

DATA BLOCK PROCESSING FOR LOW POWER IMPLEMENTATION OF DIRECT FORM FIR FILTERS ON SINGLE MULTIPLIER CMOS DSPs

T. Arslan and A.T. Erdogan

Cardiff University of Wales
Cardiff School of Engineering, PO Box 689,
Cardiff CF2 3TF,
United Kingdom

ARSLAN@Cardiff.ac.uk, ERDOGAN@Cardiff.ac.uk

ABSTRACT

In this paper, the authors propose a block processing scheme for low power implementation of FIR filters on single multiplier CMOS based Digital Signal Processors (DSPs). The authors show that the reduction in overall power is due to a decrease in switching activity at coefficient inputs of the multiplier and both data and coefficient memory buses by a factor determined by the data input block size. Results are presented which demonstrate up to 34% savings in power. The paper presents the scheme, outlines its implementation using an example and demonstrates results with various filter orders and wordlengths.

1. INTRODUCTION

The increase in demand for portable multi-media devices in the past few years has led to the emergence of low power design techniques for the design of such devices [1]. Digital signal processing systems are crucial parts of the above devices and are computationally intensive. For this reason, the majority of the above techniques have targeted the design of such systems. The techniques aim to reduce power at various levels throughout the design hierarchy of a DSP system, ranging from processor level, e.g. optimising instruction sets, to circuit level design optimisation. These techniques are well documented in the literature [1-6].

The authors have proposed a number of multiplication schemes for the low power implementation of FIR filters on single multiplier CMOS based DSPs [7-8]. These schemes have demonstrated a high potential for power reduction achieving up to 63% reduction. However, these schemes are restricted to the transpose direct form realisation of FIR filters. This dictates the need for a *modified* single multiplier DSP processor architecture since it could not be implemented on traditional DSPs using the conventional multiply-accumulate (MAC) unit. Most common single multiplier DSP processors are based

upon the conventional MAC unit. Examples are the TMS series from Texas Instruments and Motorola's DSP series. For this reason, there is a significant demand for low power implementation schemes to target the above processors with minimum (or no) modifications to their architecture.

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 buses 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 concept of filtering or processing number sequences in blocks of fixed size, *block processing*, was developed and used very early in the literature, e.g. [10]. The main objective of block processing is to implement signal processing schemes with high inherent parallelism [9]. A number of researchers have studied block processing methods for the development of computationally efficient high order recursive filters, which are less sensitive to roundoff error and coefficient accuracy [11].

In this paper, the authors investigate the use of *data block processing* for the implementation of low power FIR filters on single multiplier DSPs. During filtering, data samples in fixed size blocks, L , are processed consecutively. The authors show that this procedure reduces power consumption by decreasing the switching activity, by a factor depending on L , in the following: (1) coefficient input of the multiplier, (2) data and coefficient memory buses, (3) data and coefficient address buses. The authors show that the scheme can achieve up to 34% savings in power.

2. IMPLEMENTATION

The scheme 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. This facility is available in a number of DSPs, e.g. Texas Instruments TMS320C54x, NEC μ PD7701x, 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 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 [12]. Our investigations revealed that the number of accesses, required by the scheme, for data and coefficient memories per filter output are as follows:

$$NMA_x = 1 + \frac{N-1}{L} \cong 1 + \frac{N}{L} \quad (1)$$

$$NMA_h = \frac{N}{L} \quad (2)$$

where

NMA_x :	Number of data memory accesses
NMA_h :	Number of coefficient memory accesses
N :	Filter order
L :	Block size

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

1. Get the first filter coefficient, $h(N-1)$.
2. Get data samples $x[n-(N-1)]$, $x[n-(N-2)]$, ..., $x[n-(N-L)]$ and save them into data registers R_0 , R_1 , ..., R_{L-1} respectively.
3. Multiply $h(N-1)$ by R_0 , R_1 , ..., R_{L-1} and add the products into the accumulators ACC_0 , ACC_1 , ..., ACC_{L-1} respectively.
4. Get the second coefficient, $h(N-2)$.
5. Get the next data sample, $x[n-(N-L-1)]$, and place it in R_0 overwriting the oldest data sample in the block.
6. Process $h(N-2)$ as in step (3), however, use registers in a circular manner, e.g. multiply $h(N-2)$ by R_1 , ..., R_{L-1} , R_0 . Their respective products will be added to accumulators ACC_0 , ACC_1 , ..., ACC_{L-1} . Process the remaining coefficients as for $h(N-2)$.
7. Get the output block, $y(n)$, $y(n-1)$, ..., $y(n-L)$, from ACC_0 , ACC_1 , ..., ACC_{L-1} respectively.
8. Increment n by L and repeat steps (1) to (7) to obtain next output block.

Figure 1 illustrates 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, $n=3$. First of all, a block of 3 data samples, e.g. x_2 , x_1 and x_0 , are multiplied by the same coefficient, h_5 , and their respective products are added to accumulators ACC_0 , ACC_1 and ACC_3 respectively. Next, a second set of 3 data samples (x_{-1} , x_0 , x_1) are multiplied by the next coefficient, h_4 , and their products are again added to accumulators ACC_0 , ACC_1 and ACC_3 respectively. Note that this time only one new data sample, x_1 , is fetched from the data memory replacing the oldest sample, x_2 , and the remaining two data samples are reused without the need for fetching them from data memory. In the same way, the third (x_0 , x_1 , x_2), fourth (x_1 , x_2 , x_3), fifth (x_2 , x_3 , x_4) and sixth (x_3 , x_4 , x_5) block of 3 data samples are multiplied by h_3 , h_2 , h_1 and h_0 coefficients and their products are added to their respective accumulators. Finally, a block of 3 filter outputs (y_3 , y_4 , y_5) is obtained from ACC_0 , ACC_1 and ACC_2 respectively.

$$\begin{array}{rcl}
\boxed{y_0} & = & \boxed{x_0h_0} + \boxed{x_1h_1} + \boxed{x_2h_2} + \boxed{x_3h_3} + \boxed{x_4h_4} + \boxed{x_5h_5} \\
\boxed{y_1} & = & \boxed{x_1h_0} + \boxed{x_0h_1} + \boxed{x_1h_2} + \boxed{x_2h_3} + \boxed{x_3h_4} + \boxed{x_4h_5} \\
\boxed{y_2} & = & \boxed{x_2h_0} + \boxed{x_1h_1} + \boxed{x_0h_2} + \boxed{x_1h_3} + \boxed{x_2h_4} + \boxed{x_3h_5} \\
\boxed{y_3} & = & \boxed{x_3h_0} + \boxed{x_2h_1} + \boxed{x_1h_2} + \boxed{x_0h_3} + \boxed{x_1h_4} + \boxed{x_2h_5} \rightarrow \text{ACCR}_0 \\
\boxed{y_4} & = & \boxed{x_4h_0} + \boxed{x_3h_1} + \boxed{x_2h_2} + \boxed{x_1h_3} + \boxed{x_0h_4} + \boxed{x_1h_5} \rightarrow \text{ACCR}_1 \\
\boxed{y_5} & = & \boxed{x_5h_0} + \boxed{x_4h_1} + \boxed{x_3h_2} + \boxed{x_2h_3} + \boxed{x_1h_4} + \boxed{x_0h_5} \rightarrow \text{ACCR}_2 \\
\boxed{y_6} & = & \boxed{x_6h_0} + \boxed{x_5h_1} + \boxed{x_4h_2} + \boxed{x_3h_3} + \boxed{x_2h_4} + \boxed{x_1h_5} \\
\boxed{y_7} & = & \boxed{x_7h_0} + \boxed{x_6h_1} + \boxed{x_5h_2} + \boxed{x_4h_3} + \boxed{x_3h_4} + \boxed{x_2h_5} \\
\boxed{y_8} & = & \boxed{x_8h_0} + \boxed{x_7h_1} + \boxed{x_6h_2} + \boxed{x_5h_3} + \boxed{x_4h_4} + \boxed{x_3h_5} \\
\vdots & & \vdots \\
\vdots & & \vdots \\
\vdots & & \vdots
\end{array}$$

Figure 1: An example of a 6-tap filter with $L=3$.

The scheme is implemented in two parts, on a 140 MHz Sun Ultra platform. The first part, incorporated in an C++ routine, considers data manipulation tasks in the scheme, such as fetching data and coefficients from their respective memory locations and feeding them into the multiplier inputs. This routine also monitors the number of memory accesses using data and coefficient buses. The second part considers the hardware implementation of the multiplier circuit. Here, a Verilog model of a two's complement array multiplier circuit is constructed with AND, OR, and XOR gates only. The circuit is then simulated, under different block size conditions, using the Verilog-XL simulator. For each simulation the number of signal transitions at the output of each gate is monitored. Then capacitive loading information is used to obtain the switched capacitance of each gate which is then accumulated to obtain the overall switched capacitance of the multiplier. The steps for the latter part could be summarised as follows:

1. Generate a layout of the multiplier circuit and extract capacitive information (wiring and loading). The Cadence 0.7 μm CMOS Design Kit is used for the implementation of this stage.
2. Obtain coefficients for a given filter specification and generate a set of uniformly distributed random data input samples.
3. Apply data samples, in blocks of size L , together with the corresponding coefficient to the inputs of the multiplier.
4. Simulate the multiplier using the Verilog-XL simulator and compute the switching activity factor, k_i , for each gate G_i . Together with the capacitive information of the gate, C_i obtained in (1), calculate the overall switched capacitance for the multiplier circuit as follows:

$$\sum_{i=1}^M k_i C_i$$

where M is the number of gates in the circuit.

5. Repeat steps (3) and (4) until all data samples generated in step (2) are processed.
6. Obtain the overall switched capacitance.

3. SIMULATIONS AND RESULTS

Ten filter coefficient sets were obtained by designing 10 practical filters (5 lowpass and 5 bandpass) with orders ranging from 32 to 89. Each filter was simulated 10 times. In each case a new set of data samples was generated and simulations were repeated for 1000 samples using 8, 16, and 24-bit two's complement array multipliers. In each simulation block sizes of 2, 4, 8, and 16 were used. As it could be seen from table 1, a reduction in the overall power consumption of the multiplier circuit is achieved in all cases with a maximum reduction of 33.90% for an 8-bit multiplier with a block size of 2. The table also indicates that the power reduction increases with the increase in block size up to a peak value which is equal to the power reduction achieved with a block size of 2. The reason for this power reduction profile could be explained with reference to table 2. The table shows that the switching activity at coefficient inputs decreases with the increase in block size, whereas the switching activity at data inputs decreases only for block size of 2. This reduction in switching activity at both data and coefficient inputs of the multiplier contributes to the overall reduction for block size of 2. Table 3 shows a reduction in the number of both data and coefficient memory accesses for block sizes of 2, 4, 8, and 16.

Block Size L	Multiplier Size		
	8-bit	16-bit	24-bit
	Reduction in power consumption (%)		
2	33.90	28.51	22.03
4	20.69	18.34	15.17
8	24.07	21.36	17.78
16	25.80	22.88	19.06

Table 1: Reduction in overall power consumption for multiplier.

L	Data inputs (%)	Coefficient inputs (%)
2	50.00	50.00
4	-	75.00
8	-	87.50
16	-	93.75

Table 2: Reduction in switching activity at multiplier inputs.

L	Data memory (%)	Coefficient memory (%)
2	48.43	50.00
4	72.65	75.00
8	84.76	87.50
16	90.82	93.75

Table 3: Reduction in number of memory accesses.

4. CONCLUSION

A *data block processing* scheme is proposed for the implementation of low power FIR filters on single multiplier DSPs. Our investigations revealed that the scheme achieves the following in the multiplier part of the DSP: (1) The switching activity at coefficient inputs of the multiplier circuit decreases as block size increases, whereas the switching activity at data inputs decreases only for a block size of 2. This results in a maximum power reduction with block size of 2. (2) Block sizes greater than 2 converge to the above maximum as block size increases. When the whole DSP is considered, however, data block processing reduces the number of data and coefficient memory accesses, by a factor depending on the block size. This leads to further power savings at both data and coefficient buses and their corresponding address buses.

5. REFERENCES

- [1] T. Arslan, A.T. Erdogan, and D.H. Horrocks, "Low Power Design for DSP, Methodologies and Techniques", *Microelectronics Journal*, vol.27, no.8, pp.731-744, November 1996.
- [2] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J.M. Rabaey, and R.W. Brodersen, "Optimizing Power Using Transformations", *IEEE Trans. CAD*, vol.14, no.1, pp.12-31, January 1995.
- [3] D. Singh, J.M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T.J. Mozdzen, "Power Conscious CAD Tools and Methodologies, A perspective", *Proc. IEEE*, vol.83, no.4, pp.570-593, April 1995.
- [4] K. Roy and S.C. Prasad, "Circuit Activity Based Logic Synthesis for Low Power Reliable Operations", *IEEE Trans. VLSI Systems*, vol.1, no.4, pp.503-513, December 1993.
- [5] M.R. Stan and W.P. Burleson, "Bus-invert Coding for Low-power I/O", *IEEE Trans. VLSI Systems*, vol.3, no.1, pp.49-58, March 1995.
- [6] M.D. Ercegovac and T. Lang, "Reducing Transition Counts in Arithmetic Circuits", 1994 IEEE Symposium on Low Power Electronics, San Diego, USA, pp.64-65, October 1994.
- [7] A.T. Erdogan and T. Arslan, "Low power multiplication scheme for FIR filter implementation on single multiplier CMOS DSP processors", *IEE Electronic Letters*, vol.32, no.21, pp.1959-60, 10th October 1996.
- [8] A.T. Erdogan, T. Arslan, and D.H. Horrocks, "Low power multiplication schemes for single multiplier CMOS based FIR digital filter implementations", *IEEE ISCAS'97*, Hong Kong, 9-12 June 1997, pp.1940-43.
- [9] S.K. Mitra and J.F. Kaiser, "Handbook for Digital Signal Processing", John Wiley & Sons, 1993.
- [10] T.G. Stockham, "High speed convolution and correlation", *Proc. of 1966 Spring Joint Comput. Conf.*, vol.28, Washington.
- [11] C.S. Burrus, "Block realization of digital filters", *IEEE Trans. Audio and Electroacoustics*, vol.AU-20, no.4, pp.230-235, October 1972.
- [12] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software, A first step towards software power minimization", *IEEE Trans. VLSI Systems*, vol.2, no.4, pp.437-445, December 1994.