VLSI DESIGN
1999, Vol. 10, No. 2, pp. 177–202
Reprints available directly from the publisher
Photocopying permitted by license only

Computation Reordering: A Novel Transformation for Low Power DSP Synthesis

K. MASSELOS*, P. MERAKOS, T. STOURAITIS and C. E. GOUTIS

VLSI Design Laboratory, Department of Electrical and Computer Engineering, University of Patras, Rio 26500, Greece

(Received 13 April 1998; In final form 21 July 1998)

A novel architectural transformation for low power synthesis of inner product computational structures is presented. The proposed transformation reorders the sequence of evaluation of the multiply-accumulate operations that form the inner products. Information related to both coefficients, which are statically determined, and data, which are dynamic, is used to drive the reordering of computation. The reordering of computation reduces the switching activity at the inputs of the computational units but inside them as well leading to power consumption reduction. Different classes of algorithms requiring inner product computation are identified and the problem of computation reordering is formulated for each of them. The target architecture to which the proposed transformation applies is based on a power optimal memory organization and is described in detail. Experimental results for several DSP algorithms show that the proposed transformation leads to significant savings in net switching activity and thus in power consumption.

Keywords: Low-power, switching activity, computation reordering, transformation, multiply-accumulate, digital signal processing

1. INTRODUCTION

The recent rapid advances in the areas of wireless communications and multimedia technology made available a large number of portable battery-operated systems such as cellular phones, pagers, wireless modems, portable videophones and hand-held digital video cameras. All these systems make extensive use of digital signal processing (either one or two-dimensional). Since power consumption is the overriding issue in the design of portable systems, low power digital signal processing became an increasingly important research area. One of the most common operations performed in digital signal processing applications is the inner product computation. An N-point inner product between an N-point vector of data $D = [d_0, d_1, \ldots, d_{N-1}]$ and an N-point vector of coefficients

*Corresponding author. Tel.: (+) 30 61 997324, Fax: (+) 30 61 994798, e-mail: masselos@ee.upatras.gr

$C = [c_0, c_1, \ldots, c_{N-1}]$ is described by the following equation:

$$\text{Inner Product} \quad D \times C = \sum_{i=0}^{N-1} d_i \times c_i \qquad (1)$$

There are many ways of realizing inner product structures in hardware as part of digital signal processing systems. One way is to use dedicated hardware multipliers and adders (pure custom hardware approach). A second alternative is the use of an instruction set processor (either general purpose processor or digital signal processor) and its computational units. Especially in instruction set digital signal processors the main computational unit is the multiply accumulator. Finally a multiply accumulator based custom hardware architecture allowing sharing of hardware units (adders and multipliers forming the multiply accumulators) and offering a low-level programmability and flexibility can be used. In all cases data and coefficients are stored in buffers either in background memories or in foreground memories.

The power consumption in digital CMOS circuits is ought to three sources [1], the dynamic (or switching), the short circuit, and the leakage power dissipation:

$$\begin{aligned} P_{\text{avg}} &= P_{\text{switching}} + P_{\text{shortcircuit}} + P_{\text{leakage}} \\ &= \alpha_{0\to1} C_L V_{DD}^2 f_{\text{clk}} \\ &\quad + I_{sc} V_{DD} + I_{\text{leakage}} V_{DD} \end{aligned} \qquad (2)$$

where $\alpha_{0\to1}$ is the node transition activity factor (the average number of times the node makes a power consuming transition in one clock period), C_L is the load capacitance, f_{clk} is the clock frequency and V_{DD} is the supply voltage. I_{sc} is the average short-circuit current and I_{leakage} is the average leakage current which depends on fabrication technology.

Optimization for power can be achieved in all the levels of the design flow. Various techniques have been proposed in the past to minimize the sources of power dissipation. Since the switching component is often the most significant source of power dissipation most of the proposed power optimization techniques aim at reducing the switching or dynamic component of power dissipation. Furthermore the switching component of power dissipation can be optimized in the high levels of abstrac- tion where the most significant power savings can be achieved [1].

A large number of data path synthesis techniques for power optimization have been proposed. Transformation based data path synthesis techniques for low power are of significant importance and have been addressed in [2–5]. In the area of low power digital signal processing several techniques for power optimization of FIR filters have been proposed. In [6, 7] the transformation of the binary representation of the filter coefficients is proposed while the technique described in [8] perturbs the filter coefficients to decrease the number of operations required to obtain the filter output while preserving the desired filter characteristics. In [9] a technique for low power realization of FIR filters using differential coefficients is presented while in [10] a transformation for power reduction, through minimization of the number of multiplications, in FIR filters is described. In [11, 12] low power architectures for the Discrete Cosine Transform (DCT) and the Discrete Fourier Transform (DFT) are presented respectively. General methodologies for low power digital signal processing are described in [13–16].

In this paper a novel architectural transformation for power consumption reduction in realizations of inner product computational structures present in digital signal processing algorithms is described. The main idea is the computation reordering aiming at reducing the switching activity at the inputs of the computational units. Information related to both data and coefficients is taken into account during the reordering. As a metric of the switching activity the hamming distance is used. Different cases are identified based on the computational structures of digital signal processing algorithms. Formulations of the computation reordering problem for the different classes of algorithms as well as the target architecture template are described.

The rest of the paper is organized as follows: In Section 2 an example illustrating the proposed computation reordering transformation is offered. The different classes of algorithms are presented in Section 3 while in Section 4 the target architecture template is described in detail. The proposed transformation is described in detail in Section 5. Experimental results for several digital signal processing algorithms are presented in Section 6. In Section 7 the generalization of the proposed methodology is described and finally in Section 8 conclusions are offered.

2. MOTIVATING EXAMPLE

Let *EXAMPLE_DSP* be a digital signal processing algorithm requiring evaluation of 4-point inner products between 4-point data and coefficient vectors. Let the coefficient vector $[c_0, c_1, c_2, c_3]$ be equal to $[13, 2, 9, 6]$ and assume that the data vector $[d_0, d_1, d_2, d_3]$ at a specific instance of the algorithm is equal to $[1, 3, 14, 2]$. *EXAMPLE_DSP* requires computation of 4-point inner products y using the following equation:

$$y = \sum_{i=0}^{3} p_i \tag{3}$$

where p_i denotes the partial product $c_i \times d_i$. The hamming distance between two partial products is defined as the sum of the hamming distances between the coefficients and between the data of the partial products. The hamming distances between all possible pairs of partial products for the specific instance of *EXAMPLE_DSP* are given in Table I.

TABLE I Hamming distances between all pairs of partial products

Partial product	P_0	P_1	P_2	P_3
P_0	0	5	5	5
P_1	5	0	6	2
P_2	5	6	0	6
P_3	5	2	6	0

If the computation of the inner product is performed as defined in the *EXAMPLE_DSP* on a single multiply accumulator the total hamming distance at the inputs of the multiply accumulator would be equal to 17. This case is illustrated in Figure 1.

If Table I is examined in detail the sequence of evaluation of the partial products that results in minimum switching activity at the inputs of the multiply accumulator can be derived. In fact the sequence of partial products P_2, P_0, P_1, P_3 (as well as the P_1, P_3, P_0, P_2) results in a total hamming distance at the inputs of the multiply accumulator equal to 12. This means that if data were known before implementation the partial products could be scheduled as described above leading to a switching activity reduction equal to 5. However in real life applications data are dynamic *i.e.*, not known before system realization. This dynamic behavior of data is a major problem in power estimation since power consumption is dependent on input data.

Based on the coefficient information which is available before realization, the sequence of evaluation of the partial products that minimizes the activity at the coefficient input of the multiply accumulator can be determined. The sequence of coefficients c_0, c_2, c_1, c_3 (as well as the sequence c_1, c_3, c_0, c_2) results in the minimum activity at the coefficient input of the multiply accumulator. The activity of the sequence of coefficients c_0, c_2, c_1, c_3 is 5 while the activity of the original sequence of coefficients is 11. If the inner product is computed following the sequence of evaluation of the partial products corresponding to the minimum activity sequence of coefficients the total activity at the inputs of the multiply accumulator equals to 13 *i.e.*, it is slightly worse than the optimal (12-achieved when also data information were used to schedule the evaluation of partial products). This case is illustrated in Figure 2.

Although data cannot be known before algorithm's run time typical input data may be known before realization. Assuming that typical input data of the *EXAMPLE_DSP* algorithm are available,

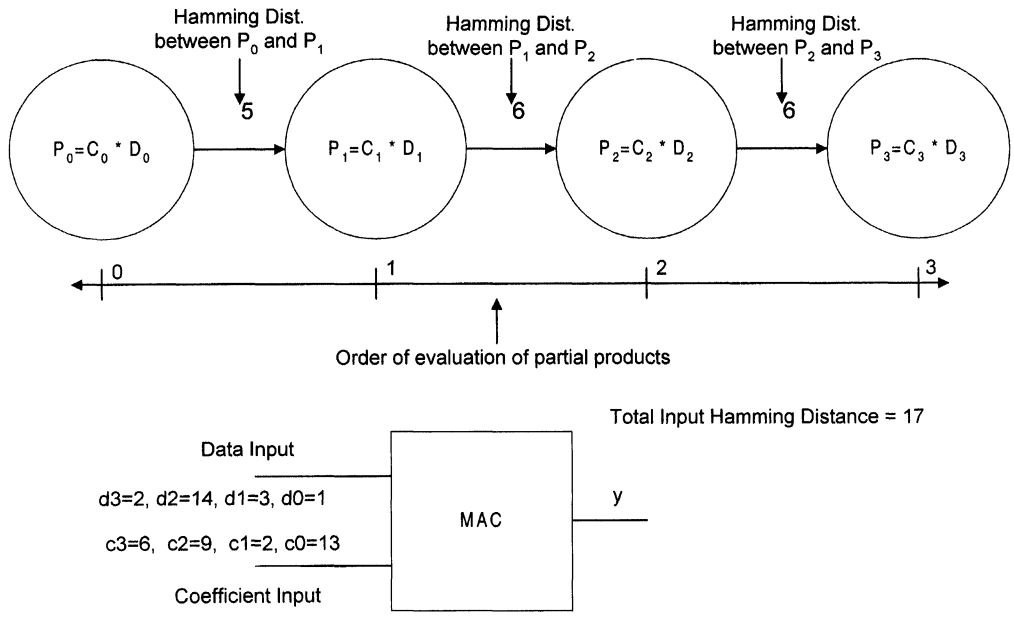

FIGURE 1 Computation of the 4 point inner product in the original case.

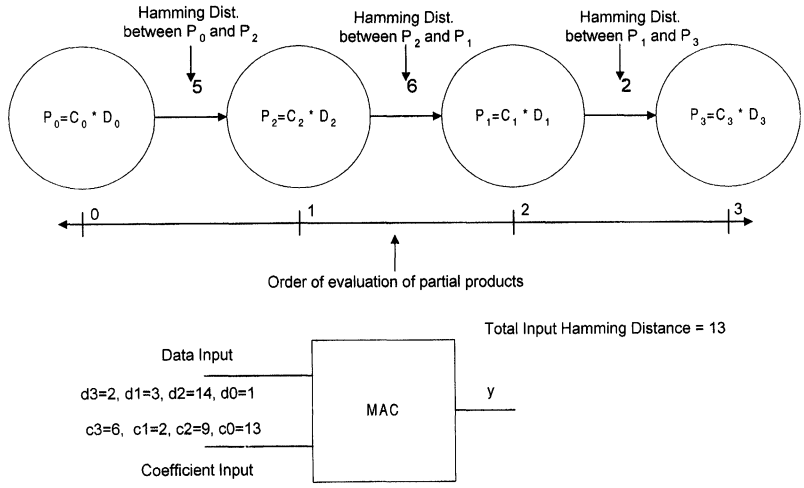

FIGURE 2 Computation of the 4 point inner product after reordering of computation wrt coefficients.

simulation of these data can be used to approximate the hamming distances between all the pairs of data elements of the 4 point data vectors. Table II describes the result of the statistical analysis of the typical input data of the *EXAMPLE_DSP* where average hamming distances for all pairs of data in the 4-point vector are given.

TABLE II Average hamming distances between all pairs of data after simulation

Data	$D_0 = 1$	$D_1 = 3$	$D_2 = 14$	$D_3 = 2$
$D_0 = 1$	0	1.2	4.5	2.6
$D_1 = 3$	1.2	0	2.8	1.5
$D_2 = 14$	4.5	2.8	0	2.3
$D_3 = 2$	2.6	1.5	2.3	0

Combining this data related information with the already available coefficient information, the hamming distances for all pairs of partial products can be evaluated. This information is included in Table III.

TABLE III Hamming distances between all pairs of partial products after statistical analysis

Partial product	P_0	P_1	P_2	P_3
P_0	0	5.2	5.5	5.6
P_1	5.2	0	5.8	2.5
P_2	5.5	5.8	0	6.3
P_3	5.6	2.5	6.3	0

Based on the values of Table III the minimum activity sequence of evaluation of the partial products is the P_2, P_0, P_1, P_3 with an activity value equal to 13.2. The real activity of this sequence is 12 *i.e.*, the real minimal activity. In the last case where the simulation determined average hamming distance of the pairs of data is taken into account the hamming distances between pairs of partial products (as well as between pairs of data) are real numbers. This case is illustrated in Figure 3.

The results of the reordering of the evaluation of the partial products are summarized in Table IV and prove the significant savings in switching activity at the inputs of the computational units that can

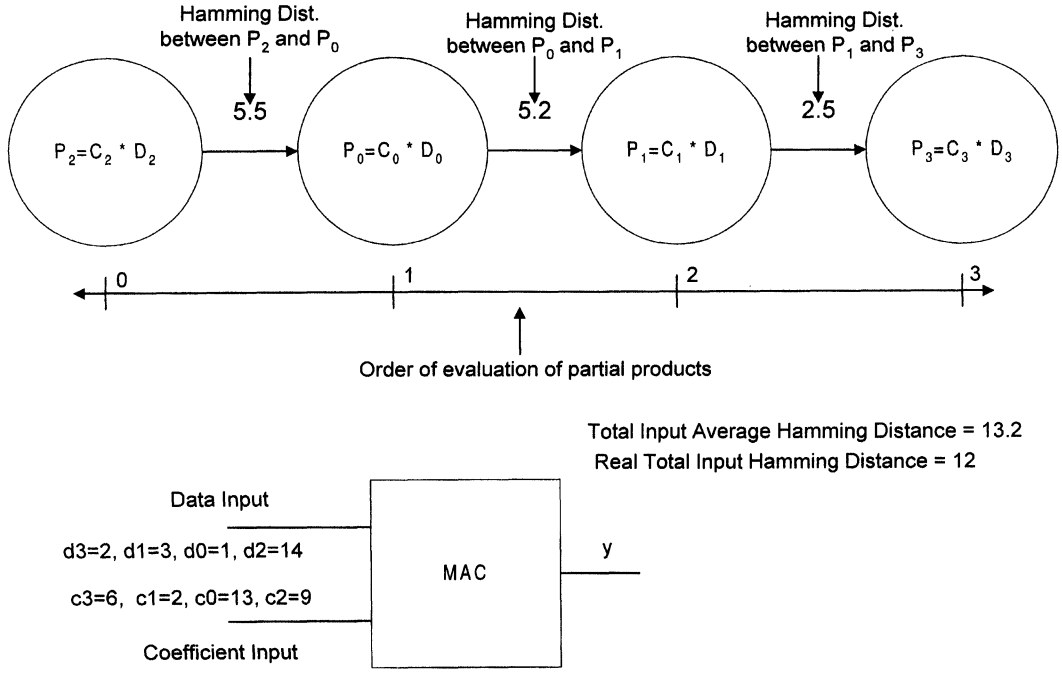

FIGURE 3 Computation of the 4 point inner product after reordering of computation wrt both coefficients and results of the simulation of typical data.

TABLE IV Results of the reordering of the evaluation of the partial products

Order of evaluation of partial products	Activity at the coefficient input	Total activity at the inputs	Savings %
Original	11	17	
Reordered wrt both coefficients and real data (not realistic)	6	12 (minimal)	29.4
Reordered wrt coefficients	5	13	23.5
Reordered wrt both coefficients and statistical data information	6	12 (real) 13.2 (statistical)	29.4

be obtained by the proposed transformation *i.e.*, by changing the schedule of partial products evaluation. This reduction of the input switching activity results to switching activity and power consumption reduction within the computational units as well. It must be noted that the use of the simulation determined data related information does not always guarantee a global optimal solution of the reordering problem.

3. CLASSES OF DSP ALGORITHMS REQUIRING INNER PRODUCT COMPUTATION

Two basic classes of digital signal processing algorithms that require inner product computations have been identified:

(a) First class: Convolution type algorithms (FIR filtering, Wavelets).
(b) Second class: Transformation type algorithms (DCT, DFT, FFT).

3.1. First Class of Algorithms

The main type of computation in the algorithms of this class is the convolution between data and coefficient vectors. A typical example belonging to this category is the Finite Impulse Response (FIR) filtering which is one of the most common DSP applications. An N-tap FIR filter performs the following convolution:

$$Y_n = \sum_{i=0}^{N-1} C_i X_{n-i} \tag{4}$$

where C_i's are the coefficients of the filter (forming an N-point coefficient vector) and X_n, Y_n are the nth terms of the input and output sequences respectively. From the above equation it is obvious that the evaluation of one point of the output requires computation of an N-point inner product between data and coefficient vectors. The main characteristic of the computation performed by the algorithms of this class is that for the evaluation of the output terms the same coefficient vector and different data vectors (overlapping by $N-1$ terms and one different term for successive output terms) are used. The computation required for one output point (as described by Eq. (4)) is defined as the basic computation for this class of algorithms. Another algorithm belonging to this class is the Discrete Wavelet Transform (forward and inverse). The main task of the Discrete Wavelet Transform is the FIR filtering.

3.2. Second Class of Algorithms

The main type of computation in the algorithms of this class is the matrix-vector multiplication between a coefficient matrix and a data vector. Typical examples of algorithms belonging to this category are the common digital signal processing transformations like the Discrete Cosine Transform (DCT), the Discrete Fourier Transform (DFT), and the Fast Fourier Transform (FFT). An $N \times M$ transformation performs the following computation:

$$Y = CX \Leftrightarrow \begin{bmatrix} Y_0 \\ Y_1 \\ \\ Y_{N-1} \end{bmatrix} = \begin{bmatrix} C_{00}, & C_{01}, \ldots, C_{0M-1} \\ C_{10}, & C_{11}, \ldots, C_{1M-1} \\ & \\ C_{(N-1)0}, & C_{(N-1)1}, \ldots, C_{(N-1)(M-1)} \end{bmatrix} \times \begin{bmatrix} X_0 \\ X_1 \\ \\ X_{M-1} \end{bmatrix} \tag{5}$$

where Y is the N-point output data (Y_i's) vector, C is the $N \times M$ coefficient (C_{ij}'s) matrix, and X is the M-point input data (X_i's) vector. Evaluation of each term Y_i of the output column vector requires computation of an M-point inner product between the ith row of the coefficient matrix and the input data vector. It must be noted that the above-described computation is present in the one-dimensional versions of the transformations. Since the two dimensional versions of the transformations are computed using the row-column decomposition it can be said that in the two dimensional versions of the algorithms the main computational structure is the one described in Eq. (5). The main characteristic of the basic computational structure

of the algorithms of the second class is that the coefficient vector used for the computation of the output points is not always the same. The number of different sets of coefficients is equal to N *i.e.*, the first transformation dimension. On the other hand the same data vector is used for N output points of the transformation. The computation required for N output points as described by Eq. (5) is defined as the basic computation for the second class of algorithms.

4. TARGET ARCHITECTURE

The structure of the proposed architecture is shown in Figure 4. The application data are stored in a background memory (either on-chip or off-chip). This memory may be quite large especially for multidimensional signal processing applications like image and video processing [17]. For both classes of algorithms a set of foreground registers is used to store the data terms required for each inner product computation. This memory hierarchy is introduced to exploit the data reuse present in both classes of algorithms and reduce the memory related power consumption [18]. This extra foreground memory does not introduce a significant penalty since in submicron technologies the area and power costs of a register are very small.

For the algorithms of the first class computation of N-point convolutions requires N accesses (reads) to the background data memory for each convolution output if no hierarchy is introduced. However for the computation of two successive output terms of the convolution, $N-1$ common input data terms and only one different are required (overlapping by $N-1$). If a set of N foreground registers is used to store the input data terms for each convolution the number of background memory accesses is heavily reduced. Specifically only one read from the background memory and $N+1$ accesses (N reads and 1 write) to the foreground registers (which are far less power expensive than background memory accesses) are required for each convolution computation. After the $n-1$th output term of the convolution is computed the data term in each register is shifted to the previous register and the new data term is transferred from the background memory to the first register (register[n]) and the computation of the nth output term of the convolution may start. An algorithm of the second class requiring computation of an $N \times M$ transformation accesses the background memory $N \times M$ times if no memory hierarchy is introduced. Introduction of M registers to which the M input data terms required for the computation of an output vector are transferred in a first step reduces the number of the background memory transfers significantly. The presence of registers leads to M accesses to the background memory and $N \times M + M$ accesses to the foreground registers for the computation of an N-point output vector.

It is straightforward that the register based foreground memory organization leads to significant power savings and is also beneficial even if no computation reordering is performed (implementation based on the original specification). Since the research under consideration aims at power consumption reduction the above described architecture model is adopted and will be used for the comparison of implementations based on the original computation ordering and implementations based on the derived computation ordering.

The proposed computation reordering transformation does not introduce any addressing penalty although the order in which the data terms are transferred to the inputs of the functional units changes. The data are transferred to the foreground registers (either from the background memory or directly from the inputs of the circuit) in the same order as in the original case but read from them according to the new order of computation. The foreground storage elements (registers or register files) are addressed directly by the control unit. Thus no address generation overhead is introduced by the reordering of computation. As far as the activity of the address lines (in case of register files) is concerned, activity reduction techniques like Gray encoding can be used as in the case where no computation reordering is performed.

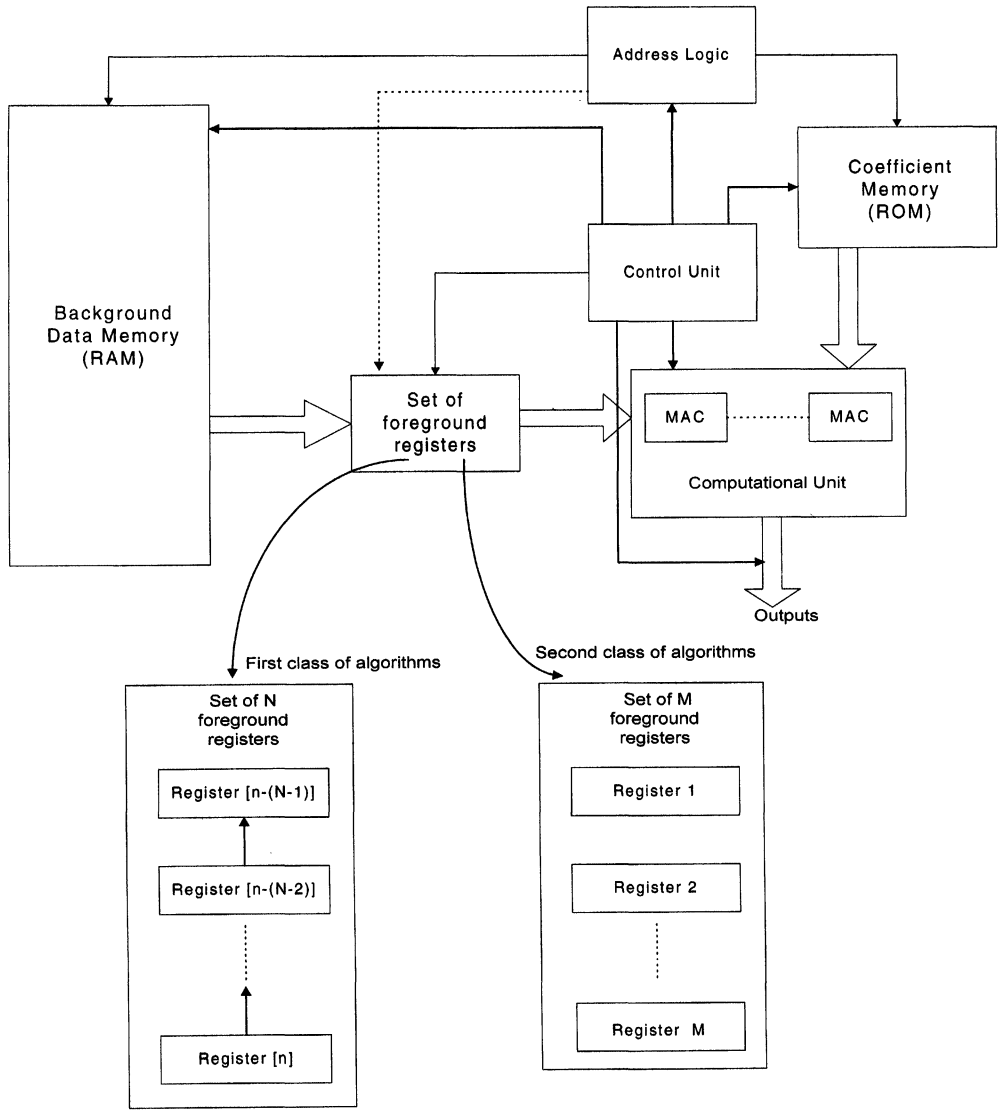

FIGURE 4 Target architecture model.

The coefficients are stored in a coefficient memory (usually a ROM). In some cases coefficients are stored in the same memory block as the data. The computation of the output terms of the convolution is performed on a multiply accumulator based computational unit. Multiply accumulator modules, generated as complete functional blocks, may be used to achieve a higher operation speed than computational units consisting from a specific number of adders and multipliers [19]. A multiply accumulator based custom hardware architecture for the implementation of the discrete wavelet transform has been proposed in [20]. The multiply accumulators can be either pipelined or not pipelined. The number of multiply accumulators (number of hardware resources) used is in general determined by the performance requirements of the application. The number of register sets required depends on the number of basic computations performed in parallel (number of

different output terms computed in parallel). One register set is required for each of the basic computations performed in parallel.

5. THE PROPOSED TRANSFORMATION

5.1. General

The proposed transformation reorders the sequence of evaluation of the partial products $c_i d_i$ that constitute an inner product. The criterion for the reordering of the computation is the minimization of the sum of hamming distances between successive pairs of partial products of the form $c_i d_i$ and $c_{i+1} d_{i+1}$. The hamming distance between two partial products $c_i d_i$ and $c_j d_j$ is defined as:

$$\mathrm{HD}(c_i d_i, c_j d_j) = \mathrm{HD}(C_i, c_j) + \mathrm{HD}(d_i, d_j) \qquad (6)$$

where HD(a,b) is the hamming distance of the binary representations of numbers a, b. Minimization of the sum of hamming distances between successive pairs of partial products results in switching activity reduction at the inputs of the computational units that perform the inner product computation. The reduction of the switching activity at the inputs of the computational units leads to switching activity and power consumption reduction inside them as well. To simplify analysis, in this section it is assumed that the computation is performed on a single multiply accumulator.

The evaluation of the hamming distance between coefficients is straightforward since coefficients are statically determined in digital signal processing algorithms *i.e.*, known before realization (hardware implementation or software compilation). On the other hand the evaluation of the hamming distance between two samples of data (belonging to the data vector of an inner product) is quite hard since data are dynamic *i.e.*, not known before run-time. An estimate of the hamming distance between two data samples can be obtained by simulation of typical data of a specific application. Consider a digital signal processing algorithm that requires computation of N point inner products. The simulation can be performed in the following way: For every possible pair of samples d_i, d_j, that belong to the inner product data vector D, the Average Hamming Distance (AHD_{ij}) is evaluated. A sequence of typical application data can be used for simulation. From this sequence the inner product data vectors are extracted (according to the algorithm) and the average hamming distances for all possible pairs of data that belong to the data vector are computed. The simulation stops when the following stopping criterion is satisfied:

$$\frac{\mathrm{AHD}_{ij}(M) - \mathrm{AHD}_{ij}(M+1)}{\mathrm{AHD}_{ij}(M)} < \text{Threshold} \quad \text{for all } i,j \qquad (7)$$

Where $\mathrm{AHD}_{ij}(M)$ is the average hamming distance between the ith and jth data elements of the inner product data vector after M simulation cycles (*i.e.*, different sets of values for the inner product data vector).

In this way information related to both data and coefficients can be used to drive the reordering of computation. However only the static information (*i.e.*, the values of the coefficients) can also be used to determine the final schedule of the computation of the partial products. In such a case the optimality (in terms of switching activity reduction) of the derived schedule is traded-off for an increased speed of the reordering procedure.

5.2. First Class of Algorithms-mathematical Model and Problem Formulation

As already stated the basic task performed by the algorithms of the first class is the convolution between data and coefficients. An N-point convolution requires evaluation of an N-point inner product for each term of the output sequence. To simplify the presentation of the proposed transformation it is assumed that the basic computation for the algorithms of this class (as defined in the previous section) is performed on a single multiply

accumulator. The convolution operation, for the evaluation of an output term, performed by the algorithms of this class is described by the following equation:

$$y_n = \sum_{k=0}^{N-1} C_{f(k)} X_{n-f(k)} \tag{8}$$

where y_n is the nth term of the output sequence, C is the coefficient vector and X is the vector of the input data. The computation is performed according to the ordering function $f(k) = k$, $k = 0, \ldots, N-1$. The total hamming distance of the sequence of evaluation of the partial products as determined by the ordering function $f(k)$ is given by the following equation:

$$\text{Total HD}\,(f(k)) = \sum_{f(k)=0}^{N-1} \text{HD}\,(p_{f(k)}, p_{f(k)+1}) + \text{HD}\,(p_0, p_{N-1}) \tag{9}$$

where $p_{f(k)}$ is the partial product $c_{f(k)} x_{n-f(k)}$. The hamming distance for a pair of partial products $\text{HD}(p_k, p_l)$ is given as

$$\text{HD}(p_k, p_l) = \text{HD}(c_k, c_l) + \text{AHD}(x_{n-k}, x_{n-l}) \tag{10}$$

The average hamming distance (AHD) between the two data samples is determined after simulation as described above. In Eq. (9) the term $\text{HD}(p_0, p_{N-1})$ denotes the hamming distance between the first and the last partial product (according to the ordering function $f(k)$). This term is included because after the computation of the last partial product (p_{N-1}) for the output term y_n the first partial product (p_0) for the term y_{n+1} must be computed.

The aim of the proposed methodology is to derive a new ordering function $g(k)$, $k = 0, \ldots, N-1$, such that the total hamming distance of the convolution computation as determined by $g(k)$ (Total HD($g(k)$)), given by Eq. (9) when $f(k)$ is replaced by $g(k)$, is minimal and the inner product value is computed according to the equation

$$y_n = \sum_{k=0}^{N-1} C_{g(k)} X_{n-g(k)} \tag{11}$$

Since there are no data dependencies between partial products and taking into consideration that the basic computation is performed on a single multiply-accumulator, the problem of finding the minimum cost (in terms of total input switching activity) ordering function is equivalent to finding the minimum cost schedule of partial products on the multiply accumulator. The cost function driving the reordering of the computation *i.e.*, the derivation of the new ordering function $g_{(k)}$ may include only the static information (related to coefficients) to speedup the reordering procedure.

The problem of computation reordering for the first class of algorithms is formulated as a Travelling Salesman Problem (TSP). The graph $G\,(V, E)$ of the problem consists of the set V of the N vertices, which are the partial products required for the computation of an inner product, and the set E of edges which model the unconstrained transition from one partial product to another. The problem's graph is complete *i.e.*, each vertex pair is connected by an edge. To each edge of the graph a weight, which is the HD between the two partial products that the edge connects as defined by Eq. (10) for a specific ordering function, is assigned. A closed path over all vertices (partial products) must be found, without passing from a vertex more than once, resulting in a minimum HD cost. The weights of the edges between the vertices are real positive values. Their lower bound is zero and their higher bound equals to the sum of the number of bits used for the representation of the coefficients and the representation of data terms. If the simpler cost function (including only the static information related to coefficients) is used the costs of the edges are well-bounded positive integer numbers. Their lower bound is zero and the higher bound equals to the number of bits used for the representation of the coefficients.

Several algorithms have been proposed for the solution of the TSP problem. If the size of the problem is relatively small, an exact solution can

be found in short time. A large number of exact algorithms have been proposed for TSP which can be best understood and explained in the context of Integer Linear Programming (ILP) [21]. The NP-complete class of the problem motivated the research for heuristic algorithms. Christofides in [22] proposed a heuristic using a minimum-cost matching algorithm and requiring $O(n^4)$ time to find a nearly optimal solution.

5.3. Second Class of Algorithms-mathematical Model and Problem Formulation

Two different cases are identified for the algorithms of the second class.

Case 1 In a high level description (for example a *C* description) of a digital signal processing algorithm the computation described in Eq. (5) is typically expressed as described in Figure 5:

This piece of code computes sequentially (one after the other) the output points of an algorithm of the second class by multiplying a row of the coefficient matrix with the input data vector.

The computation of the output points of a transformation of length $N \times M$ is described by the following equation

$$y_i = \sum_{j=0}^{M-1} c_{ij} x_j, \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{12}$$

where y_i is the ith output point, c_{ij} are the M coefficients of the ith row of the coefficient matrix and x_j are the M elements of the input data vector. Evaluation of one output term of an $N \times M$ transformation requires computation of an M-point inner product. To simplify analysis it will be assumed that the basic computation, as defined in Section 3.2, is performed on a single multiply accumulator. Since the coefficient structure is two-dimensional two ordering functions are required to describe the matrix-vector computation of the second class of algorithms. Using ordering functions the computation described in Eq. (12) can be expressed as

$$y_{f(i)} = \sum_{g(i,j)=0}^{M-1} c_{f(i),g(i,j)} x_{g(i,j)}, \quad \text{for} \quad f(i) = 0, 1, \ldots, N-1 \tag{13}$$

The computation is performed according to the ordering functions $f(i) = i$, $g(i,j) = j$, $i = 0, 1, 2, \ldots, N-1, j = 0, 1, 2, \ldots, M-1$. The $f(i)$ ordering function corresponds to the rows of the coefficient matrix and the $g(i,j)$ ordering functions (one per row of the coefficient matrix) correspond to the columns of the coefficient matrix. The $f(i)$ function determines the order in which the inner products are computed (order in which output terms are evaluated) while $g(i,j)$ function determines the order in which the partial products that form an inner product are computed and it is different for each inner product. The total hamming distance of the sequence of evaluation of the partial products that constitute the N inner products as determined by the ordering functions $f(i)$, $g(i,j)$, is given by the following equation:

$$\text{Total HD}(f,g) = \sum_{f(i)=0}^{N-1} \sum_{g(i,j)=0}^{M-1} \text{HD}(p_{f(i),g(i,j)}, p_{f(i),g(i,j)+1}) + \text{HD}(p_{f(i),N-1}, p_{f(i)+1,0}) \tag{14}$$

```
for(i=0;i<N;i++)
  for(j=0;j<M;j++)
   output_vector[i]+=coefficient_matrix[i][j]*input_vector[j];
```

FIGURE 5 Typical piece of code for Case 1.

where $p_{f(i),g(i,j)}$ is the partial product $c_{f(i)g(i,j)}x_{g(i,j)}$. The hamming distance for a pair of partial products is given as

$$\mathrm{HD}(p_{f(i)g(i,j)}, p_{f(k)g(k,l)}) = \mathrm{HD}(c_{f(i)g(i,j)}, c_{f(k)g(k,l)}) + \mathrm{AHD}(x_{g(i,j)}, x_{g(k,l)}) \tag{15}$$

The average hamming distance (AHD) between two consecutive data samples is determined after simulation as earlier. In Eq. (14) the term $\mathrm{HD}(p_{f(i),N-1}, p_{f(i)+1,0})$ denotes the hamming distance between the last partial product of the inner product $f(i)$ and the first partial product of the inner product $f(i)+1$. This term is included because after the computation of the last partial product ($p_{f(i),N-1}$) for the output term $y_{f(i)}$ the first partial product ($p_{f(i)+1,0}$) for the term $y_{f(i)+1}$ must be computed.

The aim of the proposed methodology is to derive new ordering functions $p(i)$, $r(i,j)$, such that the total hamming distance of a matrix-vector product computation as described in Eq. (14) (when f, g are replaced by p, r) is minimized and the inner product value is computed according to the equation

$$y_{p(i)} = \sum_{r(i,j)=0}^{M-1} c_{p(i)r(i,j)}\, x_{r(i,j)}, \quad \text{for} \quad p(i) = 0, 1, \ldots, N-1 \tag{16}$$

The cost function driving the reordering of the computation *i.e.*, the derivation of the new ordering functions p, r, may again include only the static information (related to coefficients) to speedup the reordering procedure.

No data dependencies exist among inner products as well as among partial products forming an inner product. Furthermore it is assumed that the basic computation is performed on a single multiply accumulator. Thus the problem of deriving a minimum switching cost ordering function for the inner products is equivalent to deriving a minimum switching cost schedule for the inner products on the multiply accumulator. The problem of deriving minimum cost ordering functions for the evaluation of the partial products forming the inner products is equivalent to deriving minimum switching cost schedules for the partial products of each inner product on the multiply accumulator.

The derivation of new ordering functions will lead to a non-sequential order of appearance of the output points at the outputs of the computational units. However this is the usual case in all the fast transformation algorithms which usually require a kind of post-processing.

The formulation of the computation reordering problem is harder for the algorithms of the second class. This is because the problem is two-dimensional in this case. In a first step the minimum cost ordering of the inner products IP_i must be determined. In the second step the minimum cost ordering of the partial products p_{ij} ($j = 0, 1, \ldots, M-1$) that constitute each inner product, must be found. The problem is tackled in the following way: For all possible pairs of inner products IP_i, IP_j, ($i = 0, 1, \ldots, N-1$, $j = 0, 1, \ldots, M-1$) the minimum cost (switching activity) connection is determined. The minimum cost connection is defined as the pair of partial products p_{ik} (belonging to the inner product $\mathrm{IP}_i, k = 0, 1, \ldots, N-1$) and p_{jl} (belonging to the inner product IP_j, $l = 0, 1, \ldots, N-1$) that minimizes the hamming distance between all possible pairs of partial products as described by Eq. (17). The complexity of this procedure is $M^2 N (N-1)$.

$$\begin{aligned} &\text{Minimum cost connection for } \mathrm{IP}_i\ \mathrm{IP}_j : (p_{ik}, p_{jl}) \\ &\quad i,j = 0, 1, \ldots, N-1 \quad k,l = 0, 1, \ldots, M-1 \\ &\qquad \mathrm{HD}(p_{ik}, p_{jl}) \le \mathrm{HD}(p_{im}, p_{jn}) \\ &\quad \text{for all } m,n, \quad m,n = 0, 1, \ldots, M-1 \end{aligned} \tag{17}$$

Since the minimum cost connection has been determined for all possible pairs of inner products the graph $G\ (V, E)$ of the problem can be constructed where V is the set of N vertices while E is the set of edges. The graph is complete and the N vertices correspond to the N inner products while edges model the unconstrained transitions

from one inner product to another. To each edge the minimum cost connection (determined previously) corresponding to the inner products (vertices) connected by the edge is assigned as a cost. The determination of a minimum cost ordering of the inner products is formulated as a Travelling Salesman Problem (TSP) on this graph. A closed path over all vertices (inner products) must be found, without passing from a node more than once, resulting in a minimum HD cost. For the solution of the TSP the strategies proposed for the algorithms of the first class can be used. As soon as the minimum switching ordering of the inner products is determined, the partial products that constitute each inner product must be ordered to minimize the switching required for its computation. In fact each vertex (inner product) of the problem's graph is a graph with vertices corresponding to the partial products that constitute the inner product represented by the initial vertex. The hamming distances between the partial products are assigned as costs to the edges of the inner product graphs. However the ordering of the inner products performed in the first step results in the determination of the partial products that will be evaluated first and last for each inner product. Thus the derivation of the minimum cost ordering of the partial products of an inner product can be formulated as a restricted minimal spanning tree problem. This problem is a minimal spanning tree since only an open path over the vertices of the inner product graph (partial products) must be found. The problem is restricted since the starting and ending vertex of the path is determined by the inner product ordering (first step). An exact solution can be found if the number of the vertices of the inner product graph is relatively small. Two widely known algorithms for the solution of the minimal spanning trees problem are the algorithms of Kruskal [23] and Prim [23]. In all cases the costs of the edges are well-bounded positive real numbers. Their lower bound is zero and the higher bound equals to the sum of the number of bits used for the representation of the coefficients and the representation of data terms. If the simpler cost function is used the costs of the edges are well-bounded positive integer numbers. Their lower bound is zero and the higher bound equals to the number of bits used for the representation of the coefficients.

Case 2 This case is described by the piece of *C* code illustrated in Figure 6. This piece of code was derived by applying a loop interchange [24] to the nested *for* statement of Case 1. The loop interchange transformation increases the locality of data access [24]. The locality of data access favours reduction of power consumption [25] since it reduces the transfers of data from background memories to computational units through highly capacitive global buses.

According to the piece of code described in Figure 6 the N output points of the transformation are computed simultaneously. Only one partial product $c_{ij}x_j$ for each output point is computed and accumulated in each iteration of j. For simplification it is assumed that the basic computation is performed on a single multiply accumulator. An implementation corresponding to the description of Case 2 requires only M accesses to the background memory (one access per input data term X_j) as a result of the increased locality of access created by the loop transformation. The penalty of an implementation based on the Case 2 description is the introduction of the N registers to accumulate the partial products of each output

```
for(j=0;j<M;j++)
 for(i=0;i<N;i++)
  output_vector[i]+=coefficient_matrix[i][j]*input_vector[j];
```

FIGURE 6 Typical piece of code for Case 2.

term as well as a number of accesses to these registers. It is straightforward that the loop interchange and the introduction of the foreground registers leads to significant power consumption reduction by reducing the number of background memory accesses.

The computation performed by the algorithms of class 2 if the Case 2 description is adopted for a transformation of length N is described by the following equation

$$\begin{aligned} &\text{for} \quad j = 0, 1, \ldots, M-1 \\ &\quad \text{for all output points } y_i, y_i = c_{ij} x_j, \end{aligned} \tag{18}$$

where y_i is the ith output point, c_{ij} is the coefficient of the ith row and the jth column of the coefficient matrix and x_j is the jth element of the input data vector. The data terms are assumed to be stored in successive memory locations and to be accessed sequentially. Evaluation of the partial products corresponding to the jth input data element for all the output terms of an N-point transformation requires computation of N partial products. For the complete computation of the output terms of the transformation computation of $N \times M$ partial products is required. Using ordering functions the computation described in Eq. (18) can be expressed as

$$\text{for} \quad j = 0, 1, \ldots, M-1 \quad \text{for all output points } y_{f(i,j)}, \quad y_{f(i,j)} = c_{f(i,j),g(j)} x_{g(j)} \tag{19}$$

The computation is performed according to the ordering functions $f(i,j) = i$, $g(j) = j$, $i, j = 0, 1, 2, \ldots, N-1$. The f ordering functions correspond to the rows of the coefficient matrix (one per column of the coefficient matrix) and the g ordering function corresponds to the columns of the coefficient matrix. The f functions determine the order in which the partial products (using a specific input data term) of the output terms are computed for each column (order in which output terms are evaluated). The g function determines the order in which the input data are read from the memory.

The total hamming distance of the sequence of evaluation of the $N \times M$ partial products required for the N output terms as determined by the ordering functions $f(i,j)$, $g(j)$, is given by the following equation:

$$\begin{aligned} \text{Total HD} &= \sum_{g(j)=0}^{M-1} \sum_{f(i,j)=0}^{N-1} \mathrm{HD}(p_{f(i,j),g(j)}, p_{f(i,j)+1,g(j)}) \\ &\quad + \mathrm{HD}(p_{N-1,g(j)}, p_{0,g(j)+1}) \end{aligned} \tag{20}$$

where $p_{f(i,j),g(j)}$ is the partial product $c_{f(i,j)g(j)} x_{g(j)}$. The hamming distance for a pair of partial products is given as

$$\begin{aligned} &\mathrm{HD}(p_{f(i,j),g(j)}, p_{f(k,l),g(l)}) \\ &\quad = \mathrm{HD}(c_{f(i,j),g(j)}, c_{f(k,l),g(l)}) \end{aligned} \tag{21}$$

Data related activity information (AHD between two consecutive data samples) is not included since the data term remain the same for N consecutive partial products (j = constant for $i = 0, 1, \ldots, N-1$ and partial products corresponding to a specific data term for all the output terms are computed). In Eq. (20) the term $\mathrm{HD}(p_{N-1,g(j)}, p_{0,g(j)+1})$ denotes the hamming distance between the last partial product of the column $g(j)$ and the first partial product of the column $g(j)+1$. This term is included because after the computation of the last partial product ($p_{N-1,g(j)}$) for the output term y_{N-1} the first partial product ($p_{0,g(j)+1}$) for the term y_0 must be computed. The sequence in which the data terms are read from the memory and the corresponding partial products are computed is determined by the fact that the data terms are accessed sequentially from the background memory. Under this assumption it can be said that the $g(j)$ ordering function is always equal to j and no reordering with respect to the input data can be performed.

The aim of the proposed methodology is to derive new ordering functions $p(i,j)$ for each of the columns of the coefficient matrix, such that the total hamming distance of a case 2 matrix-vector

product computation as described in Eq. (20) is minimized (when $f(i,j)$ is replaced by $p(i,j)$) and the inner product value is computed according to the equation

$$\begin{aligned} &\text{for } j = 0, 1, \ldots, M-1 \\ &\qquad \text{for all output points} \\ &\qquad y_{p(i,j)}, y_{p(i,j)} = c_{p(i,j),j} x_j \end{aligned} \tag{22}$$

No data dependencies exist among partial products using the same input data term. Furthermore it is assumed that the basic computation is performed on a single multiply accumulator. The problem of deriving minimum cost ordering functions for the evaluation of the partial products is equivalent to deriving minimum switching cost schedules for the partial products using the same input data term on the multiply accumulator.

The formulation of the computation reordering problem in this case is similar to that of Case 1 formulation. The inner products of Case 1 are replaced by sets of partial products using the same input data term. The order in which the sets of partial products are evaluated is determined by j (under the assumption of sequential background memory accesses). The costs assigned to the edges of the graphs corresponding to the sets of partial products are well-bounded positive integers and not real numbers. This is because no data related information is included this time. Their lower bound is zero and the higher bound equals to the sum of the number of bits used for the representation of the coefficients.

5.4. Analysis of Power Savings

In this section the effect of the proposed transformation on the power consumption of the implementations of digital processing algorithms is summarized.

The proposed transformation derives an activity optimal schedule for the evaluation of the partial products, that form the inner products required for the evaluation of the output terms, is derived. In this way the switching activity at the inputs of the computational units is significantly reduced. The switching activity of a circuit depends on the internal node structure and on the switching at the inputs. In the high levels of abstraction the detailed internal architecture (gate level implementation) of the computational units (adders, multipliers) is not known and the units are treated as black boxes. It can be safely assumed that the switching activity inside the computational units and thus the power consumption is proportional to the input switching activity. In general in the high levels of the design abstraction the power consumption is estimated based on the total input switching activities [26]. Thus the computation reordering transformation leads to important power reduction in the computational units.

The proposed transformation achieves reduction of the switching activity in the coefficient input of the computational units. This leads to reduction of the switching activity in the bus connecting the coefficient memory and the computational units (very important since bus capacitance is high leading to important power consumption in them) and to the coefficient memory periphery unit as well. The savings in these circuits are equal to the savings achieved in the coefficient inputs of the computational units.

6. EXPERIMENTAL RESULTS AND ANALYSIS

The proposed transformation was applied to several digital signal and image processing algorithms belonging to the classes described in Section 3. Word parallel implementations on multiply-accumulator based data paths were generated for the algorithms. Two's complement fixed point arithmetic was assumed for all cases. Real application data were used for determining the average hamming distances between data terms of the input vectors and for simulation. The results for the different algorithms are described in the following sub-sections. The power consumption reduction is approximated by the reduction of the total input

switching activity and of the internal switching activity of the computational units.

6.1. FIR Filters

The results of the application of the proposed methodology to several FIR filters are included in Table V.

In the second column the type of data used for the simulation is indicated. In the third and fourth columns the changes in switching activities (after the application of the proposed transformation) in the coefficient and data inputs of the computational units are given respectively. In the fifth column the total change at the inputs of the computational units is given and finally column six gives the switching activity changes inside the computational units. The reduction of switching activity inside the computational units is directly proportional to power consumption reduction since the other factors affecting power consumption (supply voltage, frequency of operation, and capacitive load) do not change by the application of the proposed methodology. For all cases − denotes savings in switching activities resulted by the application of the proposed transformation, while + denotes penalties introduced by the proposed transformation.

The first two filters (63 and 14 taps) are typical band pass filters and were simulated using random (Ran.) data. The coefficients were represented using 16 bits (15 bits dynamic range) while 12 bits represented data. For the outputs 16 bits were used. Because of the random type of the data no information related to data (average hamming distances between data terms of the input vector) could be derived and used for the reordering of computation. Only the static information related to coefficients was used for determining the optimal order of computation. The effect of the proposed transformation on the activity of the data input is random. In one case penalty is introduced while in the other savings are achieved. Both savings and penalty are very small. The remaining filters (15, 11, 9, 8, 6 taps) are low-pass wavelet filters. In this case for the simulation real image data were used. For the representation of the data 8 bits were used while for the coefficients and the outputs 16 bits were used. The dynamic range of the representation of the coefficients is 15 bits. In this case the effect of the proposed methodology on the data activity is clearer. The proposed transformation introduces significant penalty although information related to the application data was extracted (as described in Section 3) and used for the reordering of computation. This is because the reordering of computation causes a reordering of the data terms, which destroys their correlation and increases the data activity. However the penalty introduced is much smaller than the savings in coefficient activity leading to significant total switching and power savings. The above results assume computation of each output term (inner product) on the same multiply accumulator.

6.2. Two Dimensional Wavelet Transforms

The proposed methodology was applied to the typical three-level two-dimensional wavelet transform.

TABLE V Experimental results for FIR filters

Filter	Data	Change in coefficient activity (# transitions-%)	Change in data activity (# transitions-%)	% change in total input activity	% change in internal activity
63 tap	Ran.	−180181 (−62%)	−1827 (−0.4%)	−40%	−31%
14 tap	Ran.	−115296 (−75%)	+1321 (+0.9%)	−37%	−29%
15 tap	Imag.	−1127779 (−52%)	+503139 (+35%)	−28%	−13%
11 tap	Imag.	−745479 (−48%)	+283735 (+25%)	−19%	−13%
9 tap	Imag.	−496987 (−43%)	+194244 (+21%)	−17%	−15%
8 tap	Imag.	−152918 (−13%)	+75374 (+15%)	−12%	−11%
6 tap	Imag.	−133801 (−14%)	+13056 (+2%)	−12%	−7%

Six different filter sets were used and simulated. The EPIC filters are described in [27], the Daubechies filters are described in [20] while Johnston's filters are described in [28]. The simulation results for two grayscale test images (Lena and Man) and different set of filters (high-pass and low-pass) are included in Tables VI, VII.

For the coefficients, a 16-bit representation (15 bits dynamic range) was used. The input data were represented by 8 bits while for the representation of the wavelet coefficients 16 bits were used. Implementation on a single multiply accumulator was assumed. The switching activity in the coefficient inputs of the computational units is significantly reduced (13.3% worst case, 51.8% best case). The effect of the computation reordering is different in the data inputs of the computational units. In general the computation reordering introduces a penalty (worst case 12%) in the switching activity in these inputs. This is because the reordering destroys data correlation. Specifically the reordering of computation and thus of data, introduces switching activity penalty in the inputs of the three levels of the wavelet decomposition because at these points the correlation of data is high. In the intermediate and output stages of the decomposition levels data are less correlated and thus the result of the reordering is more random (introduces either small savings or small penalties in switching activity). However, the penalty in data activity introduced is much smaller in comparison to the savings achieved, making the use of the proposed methodology advantageous.

Two basic parameters affecting the result of the reordering on the switching activity at the inputs of the computational units are: (a) The filter length and (b) The symmetry of the filter. From Tables VI, VII it is obvious that as the length of the filter increases the savings in coefficient activity increase but the penalty in data activity increases also. In general as the length of the filter increases the total activity required for the wavelet decomposition increases either with or without computation reordering. In cases of symmetrical filters ($h[n] = h[N-1-n]$, N: filter length) the savings in coefficient activity by the application of computation reordering are higher.

TABLE VI Simulation results for image Lena

Filter set	Change in coefficient activity (# transitions)	Change in data activity (# transitions)	% change in total input activity	% change in internal activity
EPIC - 15 tap	−20300019 (−51.8%)	+2698508 (+12%)	−29	−19
EPIC - 11 tap	−13418613 (−48.1%)	+954075 (+5.4%)	−28	−16
EPIC - 9 tap	−8945772 (−43.3%)	+853493 (+6.5%)	−24	−13
Daubechies - 6 tap	−2408421 (−14%)	+281648 (+3.1%)	−9	−6
Daubechies - 8 tap	−2752521 (−13.3%)	+646421 (+5.4%)	−9.5	−11
Johnston's - 8 tap	−5849097 (−34.3%)	+599234 (+5.1%)	−17	−5

TABLE VII Simulation results for image Man

Filter set	Change in coefficient activity (# transitions-%)	Change in data activity (# transitions-%)	% change in total input activity	% change in internal
EPIC - 15 tap	−20300019 (−51.8%)	+2686524 (+11.1%)	−29	−18
EPIC - 11 tap	−13418613 (−48.1%)	+1037449 (+5.4%)	−27	−15
EPIC - 9 tap	−8945772 (−43.3%)	+927875 (+6.2%)	−23	−12
Daubechies - 6 tap	−2408421 (−14%)	+314922 (+3,2%)	−8	−6.5
Daubechies - 8 tap	−2752521 (−13.3%)	+659390 (+5%)	−8	−10
Johnston's - 8 tap	−5849097 (−34.3%)	+624293 (+4.9%)	−16	−15

6.3. Fourier Transform

The proposed transformation was applied to the one dimensional 8-point brute force Discrete Fourier Transform (DFT) and three common one dimensional Fast Fourier Transforms (FFT) [29], the 9-point PTL FFT, the 7-point Singleton FFT and the 9-point Swift FFT. Real image data (test images Lena, Man, Camera and Lax) were used for the simulation. The bit-width of the input data was 8 while 16 bits were used for the representation of the output fourier data. The simulation results for both Cases 1 and 2 of description (presented in Section 3) and for several coefficients bit-widths ($C = 16, 14, 12, 10, 8$) are shown in Tables VIII–XV.

As far as the case 1 descriptions are concerned a penalty in data switching activity is introduced in the brute force DFT. However this penalty is much smaller in comparison to the large savings in coefficient activity achieved by the application of the proposed methodology. The FFTs effectively reduce the length of the input data vector through a preprocessing stage to reduce the number of operations required. For the 9 points PTL and Swift FFTs the input data vector used for the computation of the fourier data is a 4 point one, while for the 7 point Singleton FFT the input data vector is reduced to a 3-point one. This leads to small changes (either savings or penalty) in the data switching activity by the application of the computation reordering. Although the length of the FFTs is relatively small the reordering of computation introduces significant savings in the coefficient inputs of the computational units. For the Case 2 descriptions large switching activitiy savings are achieved in the data inputs of the computational units. The savings at the coefficient inputs of the computational units are of the same order as for the Case 1 descriptions. Larger savings at the inputs and inside the computational units are achieved in comparison to Case 1 descriptions. The savings percentage at the inputs of the computational units is not heavily affected by the changes of the coefficient bit-width. The activity savings percentage inside the computational units almost remains unchanged as the coefficient bit-width changes.

6.4. Discrete Cosine Transform

The reordering of computation is also applied to both 1-D and 2-D Discrete Cosine Transform.

TABLE VIII Simulation results for 8-point DFT and Case 1 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Lena	16	+154132 (+10.8%)	−2555898 (−62%)	−43	−42.5
	14	+154132 (+10.8%)	−1638385 (−60%)	−37	−42.5
	12	+154132 (+10.8%)	−1547496 (−60%)	−35	−42
	10	+154132 (+10.8%)	−1452355 (−59%)	−33	−42
	8	+154132 (+10.8%)	−1310708 (−58%)	−32	−41

TABLE IX Simulation results for 8-point DFT and Case 2 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Lena	16	−1260252 (−88%)	−2211828 (−55%)	−68	−59
	14	−1260252 (−88%)	−1425393 (−54%)	−66	−59
	12	−1260252 (−88%)	−1373875 (−53%)	−65	−58
	10	−1260252 (−88%)	−1255356 (−53%)	−64	−58
	8	−1260252 (−88%)	−1163252 (−52%)	−62	−58

TABLE X Simulation results for 9-point PTL FFT and Case 1 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Man	16	+15252 (+3%)	−301048 (−18.5%)	−14.5	−10
	14	+15252 (+3%)	−258032 (−18%)	−13.5	−10
	12	+15252 (+3%)	−215016 (−17%)	−12.5	−10
	10	+15252 (+3%)	−186368 (−19%)	−12.8	−10
	8	+15252 (+3%)	−114686 (−15%)	−12	−10

TABLE XI Simulation results for 9-point PTL FFT and Case 2 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Man	16	−414046 (−72%)	−286564 (−20%)	−38	−22
	14	−414046 (−72%)	−258048 (−18%)	−33	−22
	12	−414046 (−72%)	−186716 (−23%)	−38	−21
	10	−414046 (−72%)	−157704 (−15.5%)	−36	−21
	8	−414046 (−72%)	−129030 (−16%)	−35	−20

TABLE XII Simulation results for 7-point Singleton FFT and Case 1 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Camera	16	−1355 (−0.3%)	−331780 (−25%)	−20	−13
	14	−1355 (−0.3%)	−239624 (−21%)	−16	−12
	12	−1355 (−0.3%)	−294916 (−29%)	−21	−12
	10	−1355 (−0.3%)	−258056 (−30%)	−21	−12
	8	−1355 (−0.3%)	−110594 (−18%)	−17	−12

TABLE XIII Simulation results for 7-point Singleton FFT and Case 2 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Camera	16	−225985 (−62%)	−331780 (−25%)	−35	−18
	14	−225985 (−62%)	−239624 (−21%)	−30	−18
	12	−225985 (−62%)	−294916 (−29%)	−38	−18
	10	−225985 (−62%)	−258056 (−30%)	−39	−18
	8	−225985 (−62%)	−110594 (−18%)	−34	−18

TABLE XIV Simulation results for 9-point Swift FFT and Case 1 description

Image	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Lax	16	+5045 (+0.7)	−272376 (−20%)	−13	−10
	14	+5045 (+0.7)	−272368 (−18%)	−14	−10
	12	+5045 (+0.7)	−243716 (−22%)	−14	−10
	10	+5045 (+0.7)	−186374 (−20%)	−13	−10
	8	+5045 (+0.7)	−129032 (−16%)	−12	−10

TABLE XV Simulation results for 9-point Swift FFT and Case 2 description

Image	C	Change in data act. (# trans. -%)	Change in coeff. act. (# trans. -%)	% change in total input activity	% change in internal activity
Lax	16	−447749 (−70%)	−315382 (−21%)	−35	−19
	14	−447749 (−70%)	−329724 (−24%)	−39	−19
	12	−447749 (−70%)	−229380 (−20%)	−38	−19
	10	−447749 (−70%)	−258052 (−27%)	−36	−19
	8	−447749 (−70%)	−172036 (−21%)	−33	−19

Specifically the proposed methodology was applied to 8 and 16 point 1-D DCT. The technique for fast DCT computation that decomposes an N-point input vector to two N/2 point sequences [28] was used. Real speech data were used for the simulation. Two sequences with real speech data (Speech1, Speech2) were used specifically. The simulation results for both Cases 1 and 2 of description and for several coefficients bit-widths (C = 16, 14, 12, 10, 8) are shown in Tables XVI–XXI.

The proposed methodology was applied to the 2-D DCT computed using the row column decomposition and the fast technique described in [28]. For simulation the test images Lena and Man were used. For the representation of the coefficients 16 bits were used while the input data were represented using 8 bits. For the representation of the output terms 16 bits were used. The simulation results are presented in Table XX.

Conclusions similar to those drawn in the previous sections can be derived. As the effective length of the input data vector is increased the penalty introduced by the reordering of computation is increased (row-column 8-point DCT). The reduction of the effective length of the input data vector by the preprocessing step of a fast algorithm leads to smaller changes (either savings or penalties) of the switching at the data inputs of the computational units after the application of the proposed transformation. The results for the 1-D DCTs prove that the savings in input switching activity do not always increase as the transform length increases. The same is true for the coefficient bit-width. Increase of the coefficient bit-width does not always lead to higher savings percentage. Changes of the coefficient bit-width lead to small changes of the savings percentage in the total activity at the inputs of the functional units

TABLE XVI Simulation results for 16-point 1-D DCT and Case 1 description

Input	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Speech1	16	−5470 (−0.6%)	−835583 (−20%)	−17	−11
	14	−5470 (−0.6%)	−712702 (−20%)	−16	−11
	12	−5470 (−0.6%)	−671740 (−22%)	−17	−11
	10	−5470 (−0.6%)	−548857 (−21%)	−16	−11
	8	−5470 (−0.6%)	−401405 (−19.7%)	−14	−11

TABLE XVII Simulation results for 16-point 1-D DCT and Case 2 description

Input	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Speech1	16	−583393 (−68%)	−1081340 (−26%)	−33	−20
	14	−583393 (−68%)	−811003 (−23%)	−31	−20
	12	−583393 (−68%)	−753659 (−25%)	−34	−20
	10	−583393 (−68%)	−688125 (−26%)	−36.5	−20
	8	−583393 (−68%)	−425982 (−20%)	−35	−20

TABLE XVIII Simulation results for 8-point 1-D DCT and Case 1 description

Input	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Speech2	16	+32118 (+6.7%)	−606206 (−29%)	−25	−18
	14	+32118 (+6.7%)	−589815 (−33%)	−27.5	−18
	12	+32118 (+6.7%)	−524288 (−33%)	−27	−18
	10	+32118 (+6.7%)	−425982 (−32%)	−25	−18
	8	+32118 (+6.7%)	−311288 (−30%)	−25	−18

TABLE XIX Simulation results for 8-point 1-D DCT and Case 2 description

Input	C	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Speech2	16	−357390 (−75%)	−524283 (−25%)	−34	−21
	14	−357390 (−75%)	−376826 (−21%)	−32	−21
	12	−357390 (−75%)	−344063 (−22%)	−34	−21
	10	−357390 (−75%)	−360447 (−27%)	−39	−21
	8	−357390 (−75%)	−278525 (−27%)	−33	−21

TABLE XX Simulation results for the 8-point 2-D DCT

Image	Algorithm	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Lena	Row-Col.	+660645 (+16%)	−2785283 (−35%)	−18	−14
Man	Row-Col.	+560116 (+13%)	−2785283 (−35%)	−18	−12
Lena	Fast DCT	−20340 (−1.7%)	−1212414 (−29%)	−22	−17
Man	Fast DCT	+17799 (+2%)	−1212414 (−29%)	−21	−16

TABLE XXI Simulation results for the 8-point 2-D DCT realized on the 4 parallel MACs architecture

Image	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Lena	−17848 (−2.3%)	−987484 (−25%)	−19	−15
Man	+20547 (+1.6%)	−987484 (−25%)	−18	−15

while the activity savings percentage in the computational units remains almost unchanged. For the 2-D DCT the behavior of the outputs of the rows DCTs (row- column decomposition is always assumed for DCT computation even when a fast algorithm is used) is random since these data are less correlated. This leads to smaller penalties introduced by the computation reordering transformation in the data inputs of the computational units during computation of the columns DCTs.

7. GENERALIZATION OF COMPUTATION REORDERING METHODOLOGY

So far it was assumed (to simplify analysis) that the total inner products computations required by

the algorithms were performed on a single multiply accumulator. Within inner product computations there are no data dependencies meaning that the partial products forming the inner products can be evaluated independently from each other and thus in parallel. Under these assumptions the problem of the computation reordering was formulated as a minimum cost (input switching activity) scheduling of a number of independent operations (partial products). For the second class of algorithms it was assumed that the inner products (Case 1) or the sets of partial products (Case 2) were computed one after the other in the order specified by the first ordering function.

However in many real-life cases the number of the available hardware resources (multiply accumulators) is larger and determined by the performance requirements of the specific application.

Two cases exist: (a) Each one of the basic computations included in the algorithm is performed on a single multiply accumulator and not distributed over a number of hardware resources. The available multiply accumulators are used to perform different basic computations in parallel. In this case the formulation described in Section 3 still applies. (b) Every basic computation is divided to a number of parts that are assigned to an equal number of hardware resources. The computational parts are executed in parallel. Modification of the formulation described in Section 3 is required for this case.

For the first case experimental results proved that the savings (%) from the application of the computation reordering are independent on the number of the available hardware resources and remain the same as in the case where the total computation was performed in a single multiply accumulator. As far as the second case is concerned experimental results proved that the switching activity savings from the application of the computation reordering depend on the number of resources on which the basic computations are assigned and specifically they slightly decrease as the number of the resources increases. This is illustrated in the following section.

7.1. Distribution of Basic Computation to a Number of Multiply Accumulators

In this case the basic computations are divided into equal parts, assigned to the available multiply accumulators and performed in parallel. The number of parts into which the basic computations are divided is equal to the number of the available multiply-accumulators. Since no dependencies exist between the partial products of the basic computations, for both classes of algorithms, the assignment of the computational parts to the hardware resources can be performed randomly and the only restriction is the load balancing.

The formulation of the computation reordering problem, when basic computations are distributed to a number of multiply accumulators, for the two classes of algorithms is as follows:

7.1.1. First Class

Assuming computation of convolutions of size N and M multiply accumulators available, the basic computation must be divided to parts of $\lceil N/M \rceil$ partial products. (When N mod $M \neq 0$ some multiply accumulators may perform less than N/M operations). The assignment of the partial products to the multiply-accumulators can be done in a random way. Another solution is to reorder the basic computation as described in section 3 and then assign $\lceil N/M \rceil$ consecutive (wrt the new ordering) partial products to each one of the available hardware resources. In this way the correlation of the partial products assigned to the same multiply accumulator is improved in comparison to the random assignment. In both cases the partial products assigned to each multiply-accumulator can be thought to form a "virtual" basic computation (of reduced size in comparison to the original one). The formulation described in Section 3.1 can then be applied to each virtual computation and a minimum input switching activity schedule of the partial products on the multiply accumulator can be derived.

7.1.2. Second Class

Assuming computation of transforms of size $N \times M$ and K multiply accumulators available, the basic computation must be divided to parts of $\lceil N \times M/K \rceil$. The assignment can be performed once more either in a random way or by performing reordering of computation as described in Section 3 in a first step to increase the correlation of the partial products assigned to the same multiply accumulator. In the first step complete inner products are assigned to the available multiply accumulators. Inner products are divided to different multiply accumulators only for load balancing reasons. The partial products assigned to each multiply accumulator form a "virtual" basic computation. Two cases exist: (a) The partial products assigned to a multiply accumulator belong to the same inner product of the initial computation. In such a case the virtual basic computation is though to be of first class (convolution type) and the formulation described in Section 3.1 can be applied to derive a minimum input switching activity schedule of the partial products on the multiply accumulator. (b) The partial products assigned to a multiply accumulator belong to n different inner products of the initial computation. In such a case the virtual basic computation is though to be of the second class (transformation type) and the formulation described in Section 3.2 can be applied to derive a minimum input switching activity schedule of the partial products on the multiply accumulator.

To demonstrate the effect of computation distribution on the savings achieved by the proposed methodology the four parallel multiply accumulators architecture for the computation of the two-dimensional 8-point DCT described in [19] is used. The fast DCT that decomposes the 8-point input data vectors to two 4-point vectors (u, v) is used. For the computation of the 8 output points for a specific 8-point input data vector, 8 4-point inner products must be computed. These inner products are assigned to the 4 available multiply accumulators (2 inner products on each multiply accumulator). Having determined the minimum connection costs between all pairs of inner products as described in Section 3 the pairs of inner products with the smaller minimum connection costs are selected and assigned to each multiply accumulator. Now it can be assumed that each multiply accumulator has to compute a 2×4 transform (2 4-point inner products). The problem of finding the best computation ordering for each multiply accumulator can be tackled using the formulation proposed for the algorithms of class 2.

Simulations using the test images Lena and Man were performed. The bit-width of the input data was 8 while 16 bits were used for the representation of the coefficients and the output terms. The results are described in Table XXI. The savings in coefficient activity, total input activity and activity inside the computational units are slightly decreased in comparison to the results of Table XX describing the results of the proposed transformation under the assumption that the basic computation is performed on the same multiply accumu- lator. The distribution of the inner products restricts the possibilities for computation reordering thus leading to slight decrease of the activity savings.

7.2. Application of the Proposed Methodology in General Architectures

The proposed transformation can be applied in a more general data path structure (*i.e.*, not necessarily multiply accumulator based). The sets of partial products are assigned to the available multipliers (instead of multiply accumulators) and computed in parallel. The formulation of the computation reordering problem in this case is the same as the one described in previous section (*i.e.*, for multiple multiply accumulators) except from the fact that the multiply accumulators are replaced by multipliers.

To demonstrate this the 4 parallel multipliers architecture [28] for the computation of the DCT

TABLE XXII Simulation results for the 8-point 2-D DCT realized on the 4 parallel multipliers architecture

Image	Change in data act. (# trans.-%)	Change in coeff. act. (# trans.-%)	% change in total input activity	% change in internal activity
Lena	−769244(−20%)	−1245161(−35%)	−25	−20
Man	−673366(−17%)	−1245161(−35%)	−24	−18

is used. In this architecture an 8-point DCT can be computed using the fast DCT technique described in the previous section on four multipliers and one accumulator/adder. The structure of the architecture imposes that the 4 partial products required for the evaluation of an inner product (a transform's output point) are computed in parallel each one in one of the multipliers. A heuristic strategy was adopted for scheduling of the inner products and the assignment of the partial products to the multipliers. For all the pairs of inner products the hamming distances between all possible pairs of partial products (one partial product from each inner product) are computed. The minimum cost connection on the specific architecture for every pair of inner products is determined. The connection cost between two inner products is defined as the sum of hamming distances of the partial products (belonging to the inner products) assigned to the same multiplier. Having determined the minimum connection cost for all the possible pairs of inner products the problem of finding the optimal in terms of switching activity at the multipliers inputs is formulated as a Travelling Salesman Problem on a graph with vertices the inner products. The costs of the graph's edges are the hamming distances of the minimum cost connections of the inner products corresponding to the nodes. Thus an optimal scheduling of the inner products on the available hardware can be determined.

Simulations for the test images Lena and Man were performed. The bit-width of the input data was 8 while 16 bits were used for the representation of the coefficients and the output terms. The results are described in Table XXII.

The results prove that significant savings can be obtained for the data input of the computational units by the application of the proposed methodology.

8. CONCLUSIONS

In this paper a novel architectural transformation for power consumption reduction in hardware realizations of digital signal processing algorithms requiring inner product computation was presented. Two different classes of algorithms requiring inner product computation between data and coefficients were identified. A low power target architecture for both classes of algorithms was described. It is based on a set of foreground registers as a form of memory hierarchy that allows data reuse and reduces the background memory related power consumption.

The transformation reorders the computation to reduce the switching activity at the inputs and thus inside the computational units. Reduction of the switching activity at the inputs and inside the computational units is equivalent to power consumption reduction. The reordering of computation aims at deriving an optimal, in terms of switching activity at the inputs of the computational units, schedule (and assignment in cases with more than one hardware units available) of the partial products constituting the inner products required by the algorithms on the computational units. Information related to both algorithm's coefficients (which are statically determined *i.e.*, known before run time) and application data is used for this reason. The problem of the computation reordering has not the same structure for the different classes of algorithms. Formulations of the reordering problem are proposed for the two classes of algorithms. In both cases a

Travelling Salesman Problem must be solved for the derivation of the optimal computation order. The generalization of the methodology for the cases where the inner products of a transformation are distributed over a set of hardware resources (multiply accumulators) was also presented.

The proposed methodology was applied to several digital signal-processing algorithms. Experimental results prove that the reordering of computation leads to significant activity savings in the computational units (data paths) even when the length of the inner product computation required by the algorithm is relatively small.

Future research includes further refinement of the computation reordering problem formulation for less regular data paths *i.e.*, data paths in which the computational units are discrete multipliers and adders and not multiply accumulators.

References

[1] Rabaey, J. M. and Pedram, M., "Low Power Design Methodologies", Kluwer Academic Publishers 1995.

[2] Chandrakasan, A. P., Potkonjak, M., Mehra, R., Rabaey, J. and Brodersen, R. W., "Optimizing Power Using Transformations", In: *IEEE Tran. on CAD of Integrated Circuits and Systems*, **14**(1), 12–30 (January 1995).

[3] Srivastava, M. and Potkonjak, M. (1996). "Power Optimization in Programmable Processors and ASIC Implementations of Linear Systems: Transformation based Approach", *Proceedings of the 33rd Design Automation Conference* (DAC).

[4] Kim, D. and Choi, K. (1997). "Power Conscious High Level Synthesis Using Loop Folding", *Proceedings of the 34th Design Automation Conference* (DAC), Anaheim, California.

[5] Simon, S., Schimpfle, C. V., Wroblewski, M. and Nossek, J. A., "Retiming of Latches for Power Reduction of DSP Designs", *Proceedings of the IEEE International Symposium on Circuits and Systems 1997 (ISCAS'97)*, pp. 2168–2171.

[6] Dempster, A. G. and Mcleod, M. D., "Use of minimum adder multiplier blocks in FIR digital filters", *IEEE Transactions on Circuits and Systems II*, **42**, 569–577 (September 1995).

[7] Hartley, R. (1991). "Optimization of CSD multipliers for filter design", In: *IEEE International Symposium on Circuits and Systems*, **4**, 1992–1995.

[8] Mehendale, M., Sherlekar, S. D. and Venkatesh, G. (1995). "Coefficient optimization for low power realization of FIR filters", In: *IEEE Workshop on VLSI Signal Processing*, Japan, pp. 352–361.

[9] Sankarayya, N., Roy, K. and Bhattacharya, D., "Algorithms for Low Power and High Speed FIR Filter Realization Using Differential Coefficients", *IEEE Transactions on Circuits and Systems II*, **44**(6), 488–497 (June 1997).

[10] Pearson, D. and Parhi, K., "Low Power FIR Digital Filter Architecture", *Proc. of the 1995 IEEE Intl. Symposium on Circuits and Systems*, pp. 231–234.

[11] Wu, A. Y. and Liu, K. J. R., "A Low-Power and Low Complexity DCT/IDCT VLSI Architecture Based on Backward Chebyshev Recursion", *Proc. of the 1994 IEEE Intl. Symposium on Circuits and Systems*, pp. 155–158.

[12] Hsiao, S. F. and Yen, C. Y., "Power, Speed and Area Comparison of Several New DFT Architectures", *Proc. of the 1997 IEEE Intl. Symposium on Circuits and Systems*, pp. 2577–2580.

[13] Wu, A. Y., Liu, K. J. R., Zhang, Z., Nakajima, K. and Raghupathy, A., "Low Power Design Methodology for DSP Systems Using Multirate Approach", *Proc. of the 1996 IEEE Intl. Symposium on Circuits and Systems*, pp. 292–295.

[14] Wu, A. Y. and Liu, K. J. R., "Algorithm-Based Low-Power Transform Coding Architectures", *Proc. of the 1995 IEEE Intl. Conf. on Acoustics Speech and Signal Processing*, pp. 3267–3270.

[15] Parhi, K. (1993). "Algorithms and Architectures for High-Speed and Low-Power Digital Signal Processing", *Proc. of 4th Intl. Conference on Advances in Communications and Control*, June 14–18, Rhodes, Greece, pp. 259–270.

[16] Chatterjee, A. and Roy, R., "Synthesis of Low Power Linear DSP Circuits using Activity Metrics", *Proc. of the 7th Intl. Conference on VLSI Design-January*, pp. 265–270 (1994).

[17] Wuytack, S., Catthoor, F., Nachtergaele, L. and De Man, H., "Power Exploration for Data Dominated Video Applications", *Proc. of the 1996 Intl. Symposium on Low Power Electronics and Design*, Monterey, California, pp. 359–364.

[18] Diguet, J. P., Wuytack, S., Catthoor, F. and DeMan, H., "Hierarchy Explorartion in High Level Memory Management", In: *Proc. of the 1997 International Symposium on Low Power Electronics and Design*, Monterey CA, August pp. 18–20.

[19] Miyazaki, T., Nishitani, T., Edahiro, M. and Ono, I. (1993). "DCT/IDCT Processor for HDTV Developed with DSP Silicon Compiler", *Journal of VLSI Signal Processing*, **5**, 151–158, Special issue on VLSI Video/Image Signal Processing.

[20] Grzeszczak, A., Mandal, M., Panchanathan, S. and Yeap, T., "VLSI Implementation of Discrete Wavelet Transform", *IEEE Trans. on VLSI Systems*, **4**(4), (December 1996).

[21] Papadimitriou, C. H. and Steiglitz, K. (1982). "Combinatorial Optimisation: Algorithms and Complexity", Prentice-Hall, Inc., Englewood Cliffs.

[22] Christofides, N., "Worst-case analysis of a new heuristic for the travelling salesman problem", Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA.

[23] Cormen, H., Leiserson, C. E. and Rivest, R. L. (1990). "Introduction to Algorithms", Second Edition, McGraw-Hill.

[24] Kulkarni, D. and Stumm, M., "Loop and Data Transformations: A tutorial", Technical Report CSRI-337, Computer Systems Research Institute, University of Toronto (June 1993).

[25] Rabaey, J., Guerra, L. and Mehra, R. (1995). "Design Guidance in the Power Dimension", *Proc. of the IEEE ICASSP*.

[26] Tsui, C. Y., Chan, K. K., Wu, Q., Ding, C. S. and Pedram, M., "A Power Estimation Framework for Designing Low Power Portable Video Applications", *Proc. of the 1997 Design Automation Conference* (DAC 97), Anaheim, California.
[27] Simoncelli, E. P., Adelson, E. H., "Efficient Pyramid Image Coder (EPIC)", Vision Science Group, Massachusetts Institute of Technology.
[28] Bhaskaran, V., Konstantinides, K. (1994). "Image and Video Compression Standards", Kluwer Academic Publishers.
[29] Smith, W. and Smith, J. (1995). "Handbook of Real-Time Fast Fourier Transforms", IEEE Press.

Authors' Biographies

Konstantinos Masselos was born in 1971 in Athens, Greece. He received a first degree in Electrical Engineering from University of Patras, Greece in 1994 and an MSc degree in VLSI Systems Engineering from University of Manchester Institute of Science and Technology (UMIST), United Kingdom in 1995. Since January 1996 he has been working towards a Ph.D. degree in Electrical and Computer Engineering at University of Patras, Greece. From January 1997 to July 1997 and May 1998 to November 1998 he joined the VSDM division of Inter-university Micro Electronics Centre (IMEC), Belgium. His current research interests are system level low-power design, image and video compression, including low-complexity implementations and fast algorithms, system and high level synthesis of VLSI circuits.

Panagiotis Merakos was born in Korydallos of Attiki, Greece, in 1968. He received the Diploma in Electrical Engineering from the University of Patras, Greece in 1992.

He is currently a researcher and studies towards the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Patras. His research interests include high-level algorithms and techniques for low-power DSP systems design, analysis and synthesis.

Dr. Thanos Stouraitis is an Associate Professor of Electrical and Computer Engineering at the University of Patras, Greece. He has also served on the faculty of The Ohio State University and the University of Florida.

He got his Ph.D. in Electrical Engineering from the University of Florida in 1986 (for which he received the Outstanding Ph.D. Dissertation Award), an MSc. in Electrical & Computer Engineering from the University of Cincinnati in 1983, an MS in Electronic Automation from the University of Athens, Greece, in 1981, and a BS in Physics from the University of Athens, Greece, in 1979.

His current research interests include signal and image processing, application-specific processor technology and design, design and architecture of optimal digital systems, and computer arithmetic. He leads several DSP processor design projects funded by the European Union and the Greek Covernment.

Dr. Costas Goutis was a Research Assistant and Research Fellow in the Department of Electrical and Electronic Engineering at the University of Strathclyde, U.K., from 1976 to 1979, and Lecturer in the Department of Electrical and Electronic Engineering at the University of Newcastle upon Tyne, U.K., from 1979 to 1985. Since 1985 he has been Associate Professor and Full Professor in the Department of Electrical and Computer Engineering, University of Patras, Greece. His recent research interests focus on VLSI Circuit Design, Low Power VLSI Design, Systems Design, Analysis and Design of Systems for Signal Processing and Telecommunications. He has been awarded a large number of Research Contracts from ESPRIT, RACE and National Programs.