A Digital Adaptive Filter Using a Memory-Accumulator Architecture: Theory and Realization

COLIN F. N. COWAN, STEWART G. SMITH, AND JAMES H. ELLIOTT

Abstract—**This paper presents a novel approach to the construction of an adaptive filter making use of the so-called "distributed arithmetic" filter architecture originally suggested by Peled and Liu [7] for the realization of fixed response digital frequency filters. The technique uses only the operations of memory access, addition, and scaling, without the need for digital multiplication. Since multiplication is often quoted as the major bottleneck in digital signal processing structures, the system derives considerable advantage by the exclusion of this operation.**

The paper presents the derivation of a new adaptive algorithm based on this particular hardware structure, although no rigorous theoretical proof of the algorithm convergence properties is given. Computer simulations are included to demonstrate some of the basic operational characteristics of the structure. Finally, results from a hardware prototype, constructed using standard TTL integrated circuits, are presented.

This approach differs from contemporary ideas which depend on the use of digital multipliers in either custom VLSI designs or using standard signal processing chips. It offers high-bandwidth operation at low cost using devices which are already in great demand by the computer market. Alternatively, the algorithm is ideal for implementation as a microprocessor-based system which could operate on real-time voice-bandwidth signals with a minimum of peripheral interface circuitry.

I. Introduction

THE BASIC theory of adaptive filters using finite impulse response (FIR) filters has been known and well understood for many years. One of the most commonly used adaptive algorithms is that attributed to Widrow [1], the Widrow least mean-square (LMS) algorithm. Many implementations of FIR adaptive filters using the LMS algorithm have been proposed using both classical digital filter structures [2], [3] and using analog bucket brigade device (BBD) and charge-coupled device (CCD) technologies [4]-[6].

These implementations all make use of a literal interpretation of the FIR filter convolution expression, hence using linear multiplications. In analog implementations the use of linear multipliers does not involve a heavy penalty in terms of silicon usage or power consumption; however, limitations in dynamic range must be accepted if analog techniques are to be used [5]. Digital implementations, on the other hand, have arbitrary dynamic range dependent only on the word size chosen by the designer. This dynamic range, however, must be paid for in terms of the complexity of digital multipliers. The digital multiplier has long been recognized as a major bottleneck in the design of digital signal processing (DSP) systems and a standard figure of merit used in evaluating the complexity of a DSP system is the number of multiplications/s required.

In this paper we present a technique for the implementation of a digital adaptive filter which uses no digital multiplication, but instead relies on the use of the operations of memory access, addition and scaling by integer powers of 2. The technique is based on the distributed arithmetic (alternatively ROM/accumulator) filter architecture originally proposed for fixed frequency filter implementation by Peled and Liu [7].

This technique has the advantage that only standard TTL type logic circuits need to be used without resource to specialised signal processing functions (such as hardware multipliers). The architecture is well suited to high-bandwidth implementations using Schottky TTL components, though the algorithm could equally be implemented using microprocessors for voice-bandwidth implementations.

The architecture has been reported previously for the special case of a single-bit input signal [8], [9]. This theory has been extended to the multibit case [10], thus introducing a generalized adaptive algorithm. In this paper the adaptive algorithm of [10] has been refined and extensively studied through both computer simulations and observation of a hardware prototype in various application modes.

In Sections II and III a theoretical development of the adaptive algorithm is presented, which is derived using the Widrow LMS algorithm. Computer simulations are presented in Section IV to demonstrate some of the basic operational characteristics of the system including the application of the filter to the equalization of a telephone data channel. Finally, the design of a hardware prototype module is presented with results from this module demonstrating, principally, the convergence and noise cancellation properties of the filter.

The techniques presented in this paper provide an alternative approach to the design of digital adaptive filters. The architectures proposed allow possible tradeoffs between low-cost low-power consumption systems and high-bandwidth implementations suitable for use up to video frequencies. The distributed arithmetic based algorithm, then, presents a viable alternative to custom VLSI designs of adaptive digital filters [3], [6].

II. Theoretical Development

In this section the normal equations for the convolution of two signals are rearranged in such a way that they may be

Manuscript received August 10, 1982; revised October 29, 1982.

C. F. N. Cowan and S. G. Smith are with the Department of Electrical Engineering, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JL, Scotland.

J. H. Elliott is with Hewlett-Packard Ltd., South Queensferry, West Lothian, Scotland.

0096-3518/83/0600-0541$01.00

easily restructured to implement an adaptive algorithm. The discussion following in Section III shows how the Widrow LMS algorithm is applied in this instance.

In the following discussion, all one-dimensional matrices are represented by column vectors. The superscript T refers to the transpose of a column vector or matrix, and primed vectors represent updated vectors. The following scalars are defined:

$h(n) \equiv$ the nth coefficient of an N-point digital transversal filter;

$f(k) \equiv$ the kth partial product used in the output accumulation of a distributed arithmetic filter;

$s(n) \equiv$ the K-bit signal sample present at point n of an N-point digital filter;

$y \equiv$ the output sample produced by a digital filter;

$d \equiv$ the training or conditioning signal sample presented to a digital adaptive filter;

$e \equiv d - y \equiv$ the error sample produced by a digital adaptive filter.

The following matrices are defined:

$$\boldsymbol{S}^T \equiv (s(1), s(2), \cdots, s(n), \cdots, s(N))$$

$$\boldsymbol{H}^T \equiv (h(1), h(2), \cdots, h(n), \cdots, h(N))$$

$$\boldsymbol{F}^T \equiv (f(1), f(2), \cdots, f(k), \cdots, f(K))$$

$\boldsymbol{X}^T \equiv (2^{-1}, 2^{-2}, \cdots, 2^{-k}, \cdots, 2^{-K})$, i.e., the set of the first K negative integer powers of 2;

$\boldsymbol{B} \equiv$ the $N \times K$ array of bit values which results when a K-bit input signal vector is stored in an N-point digital filter. $\boldsymbol{B}$ is merely $\boldsymbol{S}$ decomposed into component bits.

For the sake of simplicity, the variable t, denoting time-dependence, has been omitted from the above expressions. With the exception of the static weighting vector $\boldsymbol{X}$, it can be assumed that all of the defined quantities are time-dependent, although $\boldsymbol{F}$ and $\boldsymbol{H}$ are only so in the case of an adaptive filter.

Consider a set of N registers, each of length K bits. These registers provide storage for an $N \times K$ array of bit values, and this array can be represented by the $N \times K$ matrix of binary values $\boldsymbol{B}$. A classical N-point transversal filter stores a signal vector $\boldsymbol{S}$ containing N K-bit components. If the signal is coded in offset binary, wherein logical 0 takes the value -1, the signal vector can be expressed as

$$\boldsymbol{S} = \boldsymbol{BX}. \tag{1}$$

The column vector $\boldsymbol{H}$ represents the set of N filter coefficients, and thus the output of the filter is given by the convolution

$$y = \boldsymbol{S}^T\boldsymbol{H}. \tag{2}$$

Substituting (1) in (2), the output is now

$$y = (\boldsymbol{BX})^T\boldsymbol{H} \tag{3}$$

and due to a property of the transpose of a product of matrices, (3) may be expressed as

$$y = \boldsymbol{X}^T\boldsymbol{B}^T\boldsymbol{H}. \tag{4}$$

If we now define column vector $\boldsymbol{F}$ as

$$\boldsymbol{F} = \boldsymbol{B}^T\boldsymbol{H} \tag{5}$$

we may substitute (5) in (4) to produce the definitive matrix expression for the output of a distributed arithmetic filter

$$y = \boldsymbol{X}^T\boldsymbol{F} \tag{6}$$

where vector $\boldsymbol{F}$ is the set of partial products used in the output accumulation.

Equations (2) and (6) reveal the difference between the output expressions for the classical transversal filter and the distributed arithmetic filter. Both are vector inner products, but while (2) is a summation over the N filter points, (6) is a summation over the K bit-planes.

An insight into hardware implementation is afforded by inspection of (4). The filter coefficient vector $\boldsymbol{H}$ undergoes two transformations as we move from right to left. Firstly, it is premultiplied by each row vector of $\boldsymbol{B}^T$ to produce the K components of partial product vector $\boldsymbol{F}$. As these row vectors contain N binary values, it follows that there can only exist 2^N distinct row vectors, and hence 2^N distinct partial products. This set can be stored in a ROM of length 2^N and accessed by the set of row vectors of $\boldsymbol{B}^T$. It should be noted that partial products have no inherent weight. Weighting is provided by the second transformation of (4), premultiplication by vector $\boldsymbol{X}^T$.

Fig. 1 shows a basic hardware configuration for a fixed-response distributed arithmetic filter. Signal words from the input ADC (analog to digital converter) are presented in serial form to a set of N cascaded K-bit shift registers, each of which represents a filter point. As this bit stream passes through the shift registers, a set of K N-bit address words are generated on the ROM address bus. The ROM data word corresponding to each address word is scaled by 2^{-k} (i.e., right-shifted k bits) and accumulated. After K memory accesses, the accumulation is complete, and the output sample is latched on to the DAC (digital to analog converter.)

III. The Adaptive Algorithm

Conventional adaptive algorithms, based on the Widrow least mean-square (LMS) algorithm (1), operate directly on the filter coefficient vector $\boldsymbol{H}$. Such algorithms are unsuitable for distributed arithmetic realisation, in which the filter coefficients do not appear in explicit form. Cowan and Mavor (10) have proposed a new adaptive algorithm tailored specifically for distributed arithmetic implementation, as it operates directly on the partial product vector $\boldsymbol{F}$.

A universal adaptive filter configuration, using the Widrow LMS algorithm (1), is shown in Fig. 2. The input signal vector $\boldsymbol{S}$ is filtered to produce output y. This output is compared with a training or conditioning signal d, and the resultant error e is used by the adaptive algorithm to update the filter coefficients in such a manner as to reduce the error. Using (2), the error signal may be expressed as

$$e = d - \boldsymbol{S}^T\boldsymbol{H}. \tag{7}$$

The Widrow LMS algorithm approximates the mean-squared error with the square of a single error sample. The coefficient vector $\boldsymbol{H}$ is updated by subtracting the gradient of the squared error with respect to $\boldsymbol{H}$ multiplied by a convergence facter μ which determines the accuracy and stability of the solution

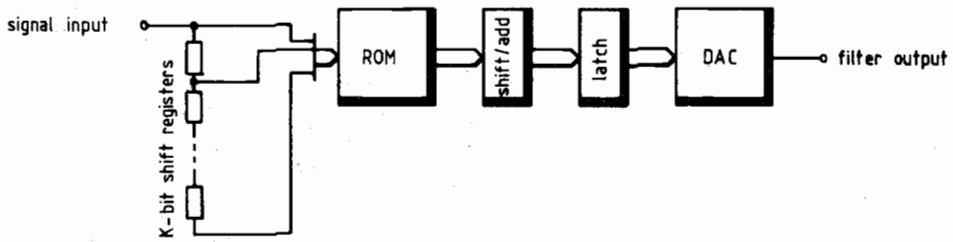


Fig. 1. Block diagram of a fixed response distributed arithmetic filter.

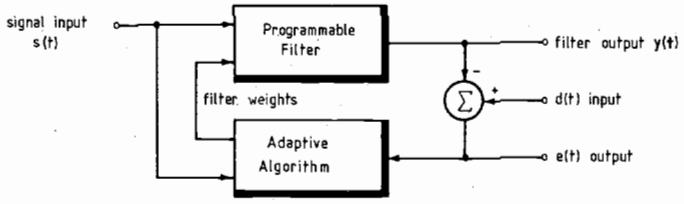


Fig. 2. Generalized adaptive filter block diagram.

for $\boldsymbol{H}$, and the speed of convergence to that solution. Hence the general form of the Widrow LMS algorithm, using primes to denote updated vectors,

$$\boldsymbol{H}' = \boldsymbol{H} - \mu\hat{\nabla} \tag{8}$$

where

$$\hat{\nabla} = \frac{\partial e^2}{\partial \boldsymbol{H}}$$

as stated above.

Squaring and partial differentiation of (7) followed by substitution in (8) leads to the linear Widrow LMS algorithm

$$\boldsymbol{H}' = \boldsymbol{H} + 2\mu e\boldsymbol{S}. \tag{9}$$

The Widrow LMS updating process is now applied to the distributed arithmetic filter. Equation (5) becomes

$$\boldsymbol{F}' = \boldsymbol{B}^T\boldsymbol{H}' \tag{10}$$

and substituting (9) gives

$$\boldsymbol{F}' = \boldsymbol{B}^T(\boldsymbol{H} + 2\mu e\boldsymbol{S}). \tag{11}$$

Using (1), (11) can be expressed as

$$\boldsymbol{F}' = \boldsymbol{B}^T(\boldsymbol{H} + 2\mu e\boldsymbol{B}\boldsymbol{X}) \tag{12}$$

which when factored out leads via (5) to

$$\boldsymbol{F}' = \boldsymbol{F} + 2\mu e\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{X}. \tag{13}$$

As the matrix $\boldsymbol{B}$ is $N \times K$, $\boldsymbol{B}^T\boldsymbol{B}$ is a $K \times K$ symmetric matrix, whose diagonal elements are equal to N. This is due to the offset binary signal coding, which implies that the square of any element of $\boldsymbol{B}$ is unity. If we assume that the input signal is zero-mean and that successive samples are uncorrelated [a common enough assumption in stochastic theory (11) and indeed in adaptive filter theory (1)], and further assume that the constituent bits of a signal sample are themselves uncorrelated; then the expectation of any nondiagonal element of $\boldsymbol{B}^T\boldsymbol{B}$ is zero, and the matrix reduces in the mean to the scalar N.

Hence the final matrix form of the algorithm

$$\boldsymbol{F}' = \boldsymbol{F} + 2\mu Ne\boldsymbol{X}. \tag{14}$$

The term $2\mu N$ can be treated as a constant, usually a negative integer power of 2. Its effect is therefore to add a constant shift of C bits, where $C = -\log_2(2\mu N)$, to the shift attributable to vector $\boldsymbol{X}$. Thus the update is simply the shifted error.

Fig. 3 shows a basic hardware configuration for the distributed arithmetic adaptive algorithm. The filter output sample is derived in the same manner as described earlier, although to allow adaptation the ROM of Fig. 1 has been replaced by a RAM. The output sample is subtracted from the training signal sample, and the resulting error word is right-shifted by C bits. The K N-bit address words are regenerated as before, but as well as being read from memory, the partial products are updated by adding in the shifted error further right-shifted by k bits, and written back to memory at the same address. After K memory accesses, the update cycle is complete, and the filter is ready to produce the next output sample.

An interesting symmetry property of partial products can increase the hardware efficiency of the filter. Recall that the address words used in each filter output accumulation are simply the row vectors which make up the matrix $\boldsymbol{B}^T$. As each address word contains N bits (one for each input signal sample or filter point), it was stated that the RAM must be of length 2^N to accomodate the set of partial products corresponding to the set of all possible address words. However, the symmetry property of the partial products renders half of the memory space redundant, and in fact hardware realizations of the distributed arithmetic filter need only use a memory of length 2^{N-1}.

The full matrix expression for the output of a distributed arithmetic filter is given by (4)

$$y = \boldsymbol{X}^T\boldsymbol{B}^T\boldsymbol{H}.$$

The following expression can be seen to be equivalent to (4):

$$y = \boldsymbol{X}^T(-(-\boldsymbol{B}^T)\boldsymbol{H}).$$

While this step appears trivial in terms of matrix algebra, it has some significance as regards hardware efficiency. The implication is that the vector of partial products accessed by input signal bit matrix $\boldsymbol{B}^T$ is simply the additive inverse of the vector accessed by $-\boldsymbol{B}^T$. The benefits of this are twofold; reduction in memory size and increase in convergence speed.

Each address word generated by the input shift registers has a redundant complement. As the only constraint on address-

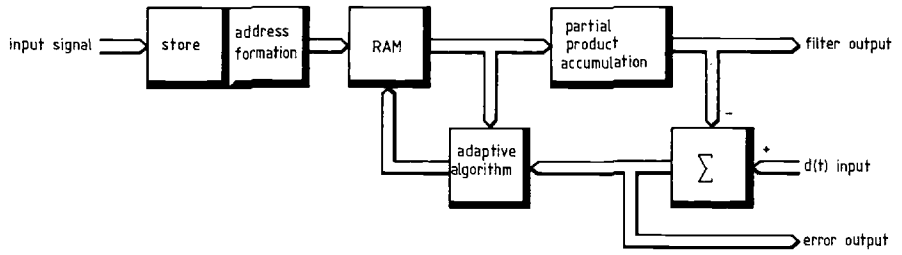


Fig. 3. System diagram for an adaptive distributed arithmetic filter.

ing is that selected memory contents should be updated with the same weight as they carried in the output accumulation, the address bus may be configured in a completely arbitrary manner. Any bit from the address bus may be separated from the bus, being used instead as an ADD/SUBSTRACT instruction for output accumulation and update of partial products, and when in SUBTRACT mode to invert the reduced address word. The consequence of this reduction in address word length is a 50 percent saving in memory size, and as each partial product is updated in the mean twice as often as in the basic hardware configuration, convergence is achieved in half the time.

A disadvantage inherent in memory reduction is that the filter can no longer compensate for dc offset in the training signal. In the full-memory filter, each partial product would undergo an increase in value equal to the dc offset during the adaptive process. However, if dc imbalance is considered a problem, an extra hardware register updated by the adaptive algorithm can provide the dc compensation. The contents of this register should be loaded into the output accumulator (which is normally cleared) at the start of the filter cycle. This is the equivalent of incorporating a dc tap in a classical adaptive filter.

IV. Computer Simulation Results

A software model of the filter was created which allowed user specification of convergence factor and filter length, as well as all digital word lengths within the filter. The use of integer arithmetic throughout enabled comprehensive simulation of the hardware at bit level. The user could create and edit signal files, each containing two signals which could be specified in terms of additive components, both deterministic and stochastic. These signals were then routed through optional delaying and distorting mechanisms to the filter inputs. Thus the performance of the model under a wide range of input conditions could be closely monitored.

Results from computer simulations are presented in this section for the following:

1) convergence on a sinusoidal input,
2) comparison of filter models with and without memory reduction,
3) system modeling using white noise,
4) cascading of filter sections, and
5) channel equalisation using DFB (decision feedback).

All the results presented stem from a filter model whose input and output bus widths were 8 bits, and whose memory depth was 24 bits. Saturation logic was applied to the update adder, where over/underflow is possible, and counts of over/underflows were taken. With the exception of the comparison test, memory reduction was used throughout.

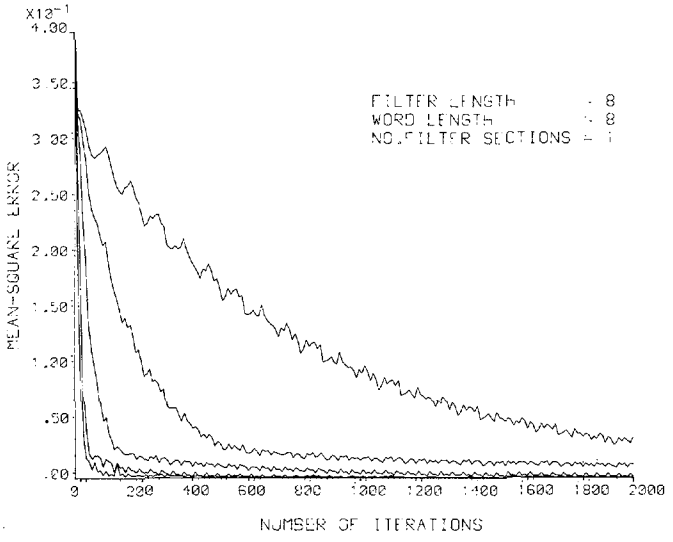


Fig. 4. Computer simulations showing the convergence of an 8-point distributed arithmetic filter to sinusoidal input signals. The convergence factors used are: 2^{-1}, 2^{-3}, 2^{-5}, 2^{-7}, and 2^{-9}.

Fig. 4 shows MSE (mean-square error) plots for various convergence factors. A sinusoid whose period was a noninteger multiple of the sampling period was applied to both signal and training inputs. The period was chosen thus to distribute addressing evenly over the entire RAM. The convergence factor μ was varied from 2^{-9} to 2^{-1} in exponential steps of 2. As expected, convergence time decreases with increasing μ.

Fig. 5(a) and (b) confirms the superior performance of the filter model using memory reduction over the full memory version, under the same sinusoidal input conditions. The result shows that, using the memory reduction algorithm, convergence is achieved in half the time required by the full algorithm, as predicted in our previous discussion.

The original filter configuration was then tested under system modelling conditions. The signal input was supplied with noise having a rectangular statistical distribution, and the training input with the same noise passed through a simulated eight-point transversal filter whose impulse response G was defined as

$$G^T = (0.5, 0, 0, -0.25, 0, 0, 0.125, 0).$$

Fig. 6(a) shows the MSE plot for $\mu = 2^{-3}$. Convergence is achieved after approximately 500 iterations, and the final

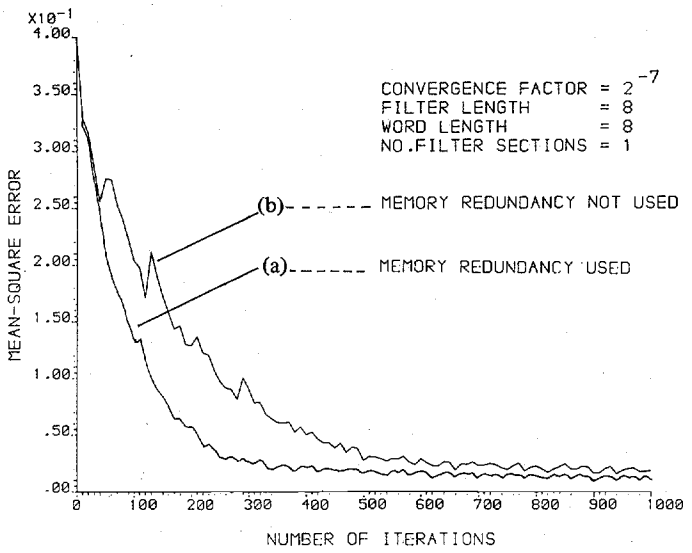


Fig. 5. (a) Convergence of a distributed arithmetic adaptive filter to sinusoidal inputs when the memory redundancy algorithm is used (convergence factor = 2^{-7}). (b) Convergence for the same input conditions and convergence factor without the use of the memory redundancy algorithm.

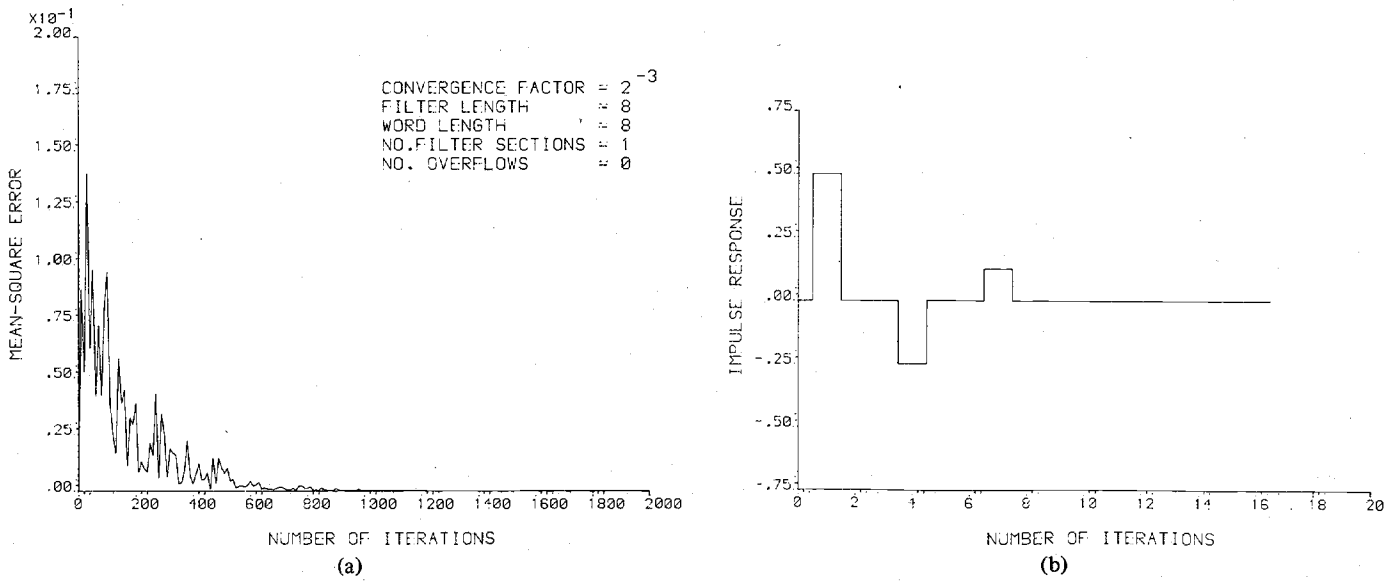


Fig. 6. (a) Trace showing the convergence of an 8-point distributed arithmetic filter in a system modeling situation using random input signals. (b) Impulse response of the filter after convergence.

impulse response, shown in Fig. 6(b), compares favorably with the modeled system.

An important point concerning the impulse response of a distributed arithmetic filter arises at this stage. If, during convergence of the filter, adaptation is halted and an impulse applied, the filter output will represent its impulse response. However, the distributed arithmetic filter differs from its classical counterpart in that, during adaptation, its transfer function cannot be equated with its impulse response. This is due to the fact that an impulse accesses only a small subset of the full set of partial products, and the response of the system to a random input cannot consequently be predicted from the impulse response.

It was stated earlier that the filter coefficient vector $\boldsymbol{H}$ does not exist in explicit form in the distributed arithmetic structure, and that any bit from the RAM address bus may be separated from the bus and used instead as an ADD/SUBTRACT instruction. When the filter approaches convergence on a random input signal, it can be assumed that the error in each partial product is small and zero-mean. The partial products form a complete set of linear equations in $\boldsymbol{H}$, i.e., any partial product can be expressed as

$$f = \sum_{n=1}^{N} b_n h_n$$

where the b_n can take the value 1 or -1. It can be shown

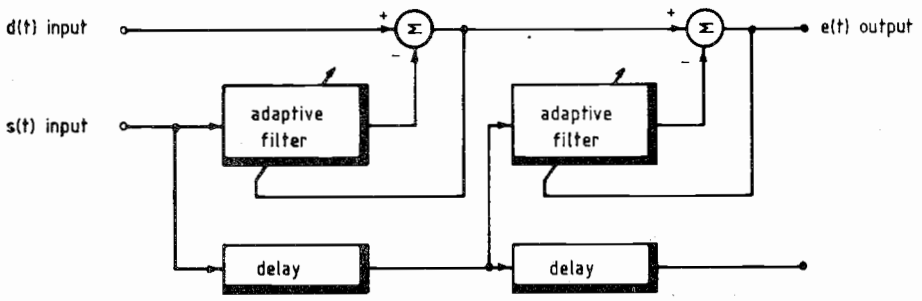


Fig. 7. Block diagram showing one possible means of cascading distributed arithmetic adaptive filters.

that accumulation of the entire set of partial products, using bit b_n as the ADD/SUBTRACT bit, will solve the equations for filter coefficient h_n.

The offset binary coding of the address words ensures that the product of the ADD/SUBTRACT bit with the bit value b_n associated with h_n (i.e., itself) will always be unity, whereas its product with the other bit values will sum to zero over the accumulation. Thus the accumulation reduces to

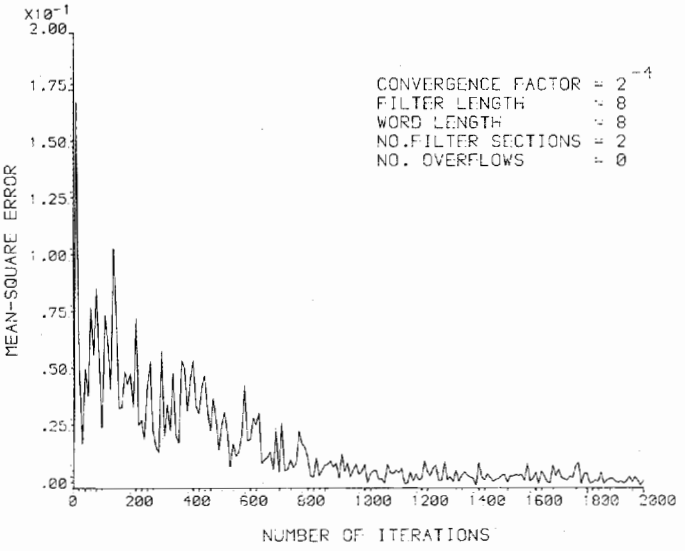
$$\sum b_n f = h_n 2^{N-1}$$

and the true value of h_n can be obtained by right shifting this sum by $N-1$ bits. Repetition of this process for all n leads to the solution for the vector $\boldsymbol{H}$, and thence to the transfer function of the filter.

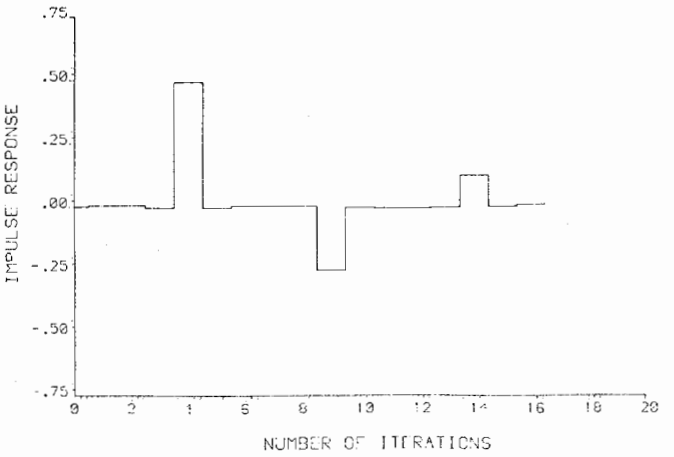
The next area of investigation was the possibility of cascading short filter sections to achieve large time-bandwidth products. As RAM size grows exponentially with filter length, it is important to establish cascading techniques. Fig. 7 shows a cascaded hardware configuration. The error signal from the first stage is used as the training signal for the second. The model was tested under system modelling conditions, again using rectangular noise, but in this case the training signal was passed through a 16-point transversal filter with nonzero elements at taps 4, 9, and 14. As in the single section case, Fig. 8(b) shows the impulse response to be identical to that of the modeled system, although Fig. 8(a) indicates a more significant residual error than the single section MSE plot.

Finally, channel equilization using decision feedback was attempted. DFB may be used in cases where the desired signal has a finite number of possible states, as in the case of a PRBS (pseudorandom binary sequence). The filter uses no external training signal, taking instead the signal state nearest to its own output as the desired signal. However, to ensure convergence, the filter must have a preset impulse response from which to adapt to the channel inverse.

In this simulation, the signal input was a PRBS passed through a distorting mechanism consisting of a central peak of unity, and two sidelobes of value 0.336. This channel was proposed by Satorius [12] as a rigorous test for modem equalizers, due to the high eigenvalue ratio (over 20) of the channel autocorrelation matrix. A filter model was created wherein the first section consisted of taps 5 to 12, and the second section taps 1 to 4 and 13 to 16. This was to ensure that the maximum amount of filter power was handled by the first



(a)



(b)

Fig. 8. Computer simulation showing the convergence characteristics for a cascade of two 8-point distributed arithmetic adaptive filters. (a) MSE convergence for a system modeling situation with random inputs. (b) Impulse response of the filter after convergence.

section, leaving the second to "tidy up the edges" of the system response. The second section had no preset response, and the first section had a single peak of unity preset at tap 9, i.e., the center of the overall filter. Presetting an impulse response in a distributed arithmetic filter as not as simple as in

the case of a classical transversal filter, due to the implicit nature of the filter coefficients. However, in the case of a single spike a tap n, address line n is used as the ADD/SUBTRACT line, and the entire RAM is loaded with the amplitude of the preset spike. This may be verified by the equation solution process used in the transfer function discussion above.

Results of the simulation show a fairly low residual error after 2000 iterations [Fig. 9(a)]. The inverse impulse response of the Satorius channel is displayed in Fig. 9(b), and Fig. 9(c) reveals the output "eye-opening" as the filter converges.

V. Prototype System Design and Experimental Results

The prototype system constructed in this study was designed with standard TTL integrated circuits using the adaptive algorithm quoted in equation (14) with the following modification:

$$\boldsymbol{F}' = \boldsymbol{F} + 2\mu \boldsymbol{N}\boldsymbol{X} \operatorname{sgn}(e) \tag{15}$$

where

$$\operatorname{sgn}(e) = \begin{cases} 1 & \text{when } e > 0 \\ -1 & \text{when } e < 0, \end{cases}$$

i.e., the linear error is replaced by the sign of the error.

A full block diagram of the prototype system is shown in Fig. 10. The input is quantised as an 8-bit digital word which is injected as a serial bit stream into a series of shift registers. Every eighth output from these shift registers forms an address line to a set of random access memories. Since the prototype filter was designed to have eight filter points only eight address lines are used, hence the memories were 256 X 4 bit RAM's. Six such memory devices were used giving a total partial product word length of 24 bits.

The operation of the filter can be split into two cycles. In the first cycle the RAM is sequentially accessed by addresses formed from all eight bit planes. The RAM contents are accumulated using a shift-add circuit to form the filter output after the eighth access. This output is then subtracted, digitally, from the $d(t)$ input to form the error. The second operation cycle is then entered, involving the update of the partial products stored in RAM. The same address locations are accessed in this cycle as in the formation of the filter output by recirculating the contents of the input shift registers. Each RAM location is incremented or decremented by a scaled constant dependent on the order of the bit plane and the sign of the error.

The final system used a total of 60 integrated circuits with a designed sampling rate of 16 kHz and a power consumption of 7.5 W. In a system designed to take full advantage of the speed given by the architecture the sampling rate for a filter with 8-bit input signal quantisation would rise to 2.5 MHz.

The result shown in Fig. 11 demonstrates the convergence properties of the prototype filter. Here, the input signal was a sinusoid with period equal to eight samples and the filter was initially trained to produce zero signal at its output. The training signal was then switched to be a copy of the input signal sinusoid. The trace in Fig. 11 shows the reduction in

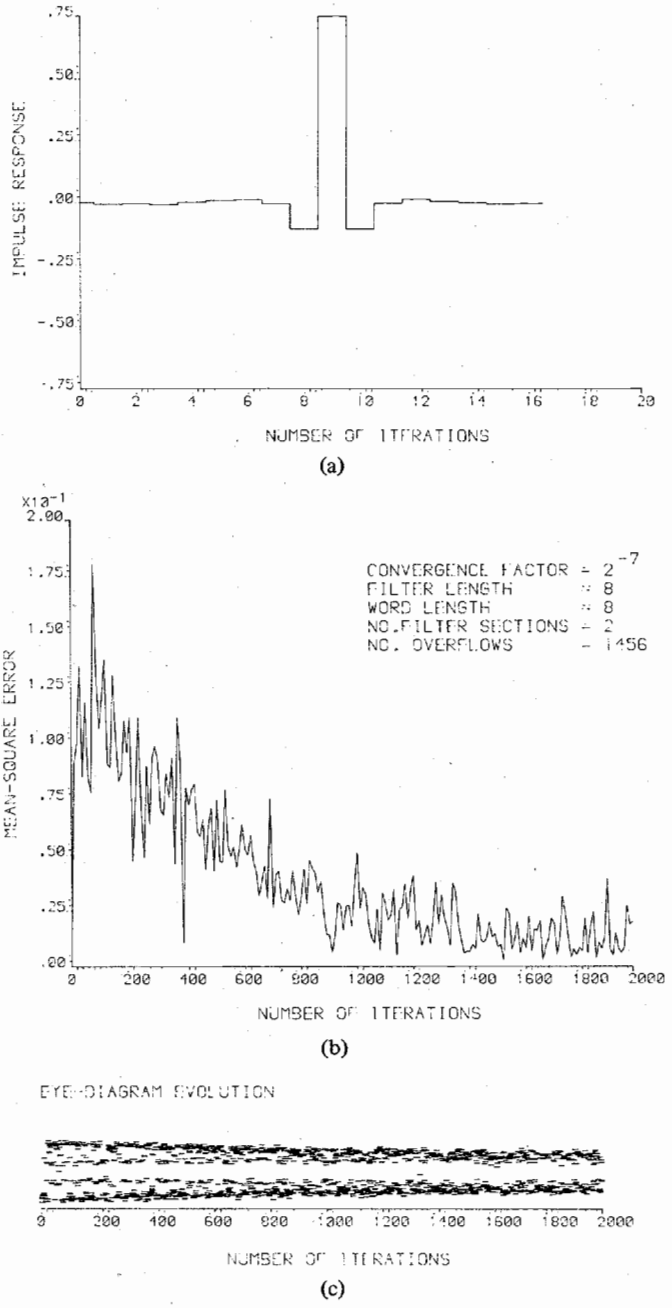


Fig. 9. Simulation result showing the use of a distributed arithmetic adaptive filter in channel equalisation. (a) Converged impulse response of the filter using a cascade of two 8-point filters. (b) MSE plotted against the number of algorithm iterations. (c) Eye diagram evolution plotted against the number of algorithm iterations.

the error output towards zero as the filter converges. The convergence factor in this case was 2^{-20} with the horizontal scale in Fig. 9 being 2 s/div. This yields an overall convergence time of approximately 192 000 samples (12 s at a sampling rate of 16 kHz). This corresponds well to extrapolated predictions from the computer simulations given in Fig. 4.

An important aspect of the performance of an adaptive filter is its ability to perform the function of cancellation. The result shown in Fig. 12 shows the use of the experimental system as a canceller. The $d(t)$ input is a composite signal made

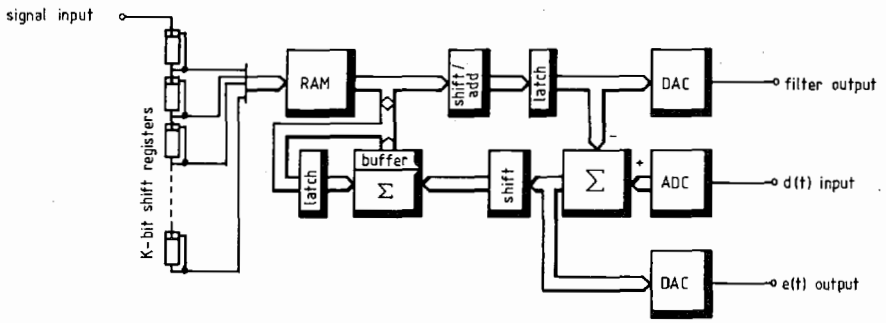


Fig. 10. Block diagram of the experimental distributed arithmetic adaptive filter prototype.

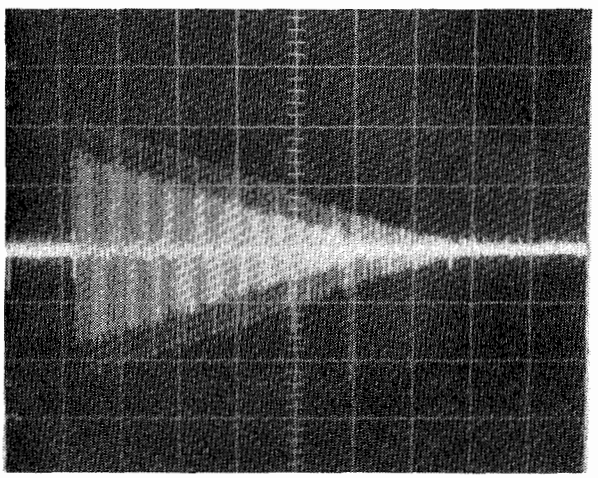


Fig. 11. Experimental result showing the convergence of the filter error signal to a sinusoidal input (vertical scale—0.5 V/div.; horizontal scale—2 s/div.; sampling rate—16 kHz; convergence factor—2^{-20}).

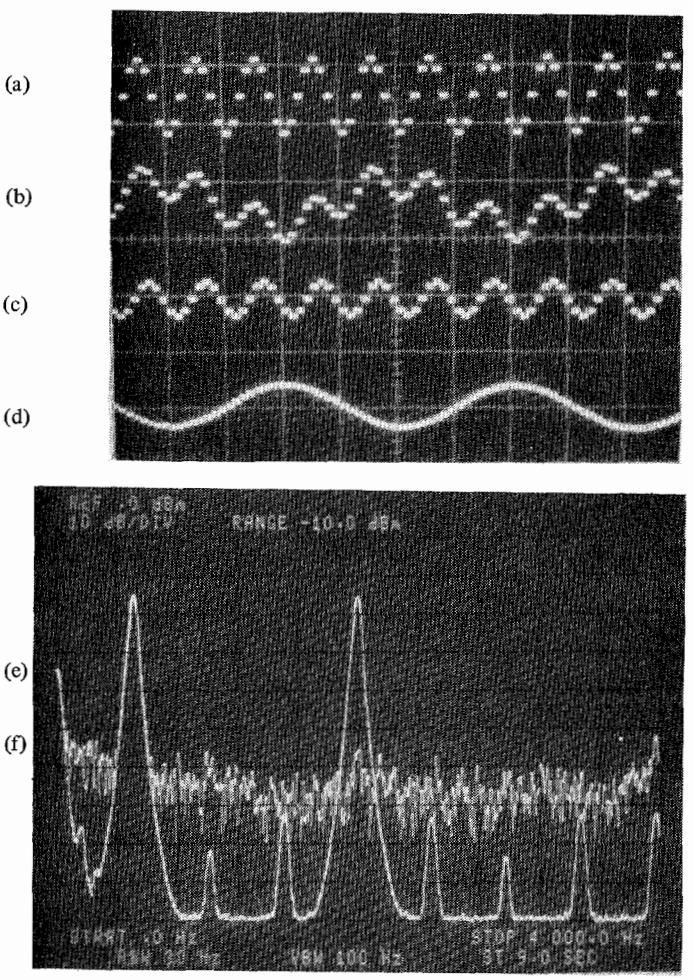


Fig. 12. Experimental result showing the use of the distributed arithmetic filter as a canceller. (a) $s(t)$ input (1 V/div.). (b) $d(t)$ input (1 V/div.). (c) Filter output (1 V/div.). (d) Error or cancellation output (1 V/div.). (e) Spectrum of d(t) input signal. (f) Spectrum of cancellation output (10 dB/div., 400 Hz/div.).

up of a signal at 400 Hz added to an interfering signal, of the same amplitude, at 2 kHz. It is desired to reproduce the 400 kHz signal by coherently subtracting the 2 kHz signal from the composite input. This is done by applying a 2 kHz sinusoid to the $s(t)$ input [Fig. 12(a)]. After convergence the filter output is a 2 kHz sinusoid [Fig. 12(c)] with the correct phase and amplitude to subtract coherently from $d(t)$ to yield the required signal on the $e(t)$ output [Fig. 12(d)]. Fig. 12(e) and (f) show frequency spectra for the $d(t)$ input and the $e(t)$ output, respectively, showing the 2 kHz interfering signal to have been cancelled by 40 dB on the $e(t)$ output. It can be seen from this spectrum that the remaining 2 kHz component on the $e(t)$ output is at the quantization noise level and further cancellation is therefore impossible in this case.

The final result shown in Fig. 13 demonstrates the use of the filter in a system modeling experiment. Here the input signal $s(t)$ is pseudonoise sequence with a length of $2^{24}-1$ samples. The eye pattern for this sequence is shown in Fig. 13(a). This signal is also passed through a transversal filter with an impulse response shown in Fig. 13(d). The output of this filter is an eight-level signal, with an eye pattern shown in Fig. 13(b), which is applied to the $d(t)$ input of the adaptive filter. After convergence the eye pattern of the signal on the filter output is a good replica of the $d(t)$ input [cf. Fig. 13(c)]. When the filter input signal is then changed to an impulse the resulting filter output is shown in Fig. 13(e) which is a good replica of the impulse response of the modelled system. This result corresponds to the computer simulation results given in Fig. 6.

VI. Conclusions

A new technique for the realisation of digital adaptive filters has been presented using an adaptive algorithm based on the Widrow LMS algorithm. The filter uses table look-up tech-

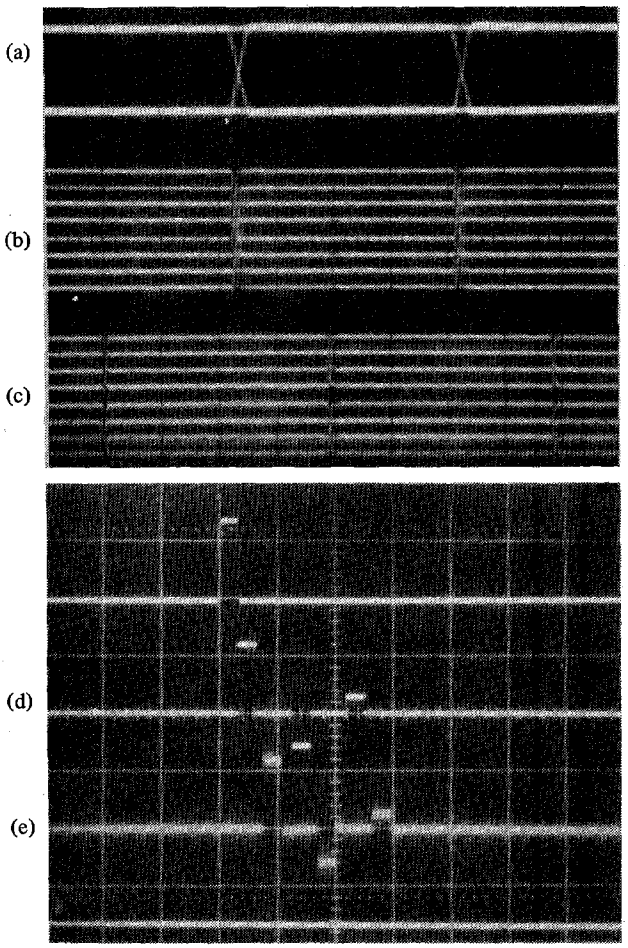


Fig. 13. Result showing the use of the prototype filter in a system modeling experiment. (a) Eye diagram of input signal (pseudonoise sequence of length 2^{24}). (b) Eye diagram of output from system to be modeled. (c) Eye diagram of adaptive filter output after convergence. (d) Impulse response of system to be modeled. (e) Converged impulse response of the experimental adaptive filter.

niques to overcome the need for hardware multipliers and the adaptive algorithm has been so designed to also avoid multiplication.

Although no theoretical proof of the algorithm convergence has been presented a great deal of empirical evidence illustrating the stability of the technique has been presented using both computer simulations and a real-time hardware prototype. This type of distributed arithmetic based adaptive filter offers great advantages in terms of hardware simplicity and may be implemented either at high bandwidth using random logic or, more simply, using a microprocessor based design for low sampling rate applications.

Since the distributed arithmetic filter aproach was initially proposed for use in recursive systems it is ideally suited to adaptive recursive filter implementations. Future efforts at Edinburgh will be focussed on the development of suitable recursive algorithms for this filter type. In addition, the nonlinear characteristics of the system are under investigation.

References

[1] B. Widrow *et al.*, "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, pp. 1692–1716, Dec. 1975.

[2] C. F. N. Cowan *et al.*, "An evaluation of analogue and digital adaptive filter realisations," in *IEEE Int. Sem. Case Studies in Advanced Signal Processing*, Peebles, Scotland, 1979, pp. 178–183.

[3] D. L. Duttweiler and Y. S. Chen, "A single-chip VLSI echo canceller," *Bell Syst. Tech. J.*, vol. 59, pp. 149–160, 1980.

[4] B. K. Ahuya, M. A. Copeland, and C. H. Chan, "A sampled analog MOS LSI adaptive filter," *IEEE J. Solid-State Circuits*, vol. SC-14, pp. 148–154, 1979.

[5] C. F. N. Cowan, J. W. Arthur, J. Mavor, and P. B. Denyer, "CCD based adaptive filters: Realization and analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 220–229, Apr. 1981.

[6] P. B. Denyer *et al.*, "Monolithic adaptive equaliser," in *Proc. ESSCIRC*, Sept. 1982.

[7] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 456–462, 1974.

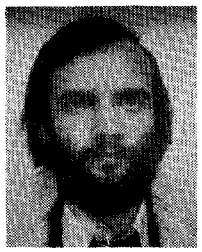
[8] B. O. Justnes, "A transmission module for the digital subscriber loop," in *IEEE Proc. Commun. '80*, Apr. 1980, pp. 73–76.

[9] N. Holte and S. Stueflotten, "A new digital echo canceler for two-wire subscriber lines," *IEEE Trans. Commun.*, vol. COM-29, pp. 1573–1581, Nov. 1981.

[10] C. F. N. Cowan and J. Mavor, "A new digital adaptive filter implementation using distributed arithmetic techniques," in *Proc. IEEE Pt. F, CRSP*, vol. 128, Aug. 1981, pp. 225–230.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.

[12] E. H. Satorius and S. T. Alexander, "Channel equalization using adaptive lattice algorithms," *IEEE Trans. Commun.*, vol. COM-27, pp. 899–905, June 1979.

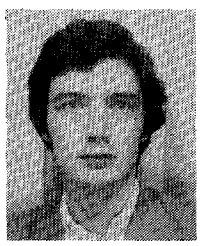


Colin F. N. Cowan was born in Newry, Ireland, on December 6, 1955. He received the B.Sc. and Ph.D. degrees in electrical and electronic engineering from the University of Edinburgh, Edinburgh, Scotland, in 1977 and 1980, respectively.

In 1980 he was appointed as a Lecturer in the Department of Electrical Engineering, University of Edinburgh. His current interests include the implementation and application of adaptive filters and image processing.

Stewart G. Smith received the B.Sc. degree from the University of Glasgow, Glasgow, Scotland, in 1974.

After working in industry for two years, he became a consultant in audio engineering and electronic music systems. In 1981 he joined the Department of Electrical Engineering, University of Edinburgh, as a Research Associate in digital signal processing. His current research interests are adaptive systems and VLSI implementations of signal processing functions.



James H. Elliott was born in Dunfermline, Scotland, in 1960. He received the B.Sc. degree in electrical and electronic engineering from the University of Edinburgh, Edinburgh, Scotland in 1982.

In the same year he joined the Telecommunications division of Hewlett-Packard Ltd., West Lothian, Scotland, as a Development Engineer in their transmission group. His current interests include microwave digital radio and adaptive filtering.