$= \sum_{m=0}^{M-1} \sum_{i=mL}^{(m+1)L-1} c_i b_{i,B-1-j} \tag{11}$$

If $s = i - mL$ and if the dummy variable s is replaced by i, then

$$f_j = \sum_{m=0}^{M-1} \left\{ \sum_{i=0}^{L-1} c_{i+mL} b_{i+mL,B-1-j} \right\} \tag{12}$$

$$f_j = \sum_{m=0}^{M-1} \left\{ \sum_{i=0}^{L-1} c_{i+mL} b_{i+mL,B-1-j} \right\} \tag{13}$$

Hence, M number of LUTs can be used each of size 2^L, taking address lines from every K sets of x_i's. The DA structure corresponding to (13) with $L = 2$ and $B = 4$

is shown in Fig. 1b where the address bits are indexed by j, the LUT block is indexed by m and the LUT word is indexed by i. The number of clock cycles to compute the inner product remains the same, but the number of arithmetic operations (adders) would increase. Specifically, the system requires a total of $M \times 2^L$ LUT locations and $M - 1$ adders and computes the inner product in B clock cycles. $L = N$ is same as (6), and $L = 1$ case will have N individual LUTs with two locations in each. In order to increase the speed of the system, multiple bits of x_i's can be used in parallel as the address lines to the LUT at the cost of increase in its size. This is obtained by splitting B as $B = P \times Q$ in which case (8) becomes

$$y = \sum_{j=0}^{PQ-1} C_{B-1-j} 2^{-j} \tag{14}$$

$$= \sum_{j=0}^{Q-1} C_{B-1-j} 2^{-j} + \sum_{j=Q}^{2Q-1} C_{B-1-j} 2^{-j} + \cdots + \sum_{j=(P-1)Q}^{PQ-1} C_{B-1-j} 2^{-j} \tag{15}$$

$$= \sum_{p=0}^{P-1} \sum_{j=pQ}^{(p+1)Q-1} C_{i,j} 2^{-j} \tag{16}$$

If $s = j - pQ$ and if the dummy variable s is replaced by j, then

$$y = \sum_{p=0}^{P-1} \sum_{s=0}^{Q-1} C_{i,s+pQ} 2^{-(s+pQ)} \tag{17}$$

The DA structure corresponding to (17) with $P = 2$ is shown in Fig. 1c. In such case, the size of the memory would be 2^{PN} and the speed is increased by a factor of P, and hence the inner product is computed in B/P clock cycles. Further it can be noted from (10) that the shifting is done for P units unlike the previous case where unit shifting is done on each of the partial product. If $P = B$, then all the B-bits of x_i's form the address lines and the inner product is computed in a single clock cycle while $P = 1$ is same as (8).

Hence, from (13) and (17), it can be noted that splitting N would result in the low memory-size requirement at the expense of increased combinational logic and employing digit-serial would speed up the system at the cost of increased parallelism. Alternately, one can employ the ROM splitting as well as the digit-serial nature of DA as shown in Fig. 1d for the trade-offs between speed and hardware complexity based on the parameters M and P as explained above. These techniques apply equally well when the partial products of x_i's are stored and the bits of c_i's are used as the address bits to the LUT. Further, the techniques work well for the signed 2's-complement representation of the elements of the vectors [15].

The ROM size in the basic DA architecture can be further reduced using the offset-binary coding (OBC) technique [15], which can be derived as follows:

Rewriting (2) as $x_{n-i} = \frac{1}{2}[x_{n-i} - (-x_{n-i})]$ we have:

$$x[n-i] = \frac{1}{2}\left[-\left(b_{i,B-1} - \bar{b}_{i,B-1}\right)\right] + \frac{1}{2}\left[+\sum_{j=1}^{B-1}\left(b_{i,B-1-j} - \bar{b}_{i,B-1-j}\right)2^{-j} - 2^{-(B-1)}\right] \tag{18}$$

Choosing

$$d_{i,j} = \begin{cases} -(b_{i,j} - \overline{b_{i,j}}), & j \neq B-1 \\ -(b_{i,B-1} - \overline{b_{i,B-1}}), & j = B-1 \end{cases} \tag{19}$$

Substituting (18), (19) in (1) and rearranging, we get

$$y[n] = \sum_{j=0}^{B-1}\left(\sum_{i=0}^{N-1}\frac{1}{2}c_i d_{i,B-1-j}\right)2^{-j} - \left(\frac{1}{2}\sum_{i=0}^{N-1}c_i\right)2^{-(B-1)}$$

Defining

$$C_{B-1-j} = \sum_{i=0}^{N-1}\tfrac{1}{2}c_i d_{i,j},\ 0 \leq j \leq B-1$$

and

$$C_{\text{initial}} = -\frac{1}{2}\sum_{i=0}^{N-1}c_i \tag{20}$$

We arrive at

$$y[n] = \sum_{j=0}^{B-1}C_{B-1-j}2^{-j} + C_{\text{initial}}2^{-(B-1)} \tag{21}$$

Now, for a given set of w_i ($i = 0, 1, \ldots, N-1$), the terms C_{B-1-j}'s would take one out of 2^N combinations, half of which would be the mirror image of other half [31]. Hence a 2^{N-1}-sized ROM can be used the address bits of which can be obtained through the Ex-OR operation of all the LSB's with the LSB of the newest sample as shown in Fig. 1e.

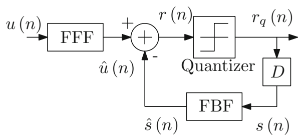


Fig. 2 General block diagram of a decision feedback equalizer

3 Proposed DA-Based Architecture

Consider a DFE shown in Fig. 2, with 'N_f' number of FFF coefficients and 'N_b' number of FBF coefficients which process the input signal $u(n)$, $(n \in Z)$, and the previous output decisions $s(n)$, respectively. The equations describing the operation of the DFE are

$$r_q(n) = Q[r(n)] \tag{22}$$

where $Q[.]$ is the quantization operation.

$$r(n) = \hat{u}(n) - \hat{s}(n) \tag{23}$$

$$s(n) = r_q(n-1) \tag{24}$$

and the output of FFF and FBF filters respectively is given as follows:

$$\hat{u}(n) = \sum_{i=0}^{N_f-1} w_i u(n-i) = \mathbf{w}^T \mathbf{u} \tag{25}$$

$$\hat{s}(n) = \sum_{j=0}^{N_b-1} v_j s(n-j) = \mathbf{v}^T \mathbf{s} \tag{26}$$

where $\mathbf{w}^T = [w_0, w_1, \ldots, w_{N_f-1}]$, $\mathbf{v}^T = [v_0, v_1, \ldots, v_{N_b-1}]$ are the coefficients of FFF and FBF respectively.
If each of $u(n)$ and $s(n)$ is represented by their signed 2's-complement representation, we have:

$$u(n-i) = -u_{i,B-1} + \sum_{b=1}^{B-1} u_{i,B-1-j} 2^{-j} \tag{27}$$

$$s(n-i) = -s_{i,B-1} + \sum_{b=1}^{B-1} s_{i,B-1-j} 2^{-j} \tag{28}$$

Employing the digit-serial arithmetic, using (17) we have

$$\begin{aligned}\hat{u}(n) = &\left[-f\left(w_i, u_{i,0}\right)2^0 + \sum_{j=1}^{P_f-1} f\left(w_i, u_{i,P_f-1-j}\right)2^{-j}\right]\left(2^{P_f}\right)^{Q_f-1}\\ &+ \sum_{k=0}^{Q_f-2}\left\{\sum_{j=0}^{P_f-1} f\left(w_i, u_{i,P_f-1-j}\right)2^{-j}\right\}\left(2^{P_f}\right)^k\end{aligned} \tag{29}$$

where $f\left(w_i, u_{i,P_f-1-j}\right)$ is given by

$$f\left(w_i, u_{i,P_f-1-j}\right) = \sum_{i=0}^{N_f-1} w_i u_{i,P_f-1-j} \tag{30}$$

Now, if the ROM decomposition technique is also employed, then using (5) and (6) we have

$$f\left(w_i, u_{i,P_f-1-j}\right) = \sum_{l=0}^{L_f-1}\sum_{i=lM_f}^{lM_f+M_f-1} w_i u_{i,P_f-1-j} \tag{31}$$

Hence, Eqs. (29) and (31) describe the realization of FFF using both digit-serial input and ROM decomposition techniques. Similarly, for FBF we can have

$$\begin{aligned}\hat{s}(n) = &\left\{-f\left(v_i, s_{i,0}\right)2^0 + \sum_{j=1}^{P_b-1} f\left(v_i, s_{i,P_b-1-j}\right)2^{-j}\right\}\left(2^{P_b}\right)^{Q_b-1}\\ &+ \sum_{k=0}^{Q_b-2}\left\{\sum_{j=0}^{P_b-1} f\left(v_i, s_{i,P_b-1-j}\right)2^{-j}\right\}\left(2^{P_b}\right)^k\end{aligned} \tag{32}$$

$$f\left(v_i, s_{i,P_b-1-j}\right) = \sum_{l=0}^{L_b-1}\sum_{i=lM_b}^{lM_b+M_b-1} v_i s_{i,P_b-1-j} \tag{33}$$

The DA-based DFE architecture derived based on the above equations is shown in Fig. 3. It can be seen that each of the filters has their own DA block that works on both digit-serial and ROM decomposition techniques. The parameters P_f and P_b decide the number of digits used from each of input register for FFF and FBF, respectively. Similarly, the parameters M_f and M_b decide the number of ROMs operated in parallel for FFF and FBF, respectively. The filters FFF and FBF can be synchronized by choosing same number of digits from each of the input register i.e. by choosing $P_f = P_b$. Hence, the output of FFF and FBF can be computed using just the shift and add operations, which makes the DFE free of hardware multipliers. The detailed internal structure of the FFF and/or FBF blocks in all the three cases, namely direct-memory architecture, reduced-memory architecture and OBC-memory architecture,

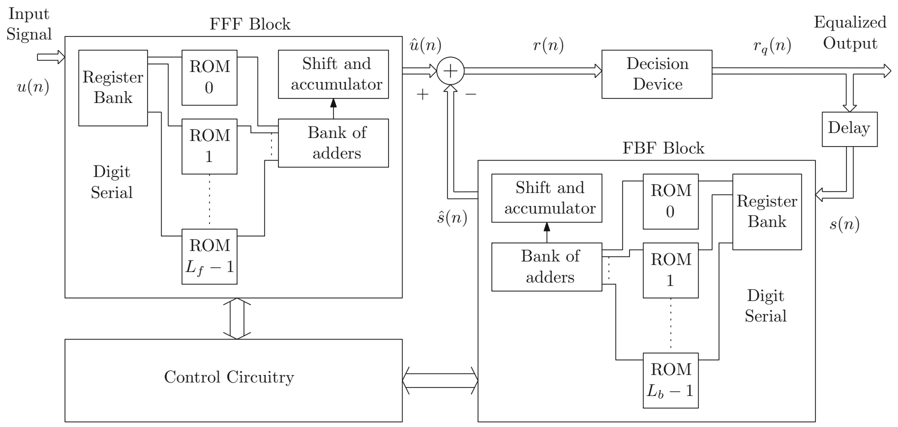


Fig. 3 Proposed DA-based decision feedback equalizer architecture that employs ROM decomposition and digit-serial input

are shown in Fig. 4. The internal structure consists of a register bank, a set of memory blocks, bank of adders and a shift-accumulator block. Input bits enter the register bank serially which form the address bits to the memories. The adder bank consists of a set of adders where the outputs from the memories are summed up. The result is then shift-accumulated to compute the output. Figure 4 shows the case where $P = B$; however, the parameters P and L can be chosen based on the speed and complexity requirements. It can be seen from Fig. 4a that the contents of upper half of the memories look like the mirror image of the lower half, and hence a memory storing only one half of the contents can be used to perform DA filtering. Specifically, one half of the partial products of any memory in case of direct-memory architecture are nothing but the additive inverse of the other half. Hence, only one half of the partial products may be stored and the other half may be obtained live by taking their complement. This operation is depicted in Fig. 4b by the NOT gate used at the output of each ROM. The idea is particularly useful when the requirement for the memory size is large. When the 'upper-half' set of the partial products are stored in the memory, the lower half can be generated by using a multiplexer (MUX) as shown in Fig. 4b. Here, the most significant bit (MSB) is neglected in forming the address bits to the memory, instead it is used to find the effective address using the Ex-OR operation with each of the remaining address bits. In case of OBC-memory-based architecture, the size of the memories would be same as that of the reduced-memory architecture but contain more complex combinations of partial products. Further it needs an extra adder and a register for the storage and addition of the C_{initial} term to the accumulator output. The decision device takes an input signal that lies between a set of signal levels and quantizes it to a set of pre-defined levels that depends on the modulation scheme employed. The control circuitry sends the required timing signals for the operation of FFF and FBF.

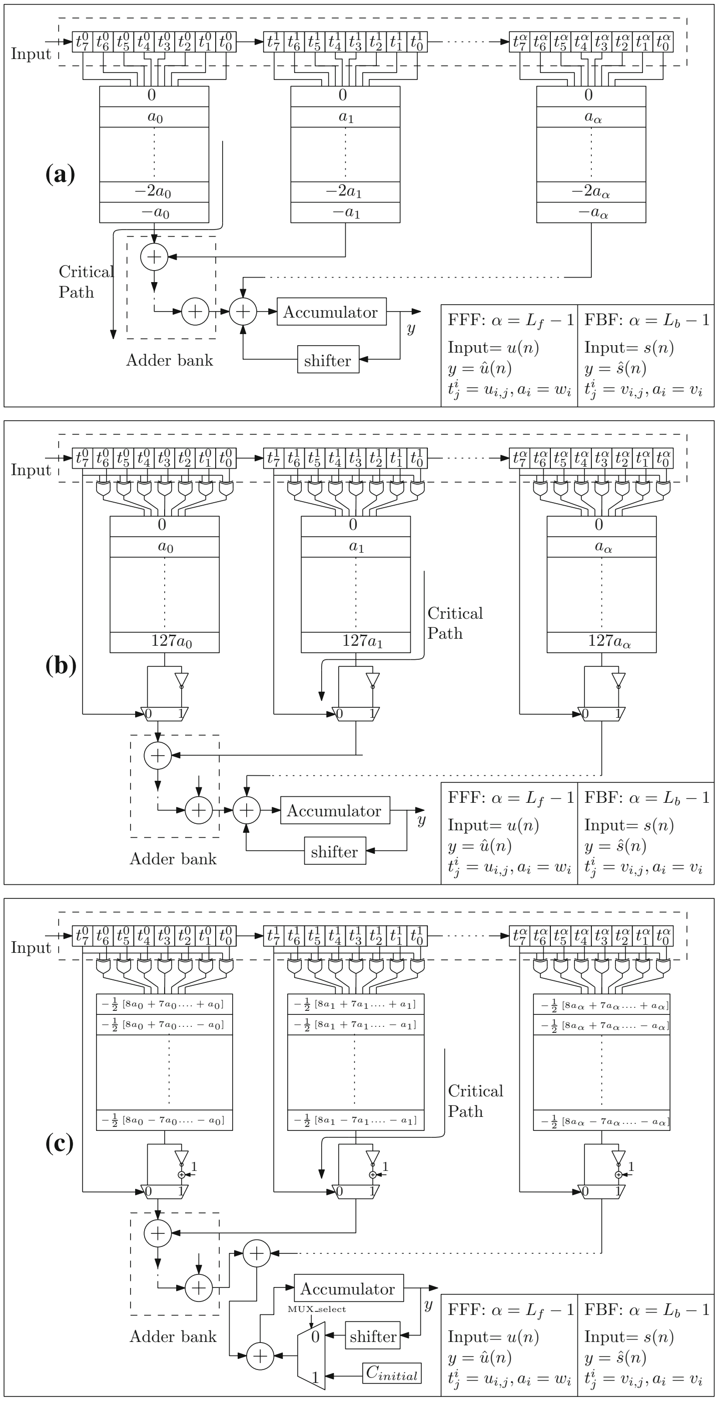


Fig. 4 The internal structure of FFF and/or FBF blocks in the proposed architectures. **a** Direct-memory architecture. **b** Reduced-memory architecture. **c** OBC-memory architecture

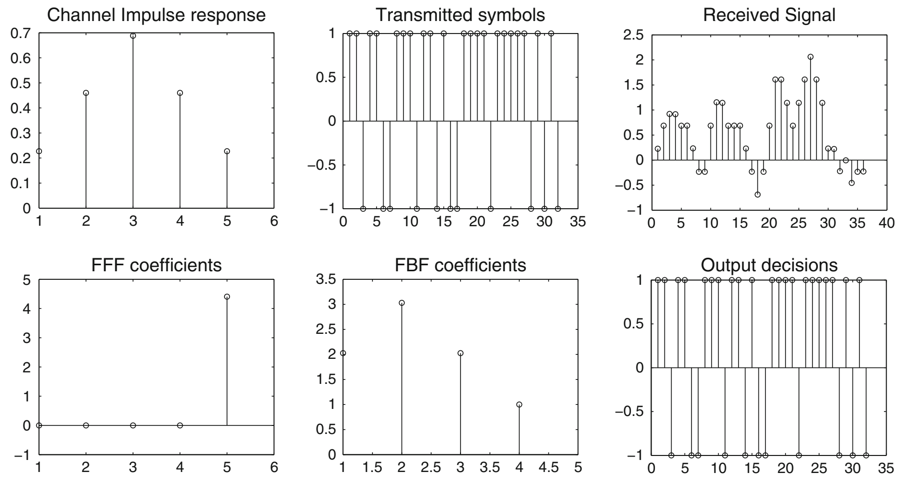


Fig. 5 A typical example of a DFE-based channel-equalizer system. The equalized output mimics the transmitted symbols assuming thtat there are no errors in the past decisions

4 Implementation Results

In order to test the proposed architecture, a DFE-based channel-equalizer system has been created like the one shown in Fig. 5 where a frequency-selective channel is taken. A set of message signals each modulated using binary phase shift keying (BPSK) has been transmitted. The received symbols (containing the ISI) are then passed through the FFF for the removal of anti-causal part of ISI. The remaining ISI is then cancelled out using the decisions taken on previous symbols, and as observed in Fig. 5, the original transmitted symbols can be retained if there is no error propagation in the DFE. For the ease of analysis, we chose $N_b = N_f - 1$. The input to the DFE and the weights of FFF and FBF are chosen to be fixed point Q2.6 signed 2's-complement representation, and the system is implemented using verilog HDL. To attain the maximum speed, P_f and P_b are chosen to be eight and the outputs of each memory are summed up by the adder bank, and hence one sample of output of both FFF and FBF is computed in one clock cycle. The decision device quantizes its input to either of the two symbols (as BPSK scheme is used). A rough estimations and comparisons of number of logic elements[1], maximum usable frequency ($F_{\max}$) and core static power consumption estimates for the MAC-based and DA-based implementations for various FBF taps implemented on Altera Cyclone III EP3C55F484C6 at different operating conditions (0, 85 °C) are summarized in Tables 1, 2 and 3, respectively. From the table, it can be observed that the DA-based implementation outperforms the MAC-based implementation in all the cases. For the case of 4-tap FBF, the difference in the number of logic elements for DA-based and MAC-based implementations is very less; however, the maximum

[1] Logic elements (LEs) are the smallest units of logic in the Cyclone III device family architecture. Each logic element consists of four input lookup table, a programmable register and many other features. (www.altera.com/literature/hb/cyc3/cyc3_ciii51002)

Table 1 Comparison of number of logic elements for MAC-based and DA-based implementations on Altera Cyclone III EP3C55F484C6

Size of FBF	MAC-based architecture	DA-based implementations		
		Direct-memory architecture	Reduced-memory architecture	OBC-memory architecture
4-tap	420	249	337	481
8-tap	1007	495	646	891
16-tap	1775	984	1247	1708
32-tap	3312	1918	2482	3376

Table 2 Comparison of $F_{\max}$ (MHz) (0, 85 °C) in MHz for MAC-based and DA-based implementations on Altera Cyclone III EP3C55F484C6

Size of FBF	MAC-based architecture	DA-based implementations		
		Direct-memory architecture	Reduced-memory architecture	OBC-memory architecture
4-tap	60.35, 56.47	67.94, 60.87	68.09, 60.66	51.86, 46.12
8-tap	30.6, 28.13	38.2, 34.15	36.56, 32.77	31.0, 27.71
16-tap	15.36, 14.38	19.56, 17.51	19.46, 17.44	18.6, 16.64
32-tap	7.76, 7.21	10.24, 9.14	10.17, 9.06	9.73, 8.7

Table 3 Comparison of core static power consumption estimates in milli-watt (mW) for MAC-based and DA-based implementations on Altera Cyclone III EP3C55F484C6

Size of FBF	MAC-based architecture	DA-based implementations		
		Direct-memory architecture	Reduced-memory architecture	OBC-memory architecture
4-tap	51.70	94.01	94.02	94.02
8-tap	51.71	94.03	94.03	94.05
16-tap	51.74	94.05	94.07	94.09
32-tap	51.80	94.11	94.14	94.19

usable frequency ($F_{\max}$) is a bit morein case of DA-based implementation. Further, it can be seen that as the filter-order increases, the difference in the number of logic elements between the MAC-based and DA-based implementation is getting increased while the maximum usable frequency of DA-based implementation is approaching to that of MAC-based implementation. This can be explained as follows: For lower-order filters, the number of MAC units used in the MAC-based implementation is quite low, and therefore the number of memories replacing the multipliers in case of DA-based implementation will also be less. But as the filter order increases, the number of memories replacing the multipliers in DA-based implementation will be high, and hence the difference in the number of logic elements utilized will be high. The critical path (that decides the maximum usable frequency) of the FBF filter for the MAC-based and DA-based direct-memory implementation is given as $\mathrm{CP}_{\mathrm{MAC}} = T_m + \mathrm{NT}_A$ and $\mathrm{CP}_{\text{DA-direct-memory}} = T_{\text{memory}} + (N-1)\,T_A$ where T_{memory}, T_{m}, and T_{A} are the computation times of memory, multiplier and adder units respectively. For lower-order filters, because of the presence of less number of adder units, the critical path of MAC-based and DA-based implementation differs. As the filter order increases, the number of adders increase, and hence the critical path of the DA-based implementation approaches that of MAC-based implementation due to the fact that the terms T_{memory} and T_M become negligible compared to the term $(N-1)\,T_A$. In case of reduced-memory architecture, even though the memory size is halved, the additional hardware overhead is added by the Ex-OR gates, multiplexer and adder units. The number of Ex-OR gates is fixed for a fixed P (which depends on B), while the number of adder units and MUXs depend upon the value of M and P. The critical path in this case would be $\mathrm{CP}_{\text{DA-reduced-memory}} = T_{\text{reduced-memory}} + T_{\mathrm{MUX}} + \mathrm{NT}_A$. Since the computation time of a memory does not vary greatly with respect to size of it, the additional computation time of reduced-memory architecture is just $T_{\mathrm{MUX}} + T_A$ units. The scheme with OBC-DA-based memory uses slightly more number of logic elements in which case, the critical path would be $\mathrm{CP}_{\text{DA-reduced-memory}} = T_{\text{OBC-memory}} + T_{\mathrm{MUX}} + (N+1)T_A$.

5 Conclusion

A distributed arithmetic-based realization of the decision feedback equalizer has been presented. In order to attain the maximum speed, digit-serial input has been employed, apart from that ROM decomposition technique is also employed to eliminate the exponential increase in the size of memories as the FFF and FBF filter orders increase. Two architectures based on digit-serial and ROM decomposition variants of DA have been presented, and results show that both are efficient when compared to MAC-based implementation. A third architecture which uses DA-OBC scheme has also been presented which uses slightly more number of hardware resources compared to reduced-memory architecture. For lower-order filters, the maximum usable frequency is a bit high compared to that of MAC-based implementation while the number of logic elements is only few units less. For higher-order filters, the number of logic elements is found to be almost half that of the MAC-based implementation. The proposed architecture can be used in high-data-rate modems such as the case of IEEE 802.11b scenarios. The speed of the proposed implementation can be further increased by using techniques that reduce the number of adder units, which results in the reduction of critical path.

References

1. D.J. Allred, H. Yoo, V. Krishnan, W. Huang, D.V. Anderson, LMS adaptive filters using distributed arithmetic for high throughput. IEEE Trans. Circuits Syst. I Regul. Pap. **52**(7), 1327–1337 (2005)
2. M. Arjmand, R.A. Roberts, On comparing hardware implementations of fixed-point digital filters. IEEE Circuits Syst. Mag. **3**(2), 2–8 (1981)

3. W.P. Burleson, L.L. Scharf, A VLSI design methodology for distributed arithmetic. J. VLSI Sig. Process. Syst. Signal Image Video Technol. **2**(4), 235–252 (1991)
4. W.P. Burleson, L.L. Scharf, A VLSI implementation of a cellular rotator array, in *IEEE Custom Integrated Circuits Conference*, pp. 8.1/1–8.1/4 (1988)
5. W.P. Burleson, L.L. Scharf, VLSI design of inner-product computers using distributed arithmetic, in *IEEE International Symposium Circuits Systems (ISCAS)*, pp. 158–161 (1989)
6. C.S. Burrus, Digital filter structures described by distributed arithmetic. IEEE Trans. Circuits Syst. **24**(12), 674–680 (1977)
7. C.P. Callender, S. Theodoridis, C.F.N. Cowan, Adaptive non-linear equalisation of digital communications channels. Signal Process. **40**(2–3), 325–333 (1994)
8. H.C. Chen, J.I. Guo, C.W. Jen, A new group distributed arithmetic design for the one dimensional discrete Fourier transform, in *IEEE International Symposium Circuits Systems (ISCAS)*, vol. 1 (2002)
9. H.C. Chen, J.I. Guo, C.W. Jen, T.S. Chang, Distributed arithmetic realisation of cyclic convolution and its DFT application, in *IEEE Proceedings on Circuits, Devices and Systems*, pp. 615–629 (2005)
10. C.F.N. Cowan, J. Mavor, New digital adaptive-filter implementation using distributed-arithmetic techniques. IEEE Proc. Commun. Radar Signal Process. **128**(4), 225–230 (1981)
11. S.P. Joshi, R. Paily, Distributed arithmetic based Split-Radix FFT. J. Signal Process. Syst. **75**(1), 85–92 (2014)
12. P.K. Meher, Hardware-efficient systolization of DA-based calculation of finite digital convolution. IEEE Trans. Circuits Syst. II Exp. Br. **53**(8), 707–711 (2006)
13. P.K. Meher, Unified systolic-like architecture for DCT and DST using distributed arithmetic. IEEE Trans. Circuits Syst. I Reg. Pap. **53**(12), 2656–2663 (2006)
14. B.K. Mohanty, P.K. Meher, A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm. IEEE Trans. Signal Process. **61**(4), 921–932 (2013)
15. K.K. Parhi, *VLSI Digital Signal Processing Systems: Design And Implementation* (Wiley India Pvt, Limited, 2007)
16. S.Y. Park, P.K. Meher, Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic. IEEE Trans. Circuits Syst. II Exp. Br. **60**(6), 346–350 (2013)
17. S.Y. Park, P.K. Meher, Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter. IEEE Trans. Circuits Syst. II Exp. Br. **61**(7), 511–515 (2014)
18. M.S. Prakash, R.A. Shaik, Low-area and high-throughput architecture for an adaptive filter using distributed arithmetic. IEEE Trans. Circuits Syst. II Exp. Br. **60**(11), 781–785 (2013)
19. J.G. Proakis, D.G. Manolakis, *Digital Communications, vol. 3* (McGraw-hill, New York, 1995)
20. S.U.H. Qureshi, Adaptive equalization. IEEE Commun. Mag. **73**(9), 1349–1387 (1985)
21. J. Ramìrez, A. Garcìa, U. Meyer-Bäse, F. Taylor, A. Lloris, Implementation of RNS-based distributed arithmetic discrete wavelet transform architectures using field-programmable logic. J. VLSI Signal Process. Syst. Signal Image Video Technol. **33**(1–2), 171–190 (2003)
22. M. Rawski, M. Wojtynski, T. Wojciechowski, P. Majkowski, Distributed arithmetic based implementation of Fourier transform targeted at FPGA architectures, in *International conference on Mixed Design of Integrated Circuits and Systems (MIXDES)*, pp. 152–156 (2007)
23. G. Seetharaman, B. Venkataramani, G. Lakshminarayanan, Design and FPGA implementation of self-tuned wave-pipelined filters with distributed arithmetic algorithm. Circuits Syst. Signal Process. **27**(3), 261–276 (2008)
24. R.A. Shaik, M. Chakraborty, An efficient realization of the decision feedback equalizer using block floating point arithmetic, in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 1047–1050 (2006)
25. R.A. Shaik, M. Chakraborty, A block floating point treatment to finite precision realization of the adaptive decision feedback equalizer. Signal Process. **93**(5), 1162–1171 (2013)
26. G. Sicuranza, G. Ramponi, Adaptive nonlinear digital filters using distributed arithmetic. IEEE Trans. Acoust. Speech Signal Process. **34**(3), 518–526 (1986)
27. S.G. Smith, S.A. White, Hardware approaches to vector plane rotation, in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 2128–2131 (1988)
28. B.S. Tan, G.J. Hawkins, Speed-optimised microprocessor implementation of a digital filter. IEEE Proc. Comput. Dig. Tech. **128**(3), 85–93 (1981)
29. L. Wanhammar, Implementation of wave digital filters using distributed arithmetic. Signal Process. **2**(3), 253–260 (1980)

30. C.H. Wei, J.J. Lou, Multimemory block structure for implementing a digital adaptive filter using distributed arithmetic. IEEE Proc. Electron. Circuits Syst. **133**(1), 19–26 (1986)
31. S.A. White, Applications of distributed arithmetic to digital signal processing: a tutorial review. IEEE Acoust. Speech Signal Process. Mag. **6**(3), 4–19 (1989)
32. J. Xie, P.K. Meher, J. He, Hardware-efficient realization of prime-length DCT based on distributed arithmetic. IEEE Trans. Comput. **62**(6), 1170–1178 (2013)
33. S. Yu, E.E. Swartziander, DCT implementation with distributed arithmetic. IEEE Trans. Comput. **50**(9), 985–991 (2001)
34. J. Zeman, H.T. Nagle, A high-speed microprogrammable digital signal processor employing distributed arithmetic. IEEE J. Solid-State Circuits **15**(1), 70–80 (1980)