

Algorithmic low power FIR cores

A.T. Erdogan, M. Hasan and T. Arslan

Abstract: The authors present a number of novel architectures for the implementation of low power FIR filtering cores. These architectures are directly translated from flexible algorithms which exploit data and coefficient correlation in order to minimise the effective switched capacitance on the multiplier, and data/coefficient buses. Another characteristic of these algorithms is that they can be combined to form more power-efficient algorithms which in turn could be mapped to more effective architectures. The paper describes the FIR filtering architectures, the arithmetic processing cores which characterise individual architectures, and provides results which demonstrate up to 39% reduction in power. In addition, the paper provides an analysis of the arithmetic processing cores and the impact of their constituent components on the overall power saving with different algorithms.

1 Introduction

There is a continuous demand for low power DSP architectures and design methodologies. This demand is because DSPs are crucial for the implementation of high performance portable systems, where longer battery life and minimum heat dissipation are key requirements. FIR cores are important blocks in processing both audio and visual data. For example, devices such as hearing aids require efficient and flexible cores for the implementation of complex algorithms which, in turn, involve recursive and repetitive implementation of various FIR filters on these cores with very constrained power and area demands. Researchers in the literature have developed a number of techniques for low power implementation of digital filters. The authors of [1] utilise a technique that involves using various orders of differences between coefficients, along with stored intermediate results, rather than using the coefficients themselves directly for computing the partial products in the FIR equation. Another approach, used in [2], is to optimise word-lengths of input/output data samples and coefficient values. This involves using a general search-based methodology, which is based on statistical precision analysis and the incorporation of cost/performance/power measures into an objective function through word-length parameterisation. In [3], Mehendale *et al.* present an algorithm for optimising the coefficients of an FIR filter, so as to reduce power consumption in its implementation on a programmable digital signal processor. Power reduction in the algorithm is achieved in two stages. In the first stage, all coefficients are scaled uniformly by a scaling factor, chosen such that the total Hamming distance between successive scaled coefficients is least. The second stage is an iterative procedure in which coefficients are selected iteratively and incremented/decremented slightly in order to achieve a reduction in the total Hamming distance.

The iterative procedure maintains the desired filter characteristics. In [4, 5], the authors have presented techniques that utilise various folded/unfolded filter realisation structures in conjunction with coefficient ordering algorithms for minimising power consumption in FIR filters. Other techniques used by researchers include the use of multirate architectures [6, 7] and dynamic adjustment of filter order [8]. In [9, 10] we have introduced coefficient segmentation and block processing algorithms for low power implementation of FIR filters. These algorithms demonstrated a significant potential for power reduction, since their impact reaches power intensive processing units such as multipliers as well as various buses. However, the evaluation of these algorithms, in terms of power and other performance parameters, has been restricted to the analysis of the multiplier unit alone, without consideration of the other units that make up the overall architecture. This paper presents the full implementation of coefficient segmentation and block processing algorithms, and a new hybrid algorithm that combines both.

2 Generic FIR core

The block diagram of a generic FIR core is shown in Fig. 1. It consists of two memory blocks for storing the coefficients (HROM) and input data (XRAM), two registers for holding the coefficient and input data, namely HREG and XREG, respectively, and the FSM along with the main arithmetic unit (AU). The XRAM is realised in the form of a latch-based circular buffer for reducing its power consumption. The FSM is responsible for applying the appropriate coefficients and input data to AU. The AU architecture, along with that of the FSM, mainly varies from one algorithm to the other. The generic core also has the power management unit (PMU) and the data management unit (DMU). These units are used by external bus interfaces for the application targeted in this work. These could be an ARM-based SoC or a hearing aid platform. Here, the main emphasis is on the basic FIR core, and hence the detailed design of these units is not taken up in this paper.

© IEE, 2003

IEE Proceedings online no. 20030346

doi:10.1049/jp-cds:20030346

Paper first received 26th November 2001 and in revised form 10th October 2002

The authors are with the Department of Electronics & Electrical Engineering, The University of Edinburgh, The King's Buildings, Mayfield Road, Edinburgh EH9 3JL, UK

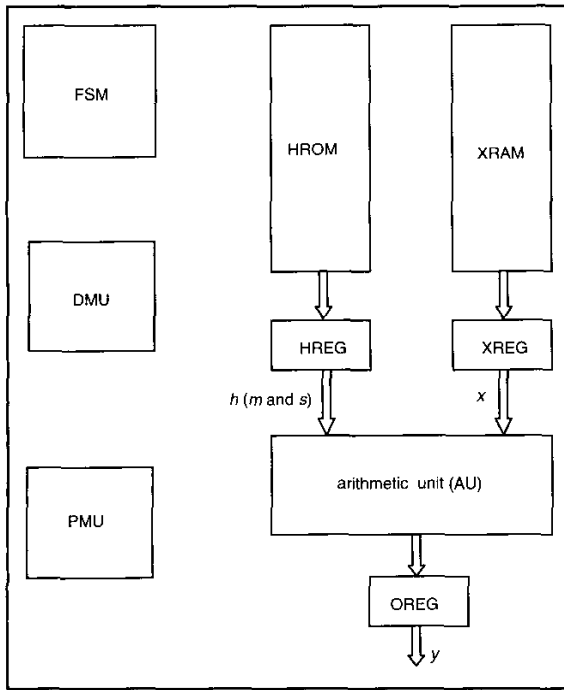


Fig. 1 Low power generic FIR core

3 Arithmetic unit architecture for the conventional filter

In the conventional implementation of an FIR filter, the AU consists of a multiplier (mult), an adder (add), an accumulator (acc) and a clearing logic block (clacc) in the form of a multiplexer as shown in Fig. 2. The clacc carries out the dual operation of feeding the accumulated values back to the adder and also for clearing the accumulator when the valid input to the AU is asserted high. The valid input is asserted high after the generation of a filter output. The mult has three different implementations namely a carry-save array type (csa), a non-Booth-coded Wallace tree type (nbw) and a Booth-coded Wallace tree type (wall). The power evaluation of the filter architectures

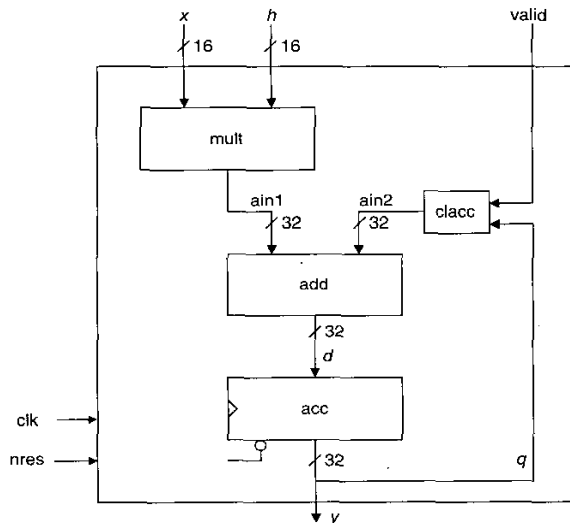


Fig. 2 AU for the conventional filter

was carried out for all three multipliers. The AU is used to multiply a coefficient h_k with an input data sample x_k , adding the previous stored accumulator register value to the product at the same time. Data samples and the filter coefficients are represented as 16-bit two's complement numbers whereas the filter output is of 32-bit. The AUs for different low power algorithms are described as follows.

4 Arithmetic unit architecture for coefficient segmentation algorithm

According to this technique [9], a 16-bit coefficient h_k is segmented into two numbers, namely a 16-bit decomposed coefficient m_k and a 5-bit shift value s_k . The MSB of s_k acts as a sign bit and the remaining four bits are a measure of shift. The input data sample x_k and the number m_k are applied to the multiplier, while a shift operation of x_k will be performed according to the shift value s_k . One more 5-bit register has to be included in the main architecture to store the s_k value. The result of the multiplication and the shifted input data are added to the previous value stored in the accumulator register. The AU for the coefficient segmentation algorithm is shown in Fig. 3. It consists of a multiplier (mult), an adder (add), a logarithmic shifter (shift) implemented using arrays of 2-to-1 multiplexers, a conditional two's complementor (xconv), a multiplexer (mux) to load and clear the shifter and a clearing block (clacc) identical to the one in the conventional FIR filtering block. The MSB of the shift value s_k determines if a negative shift has to be performed and therefore controls the conversion unit xconv. The output of xconv is the two's complement of the data only if the MSB of s_k is one, otherwise the output is equal to the input data. When h_k is zero ($m_k = 0, s_k = 0$) or one ($m_k = 1, s_k = 0$), the shift value will be zero. In these cases, the output of the shifter must be zero as well. To guarantee this behaviour, a multiplexer is needed between the conversion unit and the shifter that applies a zero vector when s_k equals zero. Since three values (multiplier, shifter

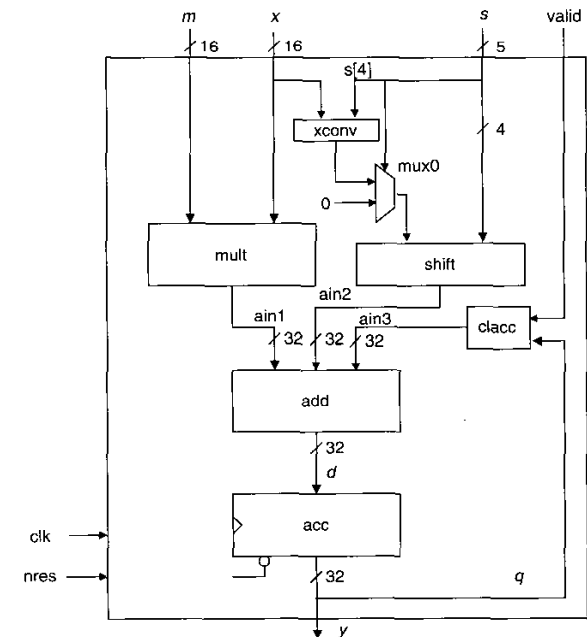


Fig. 3 AU for coefficient segmentation

and accumulator outputs) are to be added, a single multi-input adder carries out this addition.

5 Arithmetic unit architecture for block processing algorithm

In the direct form realisation of the filter, a new data sample x_k and the corresponding coefficient h_k are multiplied at each clock cycle, followed by accumulation in a conventional AU. This leads to high switching activity because both inputs of the multiplier receive new data at every clock cycle. Another source of power consumption in DSPs is the activity on data and address buses. Since each time a new data sample is to be multiplied with a new coefficient, both data and address buses experience high switching activity. According to the block processing technique [10], data samples are processed in blocks. By processing multiple data samples at the same time rather than one at a time, it is possible to reduce the switching activity not only at the multiplier inputs but also on the address and data buses, thereby achieving considerable power saving. As in earlier results [10], blocks of size two yielded best results because both inputs to the multiplier are held constant for two clock cycles rather than only one for the block sizes greater than two. Thus, AU shown in Fig. 4 is designed for a block size of two only. It consists of a multiplier (mult), an adder (add), two accumulators for storing the results for two outputs namely acc0 and acc1, a multiplexer (mux) to select the proper output of the accumulators to be fed back to the adder and also a clearing logic (clacc). The clacc initialises the accumulators in response to the active high valid signal, which goes up at the time of switching from one block of outputs (y_0 and y_1) to the next (y_2 and y_3). The appropriate accumulators are selected by the controls generated by the FSM. Once the block processing for the first two outputs (y_0 and y_1) is over, the FSM directs the loading of HREG and XREG with the new set of coefficient and data values corresponding to the outputs y_2 and y_3 . This sequence has to be repeated for all future outputs. The accumulator acc0

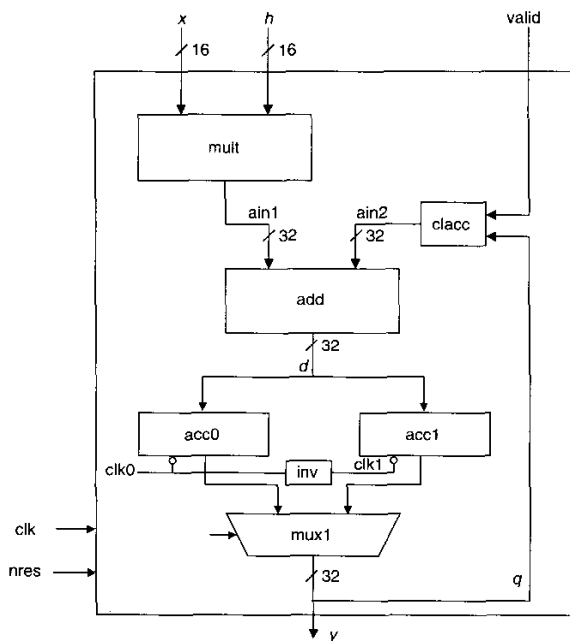


Fig. 4 AU for block processing

and acc1 are clocked by complementary clocks, namely clk0 and clk1 generated by dividing the main clock, clk, by two.

6 Arithmetic unit architecture for the combination of the two algorithms

The architectures corresponding to coefficient segmentation and block processing can be combined together to yield even more significant reduction in power with a slight area overhead. The architecture of the AU for the combination of block processing and coefficient segmentation is shown in Fig. 5. It is basically the combination of the architectures described in Figs. 3 and 4. The power saving occurs not only by segmenting the coefficients but also by holding the segmented coefficients and data values at the input of the multiplier for two clock cycles rather than one.

7 Results

The different FIR cores for all the algorithms, namely conventional (CON), coefficient segmentation (CSEG), block processing (BP), and the combination of CSEG and BP (COMB), have been analysed with regard to area usage and power consumption. The cores were designed using verilog HDL and then synthesised using Ambit BuildGates™, targeting the UMC 0.18µm standard cell CMOS library. The requirements for the synthesis were identical for all the cores. This was necessary in order to allow for a consistent power consumption and area usage comparisons. A maximum circuit delay of 35 ns has been defined for all the cores. A layout for each core was generated using the Envisia™ Silicon Ensemble™ place-and-route software. This was followed by extracting RC information and then performing RC back-annotated post-layout gate-level netlist simulations for a uniformly

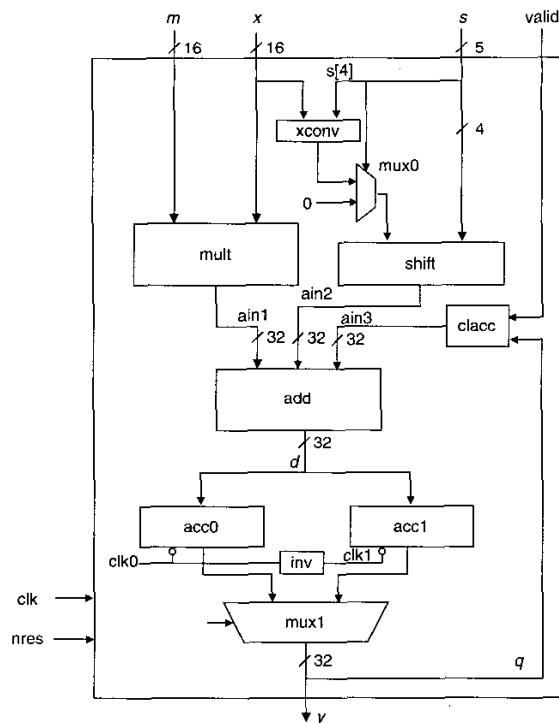


Fig. 5 AU for combination of block processing and coefficient segmentation

distributed random input data sample set equal to 1000 using Verilog-XL™ simulator. The resulting data, including switching activity of the circuit nets and the capacitive load information extracted from the layouts, was then used by the Synopsys DesignPower™ tool to compute power consumption figures for the different FIR cores. In all of the above stages a clock rate of 10 MHz and a supply voltage of 1.8 V were used. This is a typical frequency for the type of hearing aid device where these filters are targeted.

The results are shown in Tables 1–6 and Figs. 6 and 7 for a 73-tap band-pass filter. The power saving in this filter was gauged for three different multiplier types from Synopsys DesignWare: namely, *csa*, *nbw* and *wall*. Tables 1 and 2 give the power consumption for the different blocks of the overall filter and the AU, respectively, using a *csa* multiplier type. According to Table 1 and Fig. 6, the power consumption of the AU is in the range of (65–75%) of the overall FIR core power. It is also evident from Table 2 and Fig. 7 that the multiplier contributes most to the power consumption (40–76%) in the AU, and hence all power saving efforts must be directed to reduce its power consumption. Table 3 shows that the percentage saving in power at the multiplier is substantially higher for CSEG as compared to BP for the *csa* and *nbw* multiplier types, whereas the reverse is true for the *wall* multiplier. The power saving at the multiplier will be maximum for the COMB. The higher savings in power at the multiplier side in CSEG is primarily attributed to the significant reduction in the switching activity and to the effective word-length of its coefficient input. The performance of the *csa* multiplier type is best in terms of power reduction, whereas the worst performance is obtained for the *wall* multiplier. The overall power reduction of the different filter cores is listed in Table 4. This Table shows that the overall power reduction is more in BP compared with CSEG for all the multiplier types. This is because, in BP, power saving occurs not only in the multiplier but also on the address and data buses for both data and coefficient memories on account of fewer memory

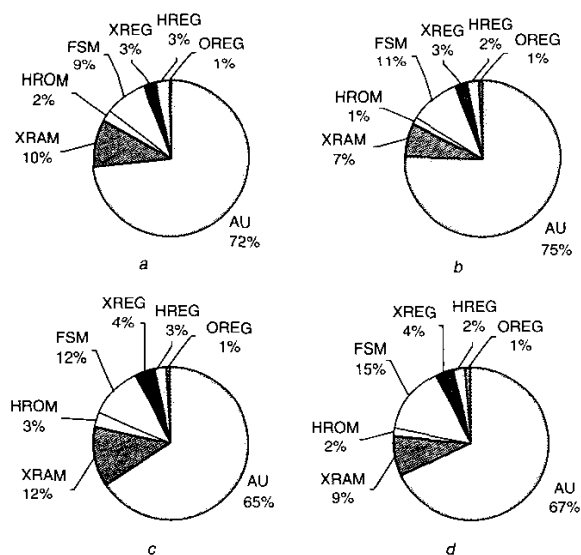


Fig. 6 Distribution of cell power for the overall FIR core
a) CON
b) BP
c) CSEG
d) COMB

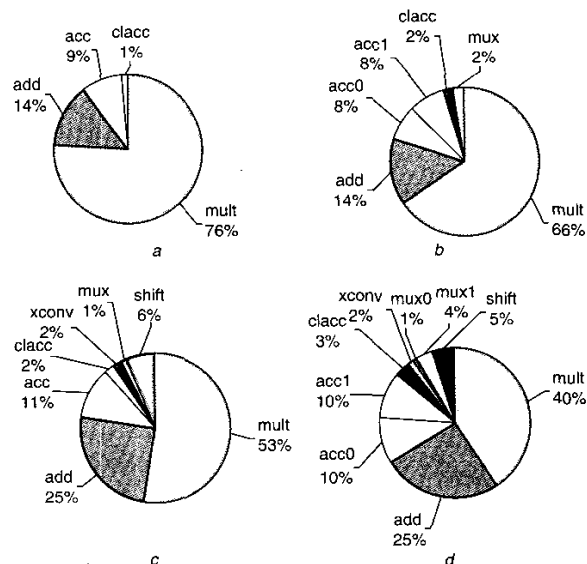


Fig. 7 Distribution of cell power for AU
a) CON
b) BP
c) CSEG
d) COMB

Table 1: FIR cell power

Cell	CON	CSEG	BP	COMB
AU	0.896	0.631	0.712	0.504
XRAM	0.122	0.121	0.065	0.065
HROM	0.026	0.026	0.013	0.012
FSM	0.105	0.111	0.100	0.108
XREG	0.034	0.036	0.026	0.027
HREG	0.031	0.025	0.020	0.018
OREG	0.007	0.007	0.008	0.008
Total	1.221	0.957	0.944	0.742

Table 2: AU cell power

Cell	CON	CSEG	BP	COMB
mult	0.682	0.331	0.466	0.207
add	0.126	0.157	0.103	0.127
acc0	0.077	0.071	0.055	0.051
acc1	—	—	0.055	0.051
clacc	0.012	0.012	0.015	0.016
xconv	—	0.013	—	0.008
mux0	—	0.006	—	0.004
mux1	—	—	0.017	0.018
shift	—	0.041	—	0.024
Total	0.896	0.631	0.712	0.504

accesses. This is evident from the power consumption of the two memory blocks, namely HROM and XRAM in Table 1. Moreover, it is clear from Table 5 that the extra hardware introduced to support CSEG is around 4% as

Table 3: Power consumption analysis for different multipliers

Algorithm	csa		nbw		wall	
	(mW)	(%)	(mW)	(%)	(mW)	(%)
CON	0.682	—	0.566	—	0.687	—
CSEG	0.331	51	0.325	43	0.597	13
BP	0.466	32	0.388	31	0.438	36
COMB	0.207	70	0.227	60	0.349	49

Table 4: Power consumption analysis for FIR cores

Algorithm	csa		nbw		wall	
	(mW)	(%)	(mW)	(%)	(mW)	(%)
CON	1.221	—	1.082	—	1.219	—
CSEG	0.957	22	0.958	11	1.263	-4
BP	0.944	23	0.844	22	0.898	26
COMB	0.742	39	0.765	29	0.901	26

Table 5: Area comparisons for FIR cores

Algorithm	csa		nbw		wall	
	(mm ²)	(%)	(mm ²)	(%)	(mm ²)	(%)
CON	0.184	—	0.186	—	0.186	—
CSEG	0.191	-4	0.193	-4	0.194	-4
BP	0.190	-3	0.192	-3	0.192	-3
COMB	0.197	-7	0.199	-7	0.199	-7

Table 6: Delay analysis for AUs using different multipliers

Algorithm	Delay (ns)		
	csa	nbw	wall
CON	10.01	9.02	9.61
CSEG	9.94	9.70	9.70
BP	10.03	9.05	9.64
COMB	9.98	9.72	9.72

compared with only 3% for BP. This additional hardware in the form of shifter, two's complementor, three-input adder etc. (listed in Table 2) is also responsible for the inferior power saving in CSEG over BP. The overall power consumption of the *wall*-based FIR core for CSEG has gone up over CON because the power saving in the multiplier is just 13%. This is not good enough to offset the power consumed by the additional hardware provided to support the algorithm. The best power savings of 39% and 29% are obtained for COMB with *csa* and *nbw* multiplier types, respectively. This is not true for the *wall* multiplier type because of the poor performance of CSEG with this type of multiplier. The best performance with the *wall* multiplier is obtained by BP both in terms of power and area. The area overhead as per Table 5 is around 3% for BP compared with 4% for CSEG for all multiplier types. The

results have proved that these algorithms are capable of reducing power consumption in the range of 11–39% with a small increment in area. It is finally inferred from the Tables that the best performance is obtained with COMB using *csa* and *nbw* multipliers at an expense of 7% increase in area, whereas BP gives the best result using the *wall* multiplier at the expense of only 3% increase in area. Any of these cores can be used for low power, depending on the area overhead budget. The results obtained have also been verified with other filter lengths and data samples in addition to the filter considered here.

The delay analysis for AUs using different multipliers is given in Table 6. It is clear from the table that the difference in delays is negligible for the different algorithms and the multipliers. The critical path for the faster *nbw* and *wall* multipliers runs through the shifter-adder combination rather than the multiplier-adder combination. This is due to the reduction in the delay of the multiplier on account of its smaller size in the case of CSEG and COMB algorithms. Therefore, the critical path for these algorithms remains the same for *nbw* and *wall* multipliers. The critical path for these cases is slightly longer than the CON algorithm, due to the additional delay introduced by a three-input adder. The critical path in case of a slower *csa* multiplier for CSEG and COMB algorithms runs through the multiplier-adder combination and hence is longer than for the other multipliers. The critical path for CON, CSEG and COMB algorithms remains almost same for the *csa* multiplier because the reduction in the delay of the smaller multiplier is partially compensated by the delay introduced by the three-input adder. In the case of BP, the critical path is almost same as CON for all the multipliers. The slight increase of critical path in BP is attributed to the higher fanout of the adder in the form of two accumulators rather than one.

8 Conclusions

Novel FIR filter cores have been developed, based on a number of low power algorithms. The algorithms can be used alone or in combination depending upon the power and area constraints. For instance, coefficient segmentation based FIR core provides up to 22% power reduction with an area overhead of 4%. A block processing-based core, on the other hand, results in a power reduction of up to 26% at the expense of only 3% in area. However, if more power saving is desired, the combination of coefficient segmentation and block processing based core can be used, providing up to 39% power reduction, but at the cost of 7% increase in area.

9 Acknowledgment

Dr A.T. Erdogan would like to thank the UK Engineering and Physical Sciences Research Council (EPSRC) for supporting this work under grant number GR/N08332.

10 References

- 1 Sankarayya, N., Roy, K., and Bhattacharya, D.: 'Algorithms for low power and high speed FIR filter realisation using differential coefficients', *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, 1997, **44**, pp. 488–497
- 2 Choi, H., and Burleson, W.P.: 'Search-based wordlength optimisation for VLSI/DSP synthesis', *VLSI Signal Process.*, 1994, **7**, pp. 198–207
- 3 Mehendale, M., Sherlekar, S.D., and Venkatesh, G.: 'Coefficient optimisation for low power realisation of FIR filters', Proceedings of IEEE workshop on VLSI signal processing, 1995, pp. 352–361
- 4 Erdogan, A.T., and Arslan, T.: 'Low power multiplication scheme for FIR filter implementation on single multiplier CMOS DSP processors', *Electron. Lett.*, 1996, **32**, (21), pp. 1959–1960

- 5 Erdogan, A.T., and Arslan, T.: 'Low power implementation of linear phase FIR filters for single multiplier CMOS based DSPs'. Proceedings of the IEEE International Symposium on circuits and systems, June 1998, pp. D425–D428
- 6 Wu, A.Y., Liu, K.J.R., Zhang, Z., Nakajima, K., Raghupathy, A., and Liu, S.C.: 'Algorithm-based low-power DSP system design: methodology and verification', VLSI Signal Processing, Vol. 8. (IEEE Press, 1995)
- 7 Mehendale, M., Sheriekar, S.D., and Venkatesh, G.: 'Low power realisation of FIR filters using multirate architectures'. Proceedings of 9th International Conference on VLSI design, Jan. 1996, pp. 370–375
- 8 Ludwig, J.T., Nawab, S.H., and Chandrakasan, A.P.: 'Low power digital filtering using approximate processing', *IEEE J. Solid-State Circuits*, 1996, 31, pp. 395–399
- 9 Erdogan, A.T., and Arslan, T.: 'A coefficient segmentation algorithm for low power implementation of FIR filters'. Proceedings of the IEEE International Symposium on Circuits and systems, 1999, pp. III.359–III.362
- 10 Erdogan, A.T., and Arslan, T.: 'Data block processing for low power implementation of direct form FIR filters on single multiplier CMOS based DSPs'. Proceedings of the IEEE international symposium on circuits and systems, June 1998, pp. D441–D444