VLSI Design, 2002 Vol. 15 (2), pp. 529–535

Taylor & Francis
Taylor & Francis Group

A Combined Coefficient Segmentation and Block Processing Algorithm for Low Power Implementation of FIR Digital Filters*

A.T. ERDOGAN[†] and T. ARSLAN[‡]

University of Edinburgh, Department of Electronics and Electrical Engineering, Edinburgh EH9 3JL, Scotland, UK

(Received 15 November 2000; Revised 17 May 2001)

A combined coefficient segmentation and block processing algorithm for low power implementation of FIR digital filters is described in this paper. The algorithm processes data and coefficients in blocks of fixed sizes. During the manipulation of each block, coefficients are segmented into two primitive components. The accumulative effect of processing a sequence of blocks and segmentation results in up to 80% reduction in power consumption in the multiplier circuit compared to conventional filtering. The paper describes the implementation of the algorithm, its constituent components, and the power evaluation environment developed. Simulations are performed using eight practical digital filter examples with various filter orders and data/coefficient wordlengths. In addition, the algorithm is compared with conventional filtering implementations and those using block processing and coefficient segmentation algorithms alone.

Keywords: Low power; FIR; VLSI; Coefficient segmentation; Block processing

INTRODUCTION

The demand for high performance portable systems incorporating multimedia capabilities has elevated the design for low power to the forefront of design requirement in order to maintain reliability and provide longer hours of operation [1]. Multimedia applications demand real-time signal processing, which consists of intensive multiply and multiply–accumulate operations unique to digital signal processing (DSP) algorithms, such as filtering, fast Fourier transforms, and discrete cosine transforms. For this reason, the multiplication procedure plays a key role in achieving low power implementation of these algorithms. In fact, in many DSP algorithms, such as filtering, the multiplier lies in the critical delay path and ultimately determines the performance of the algorithm. These DSP algorithms are implemented mainly on CMOS-based VLSI devices, which could be dedicated ASICs or DSP processors. Such devices typically integrate parallel multipliers as the central units to handle the computational burden [2].

It can be shown that the most significant factor affecting power consumption in a CMOS VLSI device is the switching power, which is expressed by the following equation [3]:

$$P_{sw} = \frac{1}{2} k C V_{dd}^2 f \tag{1}$$

where C is the physical capacitance, V_{dd} is the supply voltage, f is the frequency and k is the switching activity factor, which is defined as the average number of times that a gate makes a logic transition ($1 \rightarrow 0$ or $0 \rightarrow 1$) in each clock cycle. Therefore, one or more of these parameters must be targeted in order to reduce the power consumption of a circuit.

Researchers in the literature have developed a number of techniques for low power implementation of digital filters. The authors in Ref. [4] utilise a technique which 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.

*Based on "A Hybrid Segmentation and Block Processing Algorithm for Low Power Implementation of Digital Filters" by A.T. Erdogan and T.Arslan which appeared in *Proceedings of International Symposium on Circuits and Systems*, Geneva, Switzerland, May 28–31, 2000, Pages V-349–352.

[†]E-mail: ahmet.erdogan@ee.ed.ac.uk
[‡]Corresponding author. Tel.: +44-131-650-1000. Fax: +44-131-650-6554E-mail: tughrul.arslan@ee.ed.ac.uk

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/1065514021000012147

Another approach used in Ref. [5] is to optimise wordlengths 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 wordlength parameterisation. In Ref. [6], Mehendale *et al.* present an algorithm for optimising the coefficients of an FIR filter, so as to reduce power consumption of 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. The authors in Ref. [7] developed a dynamic programming algorithm to assist in the search for an optimal coefficient set whose member coefficients are restricted to the set $\{-1, 0, 1\}$. The authors in Ref. [8] have investigated the use of primitive operator technique in area/power reduction. In Refs. [9,10], we 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 [11,12], and dynamic adjustment of filter order for adaptive filters [13].

We have proposed block processing [14] and coefficient segmentation [15] algorithms for low power implementation of FIR filters. Both algorithms reduce power by reducing the amount of switched capacitance within the DSP hardware platform utilised for filter implementation. Block processing reduces switched capacitance within the multiplier unit and data/coefficient busses. Whereas, coefficient segmentation reduces switched capacitance within the multiplier unit in addition to a reduction in the effective wordlength of the coefficient input to the multiplier.

In this paper, we present an algorithm that utilises block processing and coefficient segmentation in a hierarchical framework to combine their advantages for more reduction in power. The algorithm processes data and coefficients in blocks of fixed sizes. During the manipulation of each block, coefficients are segmented into two primitive components. The accumulative effect of processing a sequence of blocks and the segmentation results in up to 80% reduction in power consumption compared to conventional filtering. Power reduction is achieved through a reduction in the amount of switched capacitance within the multiplier section and on both data and coefficient busses. This in turn is due to less data and coefficient memory access operations and reduced switching activity at multiplier inputs.

IMPLEMENTATION

A typical single multiplier DSP processor architecture for the implementation of FIR filters consists of input/output units, data and coefficient memories, a multiplier–accumulator (MAC) unit, and a control unit [9]. In the direct form realisation of the filter a new data sample, $x(n)$, and the corresponding coefficient, $h(k)$, are multiplied at each clock cycle. For this reason each time a multiplication is performed both inputs of the multiplier receive new data. This continuous change at both inputs of the multiplier leads to a relatively high overall switching activity within the multiplier and hence a correspondingly high power consumption. Therefore, any multiplication strategy, which could reduce the switching activity for this realisation is highly desirable. Another source of power consumption in DSPs is the activity in data and address buses. Since each time a new data sample is to be multiplied with a new coefficient, both data and address busses experience high switching activity. This has significant power overheads since bus capacitances are usually several orders of magnitude higher than those of the internal gates of a circuit. Consequently, a considerable amount of power can be saved by reducing the number of memory accesses.

The coefficient segmentation algorithm reduces the switched capacitance by decomposing individual coefficients into two less complex sub-components. The decomposition, performed using a heuristic approach, separates a given coefficient such that a part is produced which can be implemented using a single shift operation leaving another part with reduced wordlength to be applied to the inputs of the hardware multiplier. Hence, resulting in a significant reduction in the amount of switched capacitance and consequently power consumption. The flow chart in Fig. 1 shows the main stages (indicated in circles) of the algorithm developed. Given the coefficient set $H = (h_0, h_1, \ldots, h_{L-1})$, where L is the filter order, the algorithm proceeds through the coefficients sequentially. For a given coefficient h_k, the algorithm targets dividing it such that $h_k = s_k + m_k$, where s_k is the component to be implemented using a shift operation and m_k is the data to be applied to the hardware multiplier. In order to reduce the switched capacitance of the multiplier consecutive values of m_k must be of the same polarity, to minimise switching activity at the multiplier inputs, and have the smallest value possible, to minimise effective wordlength. This criteria can be met by careful selection of s_k and consequently m_k values. This selection procedure is the pivot of the stages shown in Fig. 1. For a small positive m_k, s_k must be the largest power-of-two number closest to h_k. For this reason, stage 1 is an iterative procedure which aims to find the largest power-of-two number greater than or equal to $|h_k|$. Stage 2 deals with coefficients which are already power-of-two numbers, in which case the complete coefficient is realised using a single shift operation (i.e. $s_k = h_k$ and $m_k = 0$). In stage 3, the polarity of h_k is monitored. If h_k is a positive

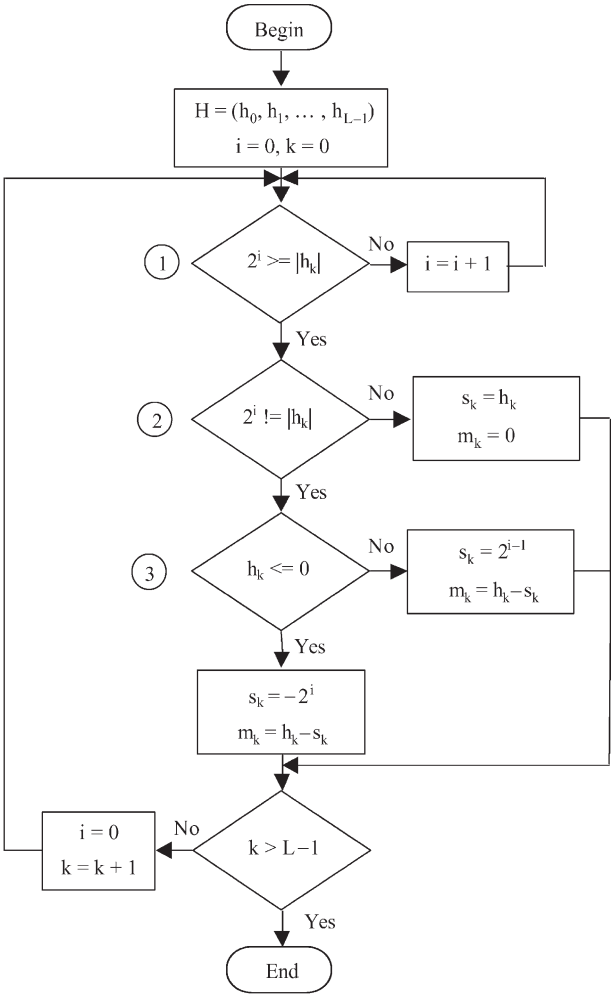

FIGURE 1 Flowchart of the coefficient segmentation algorithm.

number then s_k is chosen as the largest power-of-two number smaller than h_k (i.e. $s_k = 2^{i-1}$). On the other hand, if h_k is negative, s_k is chosen to be the smallest power-of-two number larger than $|h_k|$ (i.e. $s_k = -2^i$). In both cases, m_k is equal to $(h_k - s_k)$.

The block processing algorithm requires a number of accumulators and a bank of registers (often called a register file) that can be used as operands for arithmetic and multiplication operations [14]. This facility is available in a number of DSPs, e.g. Texas Instruments TMS320C54x, NEC μPD7701x, Zoran ZR3800x, AT&T DSP16xx, and Motorola DSP5600x. Data block processing commences when a coefficient and L data samples are fetched from the memory and stored into registers in the register file. Next, these data samples are presented to the multiplier through the registers and multiplied with the same coefficient one after the other and their products are added to their respective accumulators. This is repeated for each coefficient, with each time only one data sample (in the block) being replaced with a new one. This reduces the switching activity at coefficient inputs of the multiplier, since the same coefficient is used for all data samples in the block. In addition, less memory accesses to

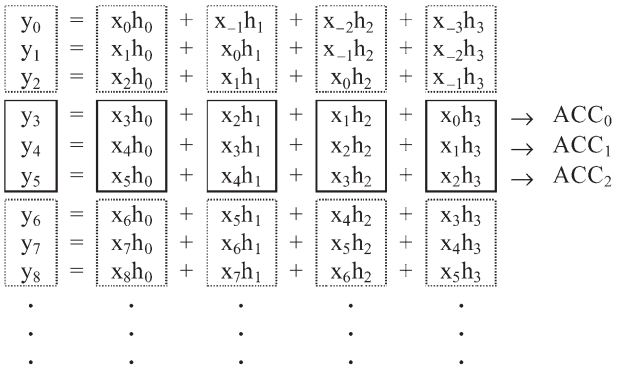

FIGURE 2 An example of a 4-tap filter with $L = 3$.

both data and coefficient memories are required since coefficient and data samples are obtained through registers. It is well known that register operations consume less power than memory operations [16]. Figure 2 shows the scheme with a block size of 3, $L = 3$, for an example of a 6-tap filter, $N = 6$, at a given instant in time, $n = 3$.

When both block processing and coefficient segmentation algorithms are combined a more powerful algorithm will emerge in which multiplications could be processed in blocks of fixed sizes, leading to a reduction in switched capacitance within multiplier and data and coefficient memory buses. Each individual multiplication operation in turn is segmented for more reduction in switched capacitance within the multiplier circuit.

Using the combined algorithm the filtering commences by fetching s_k and m_k values and applying these to both shifter and multiplier inputs, respectively. Next, a block of L data samples $(x_0, x_1, \ldots, x_{L-1})$ are fetched from the data memory and stored in the register file. This is followed by applying the first data sample, x_0, in the register file to both shifter and multiplier units. The resulting values from both shifter and multiplier units are then summed together and the final result is added to the first accumulator. This is

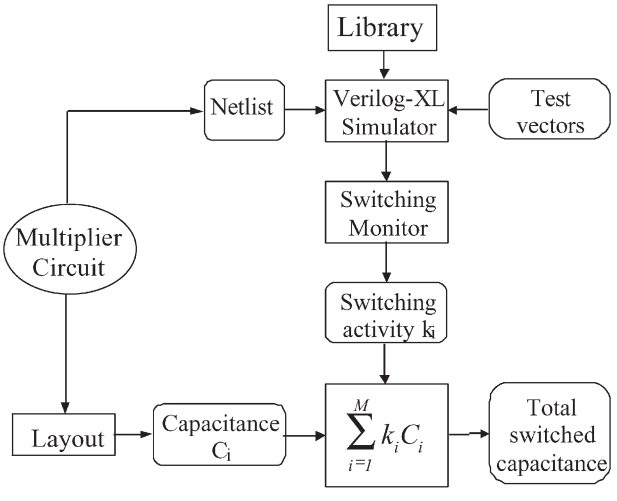

FIGURE 3 Framework for the power consumption evaluation.

TABLE I Power reduction results

		Multiplier size					
		8-Bit		16-Bit		24-Bit	
Algorithm	Block size	swcap/sample	Reduction (%)	swcap/sample	Reduction (%)	swcap/sample	Reduction (%)
C	–	362	–	2577	–	9936	–
B	2	242	33.15	1870	27.44	7658	22.93
B	4	281	22.38	2091	18.86	8241	17.06
B	8	268	25.97	2010	22.00	7963	19.86
B	16	261	27.90	1970	23.55	7823	21.27
S	–	168	53.59	1534	40.47	7822	21.28
B&S	2	102	71.82	1121	56.50	6032	39.29
B&S	4	150	58.56	1240	51.88	6477	34.81
B&S	8	148	59.12	1191	53.78	6257	37.03
B&S	16	146	59.67	1167	54.71	6146	38.14

repeated with the second data sample, x_1, in the register file and the final result of this is added to the second accumulator. All data samples in the register file are processed in a similar manner. Next, new s_k and m_k values are fetched and processed with almost the same data samples in the register file, in a manner similar to above. The contents of the register file is updated with the addition of a single new data entry which will replace the first entry in the previous cycle. This procedure reduces the switching activity at coefficient inputs of the multiplier for the following reasons: (a) the same coefficient is used for all data samples in the block, (b) the wordlength of the segmented coefficient (m_k) is less than the original coefficient (h_k), (c) the Hamming distance between consecutive coefficients (m_k values) is reduced. In addition, less memory accesses to both data and coefficient memories are required since coefficient and data samples are obtained through internal registers.

The sequence of steps for the algorithm can be summarised as follows:

1. Clear all accumulators (ACC_0 to ACC_{L-1}).
2. Get the multiplier part, $m(N-1)$, of the coefficient $h(N-1)$ and apply it to the coefficient input of the multiplier.
3. Get the shifter part, $s(N-1)$, of the coefficient $h(N-1)$ and apply this to the control inputs of the shifter.
4. Get data samples $x[n-(N-1)], x[n-(N-2)], \ldots, x[n-(N-L)]$ and store these into data registers $R_0, R_1, \ldots, R_{L-1}$, respectively. This will form the first block of data samples.
5. Apply R_0 to both the multiplier and the shifter units. Add their results and the content of accumulator ACC_0 together and store the final result into accumulator ACC_0. Repeat this for the remaining data registers, R_1–R_{L-1}, this time using accumulators ACC_1 to ACC_{L-1}, respectively.
6. Get the multiplier part, $m(N-2)$, and the shifter part, $s(N-2)$, of the next coefficient, $h(N-2)$, and apply these to the multiplier and the shifter inputs, respectively.
7. Update the data block formed in step (4) by getting the next data sample, $x[n-(N-L-1)]$, and storing it in data register R_0 overwriting the oldest data sample in the block.

TABLE II Power reduction results with multiplier inputs swapped

		Multiplier size					
		8-Bit		16-Bit		24-Bit	
Algorithm	Block size	swcap/sample	Reduction (%)	Swcap/sample	Reduction (%)	swcap/sample	Reduction (%)
C	–	380	–	2729	–	9739	–
B	2	249	34.47	1941	28.88	7189	26.18
B	4	294	22.63	2132	21.88	7831	19.59
B	8	279	26.58	2033	25.50	7518	22.81
B	16	272	28.42	1983	27.34	7359	24.44
S	–	131	65.53	1304	52.22	6385	34.44
B&S	2	76	80.00	898	67.09	4722	51.51
B&S	4	113	70.26	974	64.31	5092	47.72
B&S	8	110	71.05	918	66.36	4878	49.91
B&S	16	109	71.32	891	67.35	4770	51.02

TABLE III Effect of swapping multiplier inputs on power consumption

Algorithm	Block size	Multiplier size: 8-Bit swcap	8-Bit swcap for swapped inputs	8-Bit Reduction (%)	16-Bit swcap	16-Bit swcap for swapped inputs	16-Bit Reduction (%)	24-Bit swcap	24-Bit swcap for swapped inputs	24-Bit Reduction (%)
C	–	362	380	−4.97	2577	2729	−5.90	9936	9739	1.98
B	2	242	249	−2.89	1870	1941	−3.80	7658	7189	6.12
B	4	281	294	−4.63	2091	2132	−1.96	8241	7831	4.98
B	8	268	279	−4.10	2010	2033	−1.14	7963	7518	5.59
B	16	261	272	−4.21	1970	1983	−0.66	7823	7359	5.93
S	–	168	131	22.02	1534	1304	14.99	7822	6385	18.37
B & S	2	102	76	25.49	1121	898	19.89	6032	4722	21.72
B & S	4	150	113	24.67	1240	974	21.45	6477	5092	21.38
B & S	8	148	110	25.68	1191	918	22.92	6257	4878	22.04
B & S	16	146	109	25.34	1167	891	23.65	6146	4770	22.39

8. Process the new data block as in step (5). However, start processing with R_1, followed by $R_2, \ldots, R_{L-1}$, and R_0, in a circular manner. During this procedure use accumulators in the same order as data registers, e.g. first ACC_1, then followed by $ACC_2, \ldots ACC_{L-1}$, and finally ACC_0.
9. Process the remaining multiplier and shifter parts as in steps (6) to (8).
10. Get the first block of filter outputs, $y(n), y(n-1), \ldots, y(n-L)$, from $ACC_0, ACC_1, \ldots, ACC_{L-1}$, respectively.
11. Increment n by L and repeat steps (1) to (10) to obtain the next block of filter outputs.

SIMULATIONS AND RESULTS

To demonstrate our results, we have implemented a two's complement (Baugh–Wooley) array multiplier which was selected as an example of a commonly used multiplier in DSP implementation. 8 × 8, 16 × 16 and 24 × 24-bit multipliers were implemented using Cadence VLSI suite with 0.7 μm CMOS technology. The multipliers were constructed using AND, OR, XOR and INVERTER gates only. Coefficient sets were obtained by designing eight practical FIR filters with the Remez exchange algorithm developed by Parks and McClellan [17]. These are:

(1) A low-pass filter with a filter order of $N = 24$.
(2) A band-pass filter with $N = 32$.
(3) A band-pass filter with $N = 50$, in which unequal weighting is used in the two stop-bands. Thus the peak error in the upper stop-band is ten times smaller than the peak error in the lower stop-band.
(4) A band-stop filter for $N = 31$ and with equal weighting in both pass-bands.
(5) A five-band filter for $N = 55$ with three stop-bands and two pass-bands. The weighting in each of the stop-bands is different, making the peak approximation error differ in each of these bands.
(6) A full band differentiator for $N = 32$ and the peak approximation error = 0.0062.
(7) A Hilbert transformer for $N = 20$ and the peak approximation error = 0.02, where the upper cutoff frequency is 0.5 and the lower cutoff frequency is 0.05.
(8) A band-pass filter with an arbitrary weighting function and for $N = 128$.

These benchmark examples were obtained from Ref. [17]. The coefficient sets were quantised to 8, 16, and 24-bits. This was followed by generating zero mean uniformly distributed data samples for each filter. Next the coefficient sets were processed by the segmentation algorithm in order to produce s_k and m_k values for each coefficient h_k. This was followed by generating input simulation files, in which the generated input data samples were associated with the corresponding m_k values for a

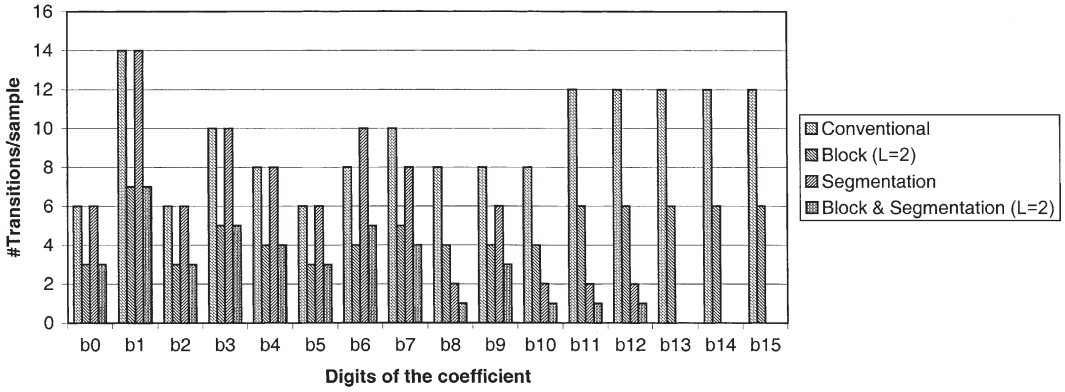

FIGURE 4 Switching activity distribution (Filter order $N = 32$, Wordlength $W = 16$).

given filter, for the Cadence's Verilog-XL™ digital simulator. Verilog-XL uses the gate level netlist of the multiplier circuit for the simulation procedure. For each simulation the number of signal transitions was monitored. Capacitive information (wiring and loading capacitances) for each gate was extracted by performing a layout of the multiplier circuit. Both capacitive information and the switching activity figure were used to obtain the switched capacitance of each gate. This was then accumulated to give an overall figure for the switched capacitance of the multiplier (see Fig. 3).

The average results for the eight filter examples are shown in Table I for different multiplier sizes. Power reduction figures are obtained by comparing switched capacitance values to that of the conventional filtering (C), where data and coefficient values are directly applied to the multiplier. The table illustrates the amount of switched capacitance per sample and the percentage reduction in switched capacitance for the following cases:

(a) Block processing (B) alone with block sizes of 2, 4, 8, and 16.
(b) Coefficient segmentation (S) alone.
(c) Block processing and coefficient segmentation (B&S) together with block sizes of 2, 4, 8 and 16.

In the case of 8 × 8-bit multiplier, block processing alone achieves a maximum of 33.15% reduction for a block size of 2. On the other hand, coefficient segmentation alone achieves 53.59% reduction. However, when block processing is used together with coefficient segmentation the reduction is increased to 71.82% for a block size of 2. The power reduction profile for 16 × 16 and 24 × 24-bit multipliers is similar to that of the 8 × 8-bit multiplier case, where the best reductions (56.50 and 39.29%, respectively) are achieved using block processing together with coefficient segmentation.

The above simulations were repeated after swapping the multiplier inputs in order to examine the effect on power reduction. The results are shown in Table II. It can be seen that the reductions in switched capacitance are increased in all cases. Specifically, the best reductions for 8 × 8, 16 × 16 and 24 × 24-bit multiplier cases are increased to 80.00, 67.09 and 51.51%, respectively.

Table III illustrates the effect of swapping multiplier inputs on the switched capacitance. It clearly demonstrates that the switched capacitance has increased for both conventional filtering and block processing cases, whereas it decreased for both coefficient segmentation alone and block processing with coefficient segmentation together, in various degrees, in the cases of 8 × 8 and 16 × 16-bit multipliers. In the case of 24 × 24-bit multiplier, switched capacitance decreased in all cases. Our analysis revealed that this could be due to the fact that switching activity at coefficient input bits of the multiplier is not uniform. This can clearly be seen in Fig. 4, where in the upper half of the coefficient word switching activity is higher for conventional filtering, and lower for both segmentation alone and block processing with segmentation together. The figure also demonstrates the reductions both in switching activity and effective wordlength resulting from the use of our algorithm.

There is an overhead element associated with the algorithm. This is mainly due to the added shift operations imposed by coefficient segmentation. However, it could be shown that this is typically under 4% [15].

CONCLUSIONS

The authors present a combined block processing and coefficient segmentation algorithm for low power implementation of FIR filters. Low power consumption is achieved through a reduction in the amount of switched capacitance within the multiplier circuit and data/coefficient memory buses. This reduction in switched capacitance is achieved within a hierarchical framework in which coefficients, processed in fixed-size blocks, are segmented into sub-components that are less computationally complex. The algorithm is compared to conventional filtering implementations and those using block processing and coefficient segmentation alone. Results, obtained with different block and multiplier sizes, indicate up to 80% reduction in power consumption.

References

[1] Abu-khater, I.S., Bellaouar, A. and Elmasry, M.I. (1996) "Circuit techniques for CMOS low-power high-performance multipliers", *IEEE Journal of Solid-State Circuits* **31**(10), 1535–1546.

[2] Nadehar, K., Kuroda, I., Daito, M. and Nakayama, T. (1995) "Low-power multimedia RISC", *IEEE Micro* **15**(6), 20–28.

[3] Chandrakasan, A.P. and Brodersen, R.W. (1995) "Minimising power consumption in digital CMOS circuits", *Proceedings of IEEE* **83**(4), 498–523.

[4] Sankarayya, N., Roy, K. and Bhattacharya, D. (1997) "Algorithms for low power and high speed fir filter realisation using differential coefficients", *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing* **44**(6), 488–497.

[5] Choi, H. and Burleson, W.P. (1994) "Search-based wordlength optimisation for VLSI/DSP synthesis", *VLSI Signal Processing VII*, (IEEE Press), pp 198–207.

[6] Mehendale, M., Sherlekar, S.D. and Venkatesh, G. (1995) "Coefficient optimisation for low power realisation of FIR filters", *IEEE Workshop on VLSI Signal Processing*, 352–361.

[7] Hezar, R. and Maisetti, V.K. (1996) "Low-power digital filter implementations using ternary coefficients", VLSI Signal Processing IX (IEEE Press), pp 179–188.

[8] Horrocks, D.H. and Wongsuwan, Y. (1998) "Power dissipation in primitive operator FIR filters", UK Low-Power Forum (Sheffield University, UK), pp 141–144.

[9] Erdogan, A.T. and Arslan, T. (1996) "Low power multiplication scheme for FIR filter implementation on single multiplier CMOS DSP processors", *IEE Electronics Letters* **32**(21), 1959–1960.

[10] Erdogan A.T. and Arslan T. "Low power implementation of linear phase FIR filters for single multiplier CMOS based DSPs", *IEEE Int. Symposium on Circuits and Systems (ISCAS'98)*, 31 May–3 June 1998, Monterey, California, USA, pp. D425–D428.

[11] Wu, A-Y., Liu, K.J.R., Zhang, Z., Nakajim, K., Raghupathy, A. and Liu, S-C. (1995) "Algorithm-based low-power DSP system design: methodology and verification", In: Nishitani, T. and Parhi, K.K., eds, VLSI Signal Processing VIII (IEEE Press).

[12] Mehendale, M., Sherlekar, S.D. and Venkatesh, G. (1996) "Low power realisation of FIR filters using multirate architectures", *9th International Conference on VLSI Design*, 370–375.

[13] Ludwig, J.T., Nawab, S.H. and Chandrakasan, A.P. (1996) "Low power digital filtering using approximate processing", *IEEE Journal of Solid-State Circuits* **31**(3), 395–399.

[14] Arslan T. and Erdogan A.T. (1998) "Data block processing for low power implementation of direct form FIR filters on single multiplier CMOS DSPs", *IEEE International Symposium on Circuits and Systems (ISCAS'98)*, Monterey, California, USA, pp. D441–D444.

[15] Erdogan A.T. and Arslan T. "A coefficient segmentation algorithm for low power implementation of FIR filters", *IEEE International Symposium on Circuits and Systems (ISCAS'99)*, May 1999, Florida, USA, pp. 359–362.

[16] Tiwari, V., Malik, S. and Wolfe, A. (1994) "Power analysis of embedded software, a first step towards software power minimization", *IEEE Transactions on VLSI Systems* **2**(4), 437–445.

[17] Mcclellan, J.H., Parks, T.W., Rabiner, L.R. and Computer, A. (1973) "Program for designing optimum FIR linear phase digital filters", *IEEE Transactions on Audio and Electroaccoustics* **AU-21**(6), 506–526.

Dr Ahmet Teyfik Erdogan is a Research associate at the University of Edinburgh. He has M.Sc. and Ph.D. degrees in Electronics Engineering from the University of Cardiff, UK. His research interests include *HDL based Design Methodologies*, *DSP Architectures*, *Low Power VLSI Design and Macro IPs*.

Dr Tughrul Arslan is a Reader at the University of Edinburgh, where he leads the System Level Integration activity. His research interests include *Low Power VLSI Design and DSP*, *System-on-Chip Applications*, *Evolvable Hardware*, *Design Automation and Test*. Dr Arslan is a member of the IEEE and the IEE.