LSI module for the implementation of digital filters

J.G.M. Goncalves, M.Sc., E.E.(I.S.T.), and R.M. Lea, B.Sc., M.Sc., C.Eng., A.R.C.S., M.I.E.E.

Indexing terms: *Signal processing, Digital filters*

Abstract: Real-time signal processing applications are stimulating LSI chip designs for cost-effective digital filter implementations. One such device, the microprogrammable arithmetic element (MAE), is considered and its potential as a building block for different filter implementations is discussed. Direct-form and cascade MAE implementations of FIR and IIR filters are described and compared in terms of bandwidth, cost and chip efficiency.

1 Introduction

Digital filters offer an attractive alternative to analogue filters in real-time signal processing applications for the following reasons:

(*a*) sharper cut-off filters are more easily obtained

(*b*) component tolerance becomes less important

(*c*) higher reliability and better noise immunity are achieved

(*d*) time-dependent and time-shared filters are easily implemented

(*e*) hardware modularity can be achieved.

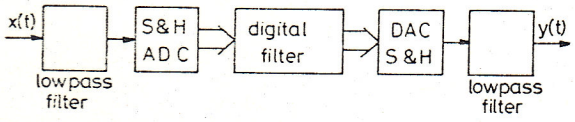

Fig. 1 *Digital filter as part of a real-time system: typical chain of blocks*

A digital filter as illustrated in Fig. 1 can be described by a summation of products as represented in eqn. 1:

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{i=1}^{M} b_i y(n-i) \qquad (1)$$

Thus a direct implementation of a digital filter would comprise a cascade of 'delay' and 'multiplier' elements feeding 'summation units' as indicated in Fig. 2. The inherent replication of the delay-multiplier pair and its potential for other signal processing applications are attracting the attention of LSI/VLSI chip designers. Indeed, two architectural trends are already evident:

(i) *single-chip processing elements;* for example word-oriented microprocessors such as the INTEL 2920 and AMI S2811, and fast multiplier-accumulator chips such as the TRW TDC1010 and AMD AM25LS2516, leading to hardware implementations comprising a small assembly of sophisticated, expensive and fairly large DIL packs.

(ii) *multiprocessing elements;* for example bit-slice microprocessors such as the AMD 2900 and Motorola 10800 families, and fast multiplier-accumulator chips such as the Plessey MAE (microprogrammable arithmetic element) leading to regular arrays of simple, cheap and small DIL packs.

This paper reports a feasibility investigation concerning the implementation of digital filters with the Plessey MAE.

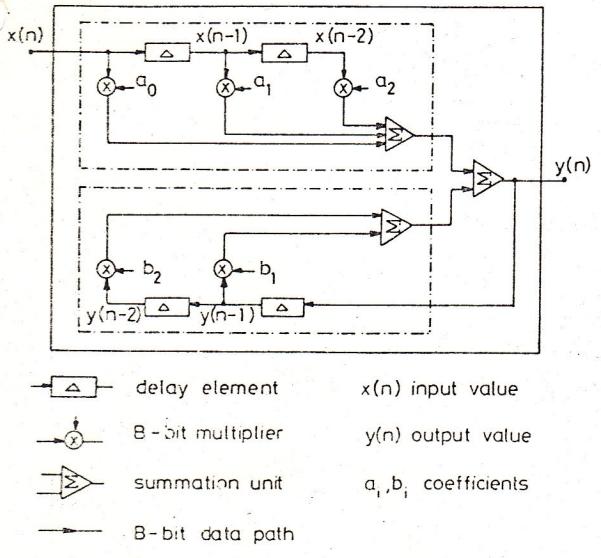

Fig. 2 *Schematic representation of eqn. 1 (N = 3, M = 2)*

Paper 1631F, first received 19th January and in revised form 31st July 1981

Mr. Goncalves is with the Department of Medical Biophysics, University of Manchester, Oxford Road, Manchester M13 9PL, England, and Mr. Lea is with the Electrical Engineering & Electronics Department, Brunel University, Uxbridge UB8 3PH, England

2 Digital filtering

2.1 Classification

Digital filters can be classified, according to the value of M in eqn. 1, as follows:

(*a*) $M = 0$, nonrecursive or finite impulse response (FIR) filter

(*b*) $M > 0$, recursive or infinite impulse response (IIR) filter.

The *order* of a digital filter is given by the maximum number (namely N or M in eqn. 1) of delay elements in either its forward (i.e. N) or feedback paths (i.e. M).

The *form* of a digital filter refers to the manner in which it represents eqn. 1.

Hence Fig. 2 shows a *direct form* representation of a *2nd-order IIR* filter, decomposed into two *2nd-order FIR* filters.

2.2 Implementation

System functions, $H(z)$, are obtained by applying the Z-transform to eqn. 1.

$$\text{FIR:} \quad H(z) = \sum_{i=0}^{N-1} a_i z^{-1} \qquad (2)$$

$$\text{IIR:} \quad H(z) = \frac{\sum_{i=0}^{N-1} a_i z^{-1}}{1 - \sum_{i=1}^{M} b_i z^{-1}} \qquad (3)$$

Some manipulations on $H(z)$ suggest other implementations, which will be separately analysed for each type of filter.

2.2.1 FIR filters: $H(z)$, being a polynominal of z^{-1}, can be rewritten in terms of its roots, or better, in terms of 2nd-order

polynominals grouping some eventual complex roots [1]:

$$H(z) = \prod_{k=1}^{m} (\alpha_{0k} + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}) \quad (4)$$

This representation suggests that the filter can be built as a cascade of 2nd-order FIR filters as shown in Figs. 3A and 3C.

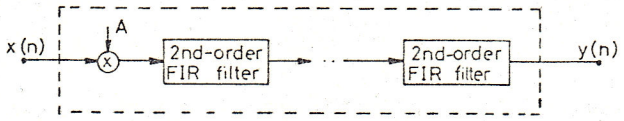

Fig. 3A *Cascaded form for an FIR filter (each block is detailed in Fig. 3C)*

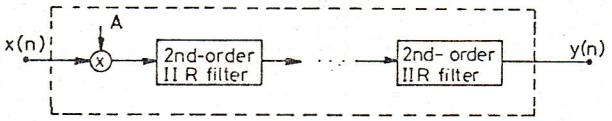

Fig. 3B *Cascaded form for an IIR filter (each block is detailed in Fig. 2)*

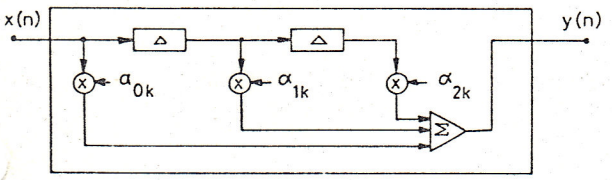

Fig. 3C *Representation of a section for the cascaded form FIR filter*

2.2.2 IIR filters: $H(z)$, being a ratio of polynomials, can be represented as

$$H(z) = A \prod_{i=1}^{m} \frac{1 + \alpha_{1i} z^{-1} + \alpha_{2i} z^{-1}}{1 - \beta_{1i} z^{-1} - \beta_{2i} z^{-2}} \quad (5)$$

Thus an IIR filter can also be implemented as a cascade of 2nd-order IIR filters as shown in Fig. 3B.

Alternatively, $H(z)$ can be decomposed as a sum of partial fractions in eqn. 6:

$$H(z) = \gamma_0 + \sum_{i=1}^{m} \frac{\gamma_{1i} z^{-1} + \gamma_{0i}}{1 - \beta_{1i} z^{-1} - \beta_{2i} z^{-2}} \quad (6)$$

Thus an IIR filter can also be represented as a parallel combination of 2nd-order IIR filters as shown in Fig. 4.

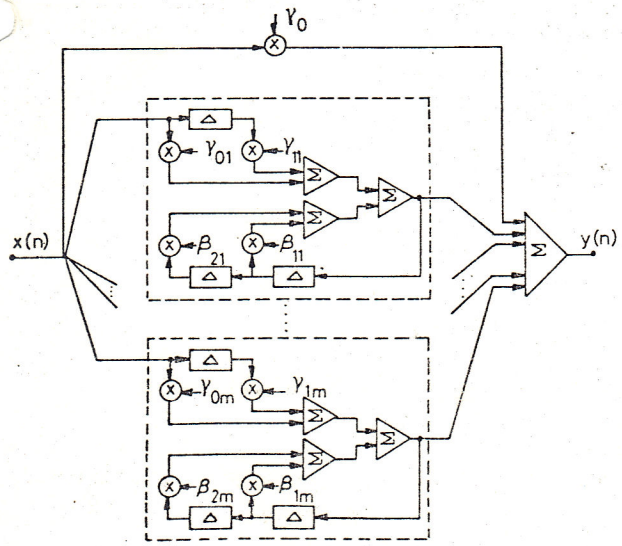

Fig. 4 *Representation of the parallel form for an IIR filter (each block is a 2nd-order IIR filter as shown in Fig. 2, with N = 2, M = 2)*

2.2.3 Practical considerations: Some problems of accuracy arise as a consequence of the use of a finite word length. For example, the binary representation of filter coefficients leads to quantisation errors which affect the filter's frequency response and ultimately its stability. However, for the same bit precision, cascaded and parallel filter implementations are less sensitive to such errors than direct-form implementations. Each multiplication introduces a truncation or rounding error, the latter having the advantage that it corresponds to 'white noise'. Fixed-point arithmetic is preferred for reasons of cost effectiveness and arithmetic overflow is usually avoided by restricting the digital filter to a low gain or by employing special hardware.

3 Microprogrammable arithmetic element

3.1 Description

The microprogrammable arithmetic element (MAE) is a digital signal processing device, designed and fabricated with ECL technology by Plessey [2, 3]. As a single-chip 4-bit twos'-complement arithmetic unit, it includes a built-in multiplier, an adder/subtractor and four registers (each one connected by a two-way multiplexer as shown in Fig. 5). The multiplier uses a serial-parallel scheme and is based on Booth's 2-bits-at-a-time algorithm.

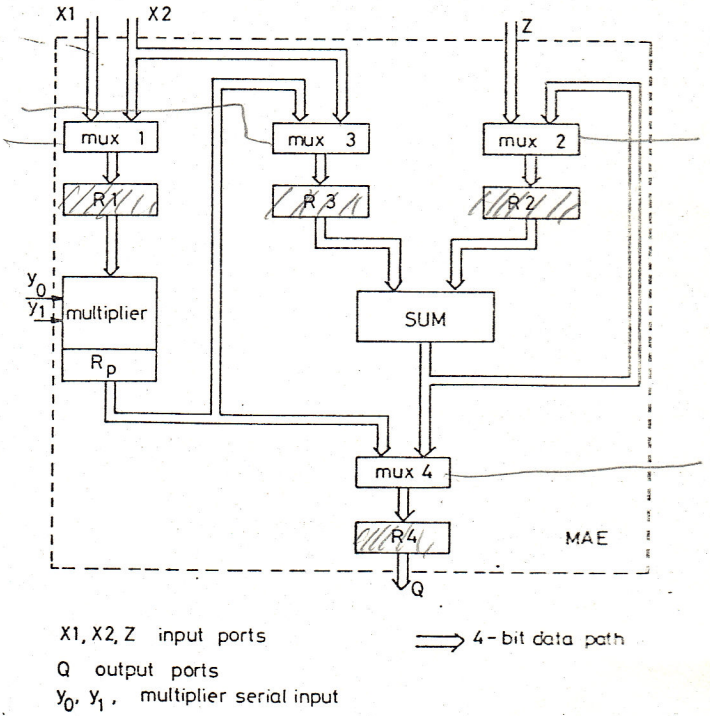

Fig. 5 *Internal architecture of the MAE*

The MAE is controlled by two independent sets of instructions, as shown in Table 1, which can be separately clocked, thereby allowing the simultaneous execution of two

Table 1: MAE instruction sets

Instruction set I		Instruction set II	
Mnemonic	operation	Mnemonic	operation
*LX1R1	R1 ← X1	LSR2	R2 ← SUM
CLKMUL	multiply	ADD	addition
ROUND	round option	NADD	subtraction (R3−R2)
LPR 3	R3 ← Rp	LSR4	R4 ← SUM
		*LX2R1	R1 ← X2
		LZR2	R2 ← Z
		SUB	subtraction (R2−R3)
		LPR4	R4 ← Rp

*When LX1R1 and LX2R1 are simultaneously clocked, LX2R1 means R3 ← X2

operations (e.g. a multiplication and an addition). All the operations except multiplication are executed with a single clock pulse. The serial-parallel multiplier requires two CLKMUL instructions (i.e. two clock pulses) to execute a 4×4 multiplication. A rounding option can also be programmed.

3.2 Arithmetic element unit

For bit precisions B greater than 4, a cascade of MAEs is required; for example, if $B = 8$, then two MAEs are used, as shown in Fig. 6. All the MAEs are clocked with the same instructions for sets I and II. The cascade of MAEs, required to process a B-bit word, will be called an arithmetic element unit (AEU).

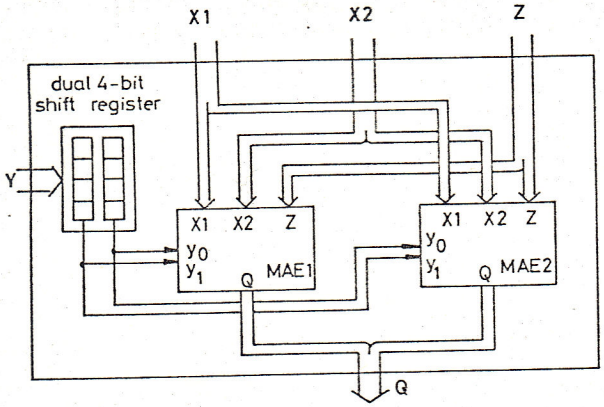

Fig. 6A *Cascade of two MAEs to process an 8-bit word*

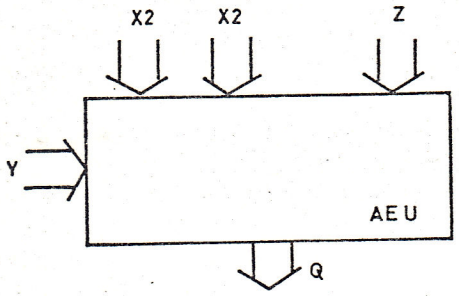

Fig. 6B *Block representation of the cascade, i.e. the arithmetic element unit (AEU)*

The maximum clock rate for the AEU is dependent on the value of B, owing to carry propagation in the cascade, and its period T_p, in nanoseconds, is given by

$$T_p = 20 + 1.25B \tag{7}$$

For example, a 16-bit word can be processed with an AEU comprising 4 MAEs and clocked at a frequency $f = 25$ MHz.

4 Digital filter implementations using the MAE

A digital filter can be specified in terms of its:

(i) sampling frequency
(ii) classification (FIR or IIR)
(iii) coefficients
(iv) bit precision
(v) structural organisation (direct-form, cascaded, parallel, . . .).

The number of coefficients determines the number of operations to be executed, which together with the bit precision and the sampling frequency defines the speed requirement of the implementation. Several examples are studied, and for each the minimum time between two consecutive samples is specified to indicate their application range. Only the arithmetic part of each implementation is considered; the control circuitry can be found elsewhere [4].

4.1 Multiplication cycle

This cycle is fundamental to MAE operation and it includes the load of the multiplicand to R1 (LX1R1), the multiplication (CLKMUL) and product storage for further processing (LPR3). As a consequence of the multiplication algorithm (namely two-bits at a time), $B/2$ consecutive CLKMUL instructions are needed for the execution of a $B \times B$ bit multiplication. The cycle time therefore uses $(B/2 + 2)$ clock pulses and is given by

$$T_c = (B/2 + 2)T_p \tag{8}$$

Table 2 indicates the sequence of operations in a typical multiplication cycle. Note that the provision of independent instruction sets allows overlapped operations to improve the overall execution speed.

Table 2: Typical multiplication cycle instructions for an 8-bit word

Comments I	Instruction I	Instruction II	Comments II
	.		
	.		
	.		
	CLKMUL		
Store product	LPR3		
Load multiplicand	LX1R1	ADD	accumulate to R2
	CLKMUL	LSR4	output result
Execute	CLKMUL		
Multiplication	CLKMUL		
	CLKMUL		
Store product	LPR3	LZR2	initialise R2
Load multiplicand	LX1R1	ADD	accumulate to R2
	CLKMUL	LSR2	store Acc
	.		
	.		
	.		

The accumulate operation is executed in parallel with the multiplication

4.2 FIR filters

4.2.1 Direct form: This implementation uses a single AEU, a ROM to store the N coefficients and a RAM to store the past N values of the input signal, as shown in Fig. 7.

When a new value $x(n)$ is ready, it is input to the AEU (as a multiplicand) and simultaneously stored in the data memory. The multiplier a_0 derived from the coefficient memory is also transferred to the AEU, and a multiplication is initiated. In parallel with the execution of this cycle, the accumulator

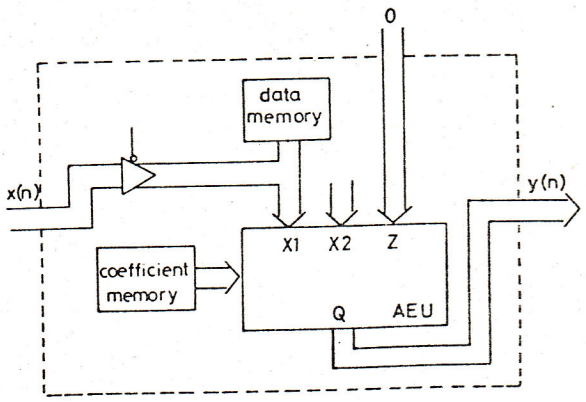

Fig. 7A *Implementation of a direct-form FIR filter using one AEU*

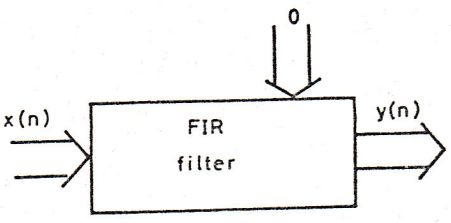

Fig. 7B *Block representation of Fig. 7A*

register (R2) is cleared using port Z. On completion of the cycle, the product is accumulated in R2. The next cycle has as operands the value $x(n-1)$ read from the data memory and the coefficient a_1 read from the coefficient memory. The accumulator is output after N multiplication cycles, thereby generating $y(n)$, and a new input value is sensed.

Since each input sample requires the execution of N multiplication cycles and the minimum sampling frequency is twice the maximum signal frequency, the maximum bandwidth of a direct-form FIR lowpass digital filter is given by

$$W_{FD} = 1/2NT_c \tag{9}$$

and, from eqn. 7 and 8,

$$W_{FD} = \frac{800}{N(B+4)(B+16)} \tag{10}$$

Hence, for a 5th-order (namely $N=5$) filter with 16-bit precision (namely $B=16$), the bandwidth W_{FD} is 0.25 MHz.

4.2.2 Cascaded form: As shown in Fig. 3, an FIR filter can be decomposed into a cascade of 2nd-order filters. Such a cascade can be implemented with AEUs, as shown in Fig. 8.

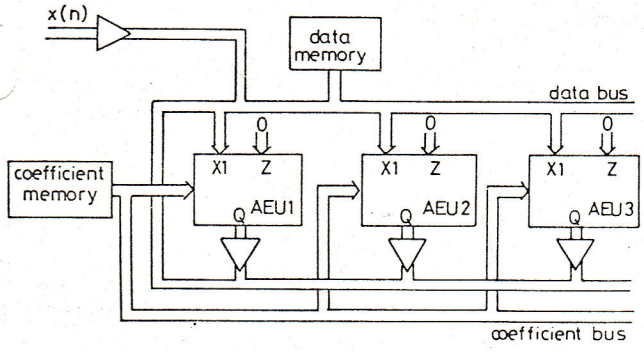

Fig. 8 *Implementation of a cascaded-form FIR filter*

Each input sample $x(n)$ is transferred, via the data bus, to AEU_1 and simultaneously loaded in the data memory. A multiplication cycle is initiated in AEU_1 to multiply $x(n)$ by α_{01} and the accumulator (R2) in AEU_1 is cleared by loading '0' through port Z. After two more AEU_1 cycles, corresponding to the coefficients α_{11} and α_{21}, its output is transferred via the data bus to AEU_2 and loaded in the data memory. Processing continues with AEU_2 and AEU_1, simultaneously performing three further multiplication cycles for the coefficients α_{02}, α_{12}, α_{22} and α_{01}, α_{11}, α_{21}, respectively Thus, when the pipeline is 'full', all the AEUs are executing multiplication cycles in parallel, and after every 3 cycles all AEU outputs are shifted along the cascade. The output of the last AEU is the filter output $y(n)$.

The two memory banks (namely data memory and coefficient memory) can be shared by different AEUs, since they are accessed only at the beginning of a multiplication cycle and the rest of the cycle they remain idle. If the AEU cycles are out of phase (i.e. they start at different time slots), the memories can be shared, provided they are fast enough to access data in a single clock cycle. This memory sharing must be accompanied by the appropriate data bus control. Clearly, from eqn. 8, the limit to the number of AEUs sharing the same memory is equal to $(B/2)+2$, the number of clock pulses in each multiplication cycle.

Once the pipeline is 'full', a new output sample will be available after every three multiplication cycles, and, since the minimum sampling frequency is twice the maximum signal frequency, the maximum bandwidth of a cascaded-form FIR lowpass digital filter is given by

$$W_{FC} = 1/6T_c \tag{11}$$

and, from eqns. 7 and 8,

$$W_{FC} = \frac{800}{3(B+4)(B+16)}\ \text{MHz} \tag{12}$$

Hence, for 16-bit precision, the bandwidth W_{FC} is 0.42 MHz, independent of the order of the filter.

4.3 IIR filters

4.3.1 Direct form: Eqn. 1 and Fig. 2 show that an IIR filter can be decomposed into two FIR filters, one implementing the forward path and the other the feedback path, with the outputs being added to generate the filter's output $y(n)$. An immediate implementation is shown in Fig. 9, where each block is an FIR filter as described in Section 4.2.1. The addition of the two outputs is done by initialising the accumulator of the second block (AEU) with the output of the first block.

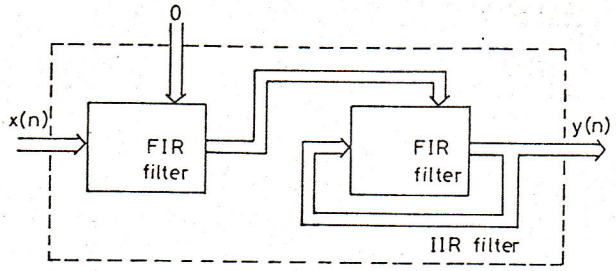

Fig. 9 *Block representation of the implementation of a direct-form IIR filter (each block is an FIR filter as detailed in Fig. 7A)*

Since P multiplication cycles are executed between two consecutive samples, where P is the filter's order [i.e. $P=\max(N,M)$] the maximum bandwidth for this configuration is

$$W_{ID} = 1/2PT_c \tag{13}$$

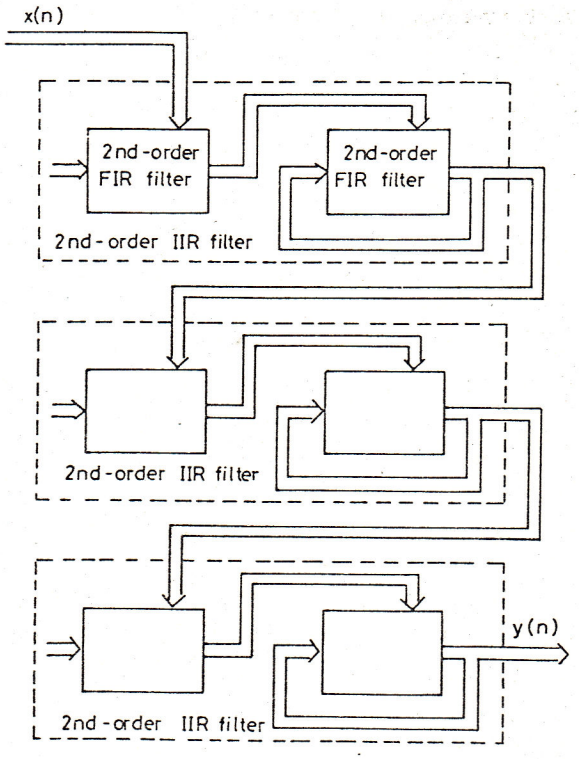

Fig. 10 *Block representation of the implementation of a 6th-order IIR filter in its cascaded form*

and, from eqns. 7 and 8,

$$W_{ID} = \frac{800}{P(B+4)(B+16)} \text{ MHz} \quad (14)$$

Hence, for a 5th-order filter ($P = 5$), with 16-bit precision ($B = 16$), the bandwidth W_{ID} is 0.25 MHz.

4.3.2 Cascaded form: In the cascaded form, the filter is implemented as a cascade of 2nd-order IIR filters (cascade section). As expressed in eqn. 5, each section requires only four multiplications: two for the FIR filter in the forward path and two for the FIR filter in the feedback path.

This implementation uses two AEUs per cascade section. Fig. 10 shows the filter as a cascade of IIR filter blocks, which are described in Section 4.3.1. This block representation assumes local memory in each block. As in Section 4.2.2, it is possible to share the memory banks using a bus-based architecture [4].

For each block in Fig. 10, the cascaded value is loaded directly into the next AEU's accumulator instead of passing through the multiplier and multiplied by one, to save one multiplication.

Since two multiplication cycles must be executed for each new sample, the achieved bandwidth is given by

$$W_{IC} = 1/4\,T_c \quad (15)$$

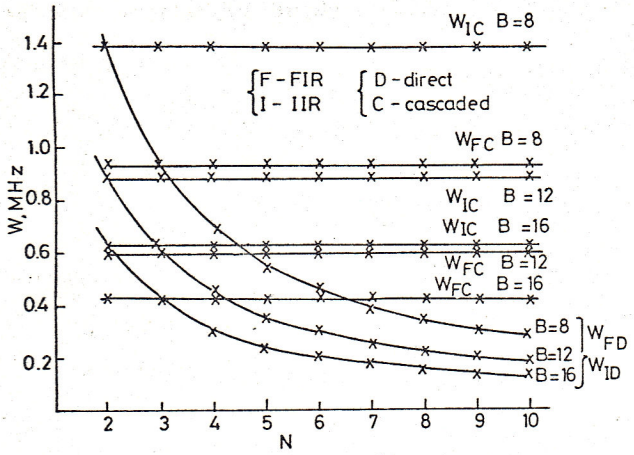

Fig. 11 *Dependence of the bandwidth W on the filter's order N for several bit precisions B*

F = FIR
I = IIR
D = direct
C = cascaded

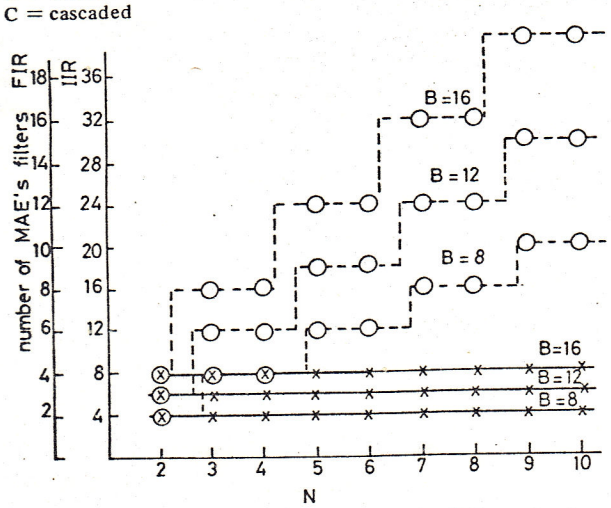

Fig. 12 *Required number of MAEs for the different implementations, depending on the filter's order N and on the bit precision B*

o cascaded form
x direct form

and, from eqns. 7 and 8,

$$W_{IC} = \frac{800}{2(B+4)(B+16)} \text{ MHz} \quad (16)$$

Hence, for 16-bit precision, the bandwidth W_{IC} is 0.625 MHz, independent of the filter's order.

4.4 Results

If a 'level of pipelining' is defined (for example, the number of AEUs divided by the total number of coefficients), the chosen implementations can be categorised into two different levels: a low level for the direct form when all the operations for an Nth-order FIR filter are executed by a single AEU, and a high level for the cascaded forms in which each AEU only implements a 2nd-order FIR filter. Other implementations are also possible [4]; for example, a pipelined direct form in which several AEUs execute part of the summation in eqn. 1, or a cascaded form with less AEUs, each one implementing more than one FIR filter. For each implementation, a maximum bandwidth expression can be derived, depending only on the level of pipelining and on the bit precision B.

Fig. 11 illustrates the dependence of the bandwidth on the filter's order, and Fig. 12 shows how the number of MAEs varies with the filter's order. The latter Figure also provides a 'cost estimate' for the arithmetic part. Implementations with a high level of pipelining (namely cascaded form) can provide a constant bandwidth. This is paid for by the cost of the number of AEUs, which is proportional to the filter's order as shown in Fig. 12. In addition, Fig. 11 demonstrates that the low cost associated with the direct forms (namely low level of pipelining) is accompanied by a rapid decrease of the bandwidth with the filter's order. Moreover, implementation with an intermediate level of pipelining will show a low decrease of the bandwidth and also a low increase of the number of AEUs.

Table 3: Figures of merit of the MAE's use for the different implementations

Order	Bit precision			Type	Form
	$B = 8$	$B = 12$	$B = 16$		
$N = 10$	139.1	59.7	31.3	FIR	direct
$N = 10$	92.6	39.7	20.9	FIR	cascaded
$N = M = 10$	69.4	29.8	15.6	IIR	direct
$N = M = 10$	69.4	29.8	15.6	IIR	cascaded

Another way of assessing the different implementations can be gained by defining a figure of merit for the MAE (AEU) as the bandwidth kHz divided by the number of MAEs (AEUs). Different figures can only be compared if they correspond to equivalent implementations of the same filter. Table 3 displays these figures of merit for the chosen implementations. Two consequences arise from this Table. First, the cascaded FIR filter, although providing a larger bandwidth, uses the MAEs less efficiently than the direct-form FIR filter. This is due to the fact that each cascade section has three multiplications instead of two. However, a closer look at eqn. 4 shows that it is possible to reduce the number of multiplications per cascade section to two. Nevertheless, this alternative is avoided, for it tends to increase the gain of each section and therefore increase the risk of arithmetic overflow. Secondly, for the IIR filters, there is a linear relationship between the number of AEUs and the achieved bandwidths. Although it is only shown for IIR filters, this relationship also extends to some FIR implementations [4].

5 Conclusions

It has been shown that the MAE as a component and the AEU as a logical module provide many alternatives for the implementation of digital filters. Most importantly, these alternatives can match filter specifications, in terms of structure, order and bit precision. In addition, the linear relationship between the number of MAEs and the filter's bandwidth shows that there are no overhead losses due to an increase in the level of pipelining, and it provides a very wide range of applications. By choosing the required level of pipelining, the filter designer can restrict his design to the specified bandwidth, thereby reducing costs. Another important feature is that MAE implementations reflect the inherent modularity of digital filters. For example, new implementations can be built with previously defined blocks. This versatility arises from the following characteristics of the MAE:

(*a*) register-based architecture
(*b*) programmability
(*c*) instruction overlapping possibilities.

Certain limitations of the MAE suggest possible improvements:

(i) Nonavailability of arithmetic overflow protection imposes low gain throughout the filter, thus reducing its dynamic range. A built-in 'out-of-range' correction circuit using saturated arithmetic would require only two cascading lines from MAE to MAE to indicate the need of correction.

(ii) Instruction LX2R1 seems to be unnecessary; close inspection of examples presented elsewhere [3, 4] reveals that the input port X2 has never been used. Since external data cannot be loaded into R3 without simultaneously loading R1 (see Table 1), the adder can only operate on the results from the multiplier. Thus use of the MAE is restricted to pipelined designs. This is confirmed by the unused clock cycles in the accumulating part of the multiplication cycle (see Table 2). However, replacement of LX2R1 with a new instruction LX2R3 (R3 ← X2) would allow the MAE to support parallel designs. External data could then be accumulated simultaneously during multiplication.

(iii) The introduction of a tristate output would facilitate the use of the device in a bus-based architecture.

In summary, the results of this study augur a good future for MAE-like devices (namely block-slice fast multiplier-accumulator chips). Modularity may encourage the introduction of new designs (e.g. chips incorporating a small data memory, or off-the-shelf module boards) which, being independent of word length, would meet the basic requirements of digital filtering. Word-length independence becomes important for the implementation of digital filters using Winograd's algorithm [5], where a sharp reduction in the number of multiplications can be achieved (e.g. K consecutive outputs of an Nth-order FIR filter require only $N+K-1$ multiplications instead of KN) with a corresponding increase in the bit precision of each operation to overcome round-off problems.

6 Acknowledgments

The authors gratefully acknowlege the helpful discussions with S.S. Magar and D.A. Robinson (formerly of Plessey Research Ltd.) and the Instituto Nacional de Investigacao Cientifica, Portugal for its sponsorship of J.G.M. Goncalves at Brunel University.

7 References

1 OPPENHEIM, A., and SCHAFER, R.: 'Digital signal processing' (Prentice-Hall, New Jersey, 1975)
2 MAGAR, S.S.: 'Integrated circuits for fast Fourier transform'. ESSC1RC 77, Sept. 1977, pp. 84–87
3 MAGAR, S.S., and ROBINSON, D.A.: 'Microprogrammable arithmetic element and its applications to digital signal processing', *IEE Proc. F, Commun., Radar & Signal Process.*, 1980, 127, (2), pp. 99–106
4 GONCALVES, J.G.M.: 'Implementation of digital filters using associative parallel processors – a comparative study'. M.Sc. dissertation, Brunel University, 1979
5 WINOGRAD, S.: 'Application of complexity of computations to signal processing'. Proceedings of NATO-ASI on digital image processing and analysis, June 23–July 4, 1980, France, pp. 209–225