A Novel Design Methodology for High Performance and Low Power Digital Filters

Khurram Muhammad and Kaushik Roy
Email: k-muhammad1@ti.com and kaushik@ecn.purdue.edu
Storage Products Group, Texas Instruments, Dallas, TX and School of ECE, Pur due University, West Lafayette, IN, USA.

I. INTRODUCTION

Future mobile radio and portable computing systems are expected to provide increased services, faster data rates and higher processing speeds at reduced power dissipation levels. Therefore new approaches are required in design of low-complexity and high-performance *digital signal processing* (DSP) blocks. The *classical* approach in complexity reduction is the use of techniques such as *recursion* (e.g. RLS, FFT algorithms), *multi-rate signal processing* and *low rank approximation* [1]. The last technique can be considered as a quantization approach. Low-complexity design not only improves the speed at which an algorithm processes data, but it also leads to low power design at the highest level of abstraction by reducing computation. High speed design can further help in power reduction by use of *voltage scaling*.

Complexity reduction in FIR digital filter implementations has been of particular interest to the DSP system design community [2]. Most previous approaches [3], [4], [5], [6] consider simplified parallel implementations of FIR filters for *signed powers-of-two* (SPT) and *canonical sign-digit* (CSD) number representations. One approach uses *integer linear programming* (ILP) to search for a filter which conforms to desired frequency response. Another approach is to start from a known optimal filter solution and search for *quantizations* in the vicinity of the optimal solution which gives a lower implementation cost. The prior approach is computationally impractical as the filter size or the number of bits used to represent a coefficient increases. In this paper, we present a new approach for low-complexity design of FIR filters. The basic approach is to *search for quantizations which increase computation sharing*. For given constraints on frequency domain transfer characteristics, we search for a solution in the vicinity of the known optimal solution such that computation sharing is maximized. The solution is used to obtain a very low-complexity implementation of parallel filters by reducing *computational redundancy* [7] using a graph theoretic approach which re-orders computation optimally. We define computational redundancy as the *excess computation over the minimum number of bit operations needed for a given sequence of operations*. The main contributions of this work are (1) The technique presented is independent of the choice of number representation scheme and/or the coefficient word-length, (2) We present a novel approach of *quantization for increased computation sharing* and (3) We show that extremely simple implementations can be obtained using simple graph algorithms.

II. General Background

Consider a *linear time-invariant* (LTI) FIR filter of length M described by an input-output relationship of the form

$$y(n) = \sum_{i=0}^{M-1} c_i\, x(n-i) = \sum_{i=0}^{M-1} P_i^{(n)} \qquad (1)$$

This work was supported in part by DARPA (F33615-95-C-1625), NSF CAREER award (9501869-MIP), Rockwell, AT&T and Lucent foundation.

In this context, c_i represents the *ith* coefficient and $x(n-i)$ denotes the data sample at time instant $n-i$. $P_i^{(n)}$ represents the partial product $c_i x(n-i)$ for $i = 0, 1, \ldots, M-1$ computed at time instant n. Figure 1 shows a graph $G = \{V, E\}$ representation of a 4-tap ($M = 4$) FIR filter in which each vertex represents a coefficient. Let us consider an implementation which is based on using first order differential coefficients (i.e. $c_i - c_j$, where $i \neq j$) rather than the original values. One such implementation is explored in [8] and is referred to as the *differential coefficient method*. Then, the edge $e_{i,j}$, $i, j = 0, 1, 2, 3$ represents *the resources required to multiply a data sample with the differential coefficient* $c_i - c_j$. If an array multiplier is used to compute the products in a parallel implementation of the filter, $e_{i,j}$ represents the number of rows of adders required to implement the multiplier and is given as the number of 1-bits in $c_i - c_j$. The rows corresponding to 0-bits of $c_i - c_j$ can be removed from the multiplier since each coefficient value is fixed. Therefore, $M = 4$ parallel multipliers are required to obtain a parallel implementation of the M-tap filter and $e_{i,j}$ depends only on the number representation scheme and the type of multiplier employed. We also note that G is undirected and $e_{i,j} = e_{j,i}$ for all $i, j = 0, 1, \ldots, M-1$.

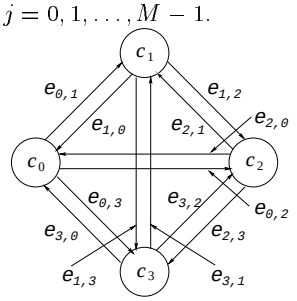

Fig. 1. Graph representation of an example filter with $M = 4$.

III. THE MINIMUM SPANNING TREE SOLUTION

Our goal is to compute $y(n)$ using equation 1 by computing the individual partial products. Consider the coefficient difference $\Delta_{i,j} = c_i - c_j$, for some $0 \leq i, j \leq M-1$ where $i \neq j$. Then $P_i^{(n)} = (c_i - c_j)x(n-i) + c_j x(n-i) = \Delta_{i,j} x(n-i) + P^{(n+j-i)}$. Now $e_{i,j}$ represents the resources required to compute the product of $\Delta_{i,j}$ with a data sample $x(n)$, at any time instant n and it equals the number of '1' bits in $\Delta_{i,j}$ (since each '1' bit corresponds to a row of adders). If $\Delta_{i,j}$ is simpler than c_i (e.g. it is a power-of-two), the complexity of implementation can be reduced since a full multiplication operation ($c_i x(n-i)$) is replaced by a simpler multiplication with an overhead of one *add* operation. This overhead add operation adds the correction term $P^{(n+j-i)}$ to $\Delta_{i,j} x(n-i)$.

A differential coefficient involving a given c_i can be obtained using any $c_i - c_j$, $j \neq i$. The amount of resources required in this selection can be expressed by assigning that value to the edge $e_{i,j}$. This value can also take into consideration the scheme used for number representation. As an example, if *sign-magnitude* (SM) number representation is used, $c_i = 33$, and, $c_j = 17$, $c_i - c_j = 33 - 17 = 16$ requires only one adder in implementation of the

multiplier. Since, the coefficient is differential, an overhead add operation is required with its use, we assign the value 2 to $e_{i,j}$.

The solution which minimizes the total number of resources in the implementation of the entire filter can be obtained using the *minimum spanning tree* (MST) [9] of G. The MST of a graph $G = \{V, E\}$ is defined as *an acyclic subset of edges in E which connects all of the vertices in V such that the sum of weights of these edges is minimized.* Hence, MST on G gives a coefficient order such that all the vertices of the graph are visited once and the total resources required to implement the differential filter are minimum. We refer to this solution as the *optimal differential coefficients for multiplierless implementation* (ODCMI) technique.

The approach discussed above can be *improved* if we also allow original coefficient values in addition to first order differential coefficients. Since the new solution is not strictly restricted to first order differential coefficients, it will be referred to as the *01-hybrid solution.* The basic approach is to obtain modified graph $\tilde{G}$ from G by inserting a new vertex c_M, which connects to all the other vertices c_i, $i = 1, 2, \ldots, M-1$, such that the weight of $e_{M,i}$ is assigned a value equal to the number of adds required if we use the original coefficient c_i without differencing (note that we choose $c_M = 0$). We observe that no overhead add operation is required with these edges. Further, $\tilde{G}$ is undirected and complete. The MST on $\tilde{G}$ will be the *optimal 01-hybrid ODCMI* solution which requires the least number of adders to implement the multiplication when both the original as well as first order coefficients are considered. The final solution can be partitioned into two sets; the first set constituting differential coefficients in which c_M is a parent or a child, and, the second set containing the remaining pairs. The elements in the first set are the coefficients which are applied without differencing. All pairs in the second set are employed using the first order difference. The cost of final solution is simply the sum of edge weights on all the parent-child pairs in the MST of $\tilde{G}$. For the problem of using least amount of resources in forming the partial products in equation 1, MST yields the optimal solution to the problem of minimizing the number of add operations. The best known algorithm to compute the MST executes in $\Theta(|E| + |V|log|V|) \approx \Theta(|V|^2)$ run time by employing *Fibonacci heaps* [9]. A simple algorithm by Prim runs in $\Theta(|V|^2 log|V|)$ time [9].

IV. IMPLEMENTATION OF THE ODCMI FILTER

Figure 2 shows the implementation of an FIR filter using the MST solution (MST shown in figure 3). The coefficient sequence applied using the MST of G is obtained by applying the differential coefficients $c_{child} - c_{parent}$, where c_{child} and c_{parent} pair consists of all possible *parent-child* pairs in the MST (leaf-nodes have no child). The *parent* of the *root node* of the MST is defined as 0. For example, the MST in figure 3 yields the coefficients $c_0, c_1 - c_0, c_2 - c_3, c_3 - c_0, c_4 - c_1, c_5 - c_2, c_6 - c_3, c_7 - c_3$, as shown in figure 2(a). Let $\mathcal{P} = \{p_0, p_1, \ldots, p_{M-1}\}$ and $\mathcal{Q} = \{q_0, q_1, \ldots, q_{M-1}\}$ denote the index sets of parent and child nodes, respectively. In the above example, $\mathcal{P} = \{0, 0, 3, 0, 1, 2, 3, 3\}$ and $\mathcal{Q} = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Then, the partial products are

$$P_i^{(n)} = (c_{q_i} - c_{p_i})x(n - q_i) + P_{p_i}^{(n-q_i+p_i)} \qquad (2)$$

for $i = 0, 1, \ldots, M-1$. An implementation for the MST in in figure 3 is shown in figure 2(a). Note that it is non-causal and output is available after a delay equal to $max(p_i - q_i, 0)$ for $0 \leq i \leq M-1$.

We can simplify the implementation shown in figure 2(a) by re-timing the filter to obtain a transpose direct form. The first step is to move the delay elements to the multiplier inputs. Consequently, the ith branch containing a multiplier has i delay elements on it after the first step is completed. Next, we can move the delay elements further down such that the the multiplier precedes the delay elements, and then, move them even further down such that the overhead add operations also precede these delay elements. Next, we determine the correct partial product term as a consequence of moving the delay elements across the overhead partial product add operations. Finally, the delay elements are moved out of these branches to get the transposed-direct form implementation shown in figure 2(b). We observe that the filter output is available with an extra delay of $\Theta(logM)$ add operations due to the overhead add network, however, no overhead memory is required. This delay can be eliminated by pipelining the structure of figure 2(b). Next, we show that such a structure can be obtained for any MST describing a coefficient ordering.

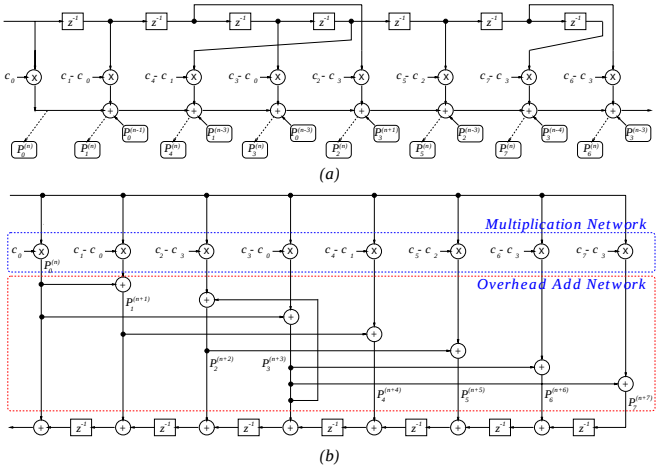

Fig. 2. Implementation of the 8-tap example filter using ODCMI.

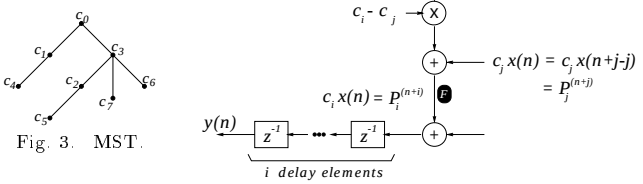

Fig. 3. MST.

Fig. 4. ith branch of the ODCMI filter.

Lemma: Every ODCMI filter can be implemented using a transpose direct form.

Proof: Consider the transpose-direct form filter implementations of figure 2(b). Let us define the ith branch as the branch which is separated from the filter output by i delay elements. At time instant n, the output of the filter must be $y(n) = \sum_{i=0}^{M-1} P_i^{(n)}$, which implies that ith branch must have $P_i^{(n+i)}$ at its own output (point F shown in figure 4) since the contribution to the filter output by this branch is delayed by i delay elements. The ith branch is shown for clarity in figure 4. Let the differential coefficient at this branch be $c_i - c_j$ for any $j \neq i$ such that $0 \leq j \leq M-1$. If $c_j = 0$, we do not need an overhead add operation. Otherwise, we must add $c_j x(n)$ to $(c_i - c_j)x(n)$ in order to obtain $c_i x(n) = c_i x(n+i-i) = P_i^{(n+i)}$ which is the correct desired output at point F. But we note that $c_j x(n) = c_j x(n+j-j) = P_j^{(n+j)}$. Since $0 \leq i, j \leq M-1$, for all $P_i^{(n+i)}$ such that $0 \leq i \leq M-1$ a $P_j^{(n+j)}$ is available at the point F of branch j. Therefore, for any i, j, we can use the output of branch j at point F to ensure that the output of branch i is $P_i^{(n+i)}$. Therefore we can obtain

a transpose direct form for any coefficient ordering which does not contain cycles. Hence, any ODCMI filter can be implemented using a transpose direct form since it is a tree based solution.

With the above theorem, obtaining implementation shown in figures 2(b) now becomes trivial. We further draw the readers attention to the fact that the interconnection network of the overhead add network traces an image of the tree solution. This is because it must preserve the parent-child relationship between computations as parent computation is used to simplify the child computation in the ODCMI scheme. Note that the image of tree in figure 3 can be traced in the overhead add wiring network.

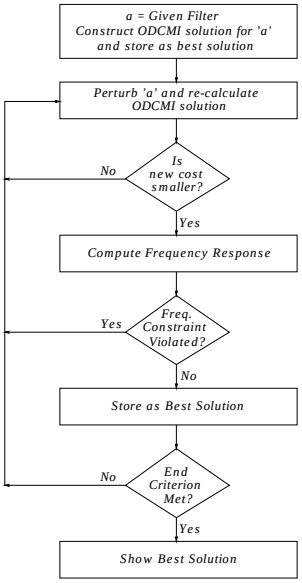

Fig. 5. Flow-chart of the perturbation based ODCMI approach.

V. QUANTIZATION FOR INCREASED COMPUTATION SHARING

In this section we will consider further complexity reduction using ODCMI approach when we are allowed to perturb the filter coefficients such that the constraints specified on the frequency response are not violated. The constraints will be assumed to be specified as a window which provides upper and lower bounds on the maximum allowed deviation from the optimal response. The flow chart of our approach is shown in figure 5 and we will refer to it as the *quantization based ODCMI* (QODCMI) approach. The algorithm starts from a given filter with coefficient vector $\mathbf{c}$ and computes the ODCMI solution. The cost of MST obtained for the vector $\mathbf{c}$ is stored as the *best solution*. Next, we explore improving the cost of the MST by perturbing $\mathbf{c}$. This is accomplished by adding a vector $\mathbf{e}$ to $\mathbf{c}$. The new coefficient vector (i.e. $\mathbf{c}+\mathbf{e}$) is used to compute the ODCMI solution for perturbed filter. The cost of MST obtained using this approach is compared with the best cost and if worse, we try a new perturbation vector.

An appropriate selection of $\mathbf{e}$ significantly speeds up the execution time of the QODCMI approach. If $\mathbf{e}$ improves the cost of ODCMI solution, we compute the frequency response of the new filter using the FFT algorithm. The frequency response is checked against the constraint window and if it violates the window at any frequency, we reject the solution and try a new perturbation. However, if the constraints on frequency response are not violated, we select this vector as the new best solution. If the end criterion is not met, we proceed to search for an even better solution by returning to the step where a new perturbation is computed and explored. Otherwise, we stop the algorithm and show the best ODCMI solution. Since, the frequency response characteristics of the perturbed filter are computed for every potential improvement in the best solution, the final solution is guaranteed to satisfy the constraints on the frequency response transfer characteristics.

The complexity of this algorithm is dependent mainly on the choice of the perturbation strategy and the end criterion. End criterion can be based on relative improvement obtained in successful attempts. If no improvement is obtained for a predetermined number of iterations, we can terminate search. If the number of perturbations explored before stopping the search is a polynomial function of M, the complexity of the algorithm is also polynomial function of M and is mainly dominated by the cost of solving ODCMI problem. The frequency response computation has a complexity of $NlogN$ where N is the number of points considered in computing the FFT.

VI. Perturbation Strategy

The MST solution obtained by the ODCMI technique gives the least cost spanning-tree which is the optimal solution. In order to improve the cost, we would like to remove the edges which have higher cost. Given a high cost edge, our goal is to reduce the cost of the edge by minimal perturbation of the filter coefficients. Figure 6 shows a MST for a graph with 13 vertices. The MST shown is divided into two sub-trees T_1 and T_2. Let us suppose that the edge connecting c_j to c_k is the most expensive edge in the MST. Then, our goal is to perturb the value of c_k by the smallest possible amount such that we can obtain another edge of smallest cost which connects T_1 to T_2. That is, for all $c_i \in T_2$, we would like the cost of $(c_k+\delta)-c_i$ to be smallest, where δ is the perturbation value added to c_k. Note that one could not search for a $c_i \in T_1$ since it would not connect T_1 to T_2 although c_k can be reached from any vertex in T_1. Further note that if one considers connecting T_1 to T_2 by selecting an edge that would connect at $c_i \in T_1$ to a $c_l \in T_2$, where $l \neq k$, then we still cannot reach c_k because its value has been perturbed and it is no longer reachable from vertices in T_2 through e_{k1} and e_{k2}. That is, after perturbing c_k, the edges e_{k1} and e_{k2} shown in the figure are no longer valid. Hence, the vertices reached by c_k need new connectivity in order to be reachable.

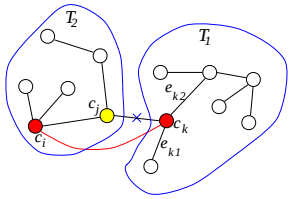

Fig. 6. Improving the MST solution by perturbation of coefficient.

Our approach is to compute the cost reduction of the edge connecting T_1 to T_2 due to perturbation using the following procedure. We compute the cost reduction due to replacing the edge between c_j and c_k by the edge between c_i and c_k. Next, we add to this any increase in the cost of new edges which replace e_{k1} and e_{k2}. If the perturbation is justifiable, we accept it and update the value of c_k by δ. Otherwise, we choose another vertex for perturbation. Note that the simplest vertices to perturb are the ones which are the *leaf vertices* of the MST since we replace one connection with another without changing the remaining part of the MST. In this paper, we only consider the simple perturbation scheme based on the greedy approach described above. In each iteration we select the most expensive available edge for coefficient perturbation. We next show that even such a simple scheme yields dramatic improvement in filter complexity reduction.

Function	A_s (dB)	Tolerance (dB)	M	Type	Name
Low Pass	-45	± 4	25	PM	A
			35	LS	B
Low Pass	-60	± 3	67	PM	C
			111	LS	D
Band Pass	-30	± 4	251	PM	E
			251	LS	F
High Pass	-28	± 3	251	PM	G
			251	LS	H

TABLE I
DESCRIPTION OF EXAMPLE FILTERS

VII. NUMERICAL RESULTS

In this section, we will demonstrate the power and potential of the proposed ODCMI and QODCMI approaches. The original coefficient values, expressed in W-bits, will be referred to as *unscaled* coefficients. We will also consider scaled coefficients where each coefficient c_i is expressed as $c_i' \times 2^{-k}$, such that $k \geq 0$ and $c_i' \geq 2^{W-2}$. This scheme will be referred to as *maximally scaled* coefficients. Scaling is routinely employed in all DSP processors and significantly reduces the quantization noise. (Note that this operation also remove any correlation which is present between successive coefficients.)

The example filters considered are described in table I in which A_s represents the peak stop-band attenuation and *tolerance* values show the maximum allowed deviation from the original filter response. The lower bound on the frequency response in the stop band was allowed to be zero. The passband gain is unity in all examples. Figures 7 and 8 shows the number of adders in the multiplication and overhead add networks for the original, 01-hybrid ODCMI and 01-hybrid QODCMI filters, respectively. Note that filter C for $W = 8$ could not be computed since the original filter quantized and maximally scaled to $W = 8$ did not meet the specifications itself. It is clearly seen that the ODCMI filter greatly reduces the required resources of the parallel filter implementation. We note that the conventional approaches in literature show more than 2 adders per coefficient and only for small filter lengths with small word-lengths [3], [4], [5], [6]. The proposed approaches, in contrast, provide solutions for filters with large M and W. It is worth mentioning that all the presented solutions were obtained within a few minutes of CPU time on a SUN Ultra Sparc station.

Most notably, the complexity of the 01-hybrid QODCMI filter is dramatically lower than the complexity of the original filter. In fact, less than 1 adder per coefficient is required for $W = 8$. Even for higher values of W, a fraction of adders is required compared to the original filter implementation using the proposed perturbation approach. This tremendous reduction clearly indicates the potential of the proposed design methodology. Notice that even for high word-lengths, the implementation obtained using 01-hybrid QODCMI are very simple. Further reductions in complexity are possible if the constraints on frequency domain characteristics are further relaxed. Low-complexity not only yields a high speed design, but also reduces the power consumption because of smaller number of energy consuming operations. One could also use voltage scaling to further reduce the power dissipation for any desired performance. Hence, the novel approach presented in this paper can prove to be a useful compliment to the existing approaches for low-complexity design for high-performance and low power.

VIII. CONCLUSION

We presented novel design methodologies which can be used to dramatically reduce the complexity of parallel implementations of digital FIR filters. These approaches are also applicable to IIR filters. Two ideas were presented. First, we remove the redundant computation by using a graph theoretic frame-work in which we find optimal re-ordering of computation for *maximal computation sharing*. Second, we presented the novel approach of searching for *quantization which improves computation sharing*, when the frequency domain transfer function is allowed to deviate within given bounds. A simple search scheme was presented and it was shown that by appropriate perturbation of filter coefficients, one can dramatically reduce the number of adders required in the filter implementation. Using these approaches, on an average, less than 1 adder per coefficient is required in contrast to a full width multiplier. Hence, the presented methodologies are a useful compliment to the existing design approaches of high-performance and low-power digital filters for future mobile computing and communication systems.

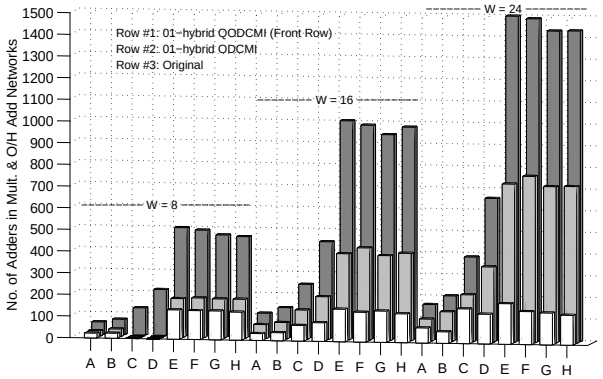

Fig. 7. Comparison of ODCMI and QODCMI for SM representation.

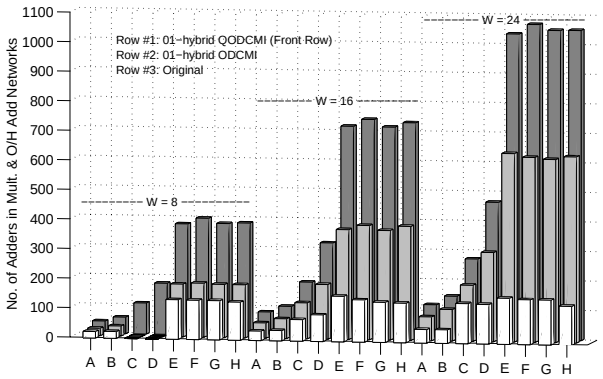

Fig. 8. Comparison of ODCMI and QODCMI for SPT representation.

REFERENCES

[1] S. Haykin, "Adaptive Filter Theory," Prentice Hall, NJ, 1996.

[2] R. Jain, P. T. Yang and T. Yoshino, "FIRGEN: A computer-aided design system for high performance FIR filter integrated circuits," *IEEE Transactions on Signal Processing*, Vol. 39, No. 7, pp. 1655–1668, Jul. 1991.

[3] D. Li. and Y. C. Lim, "Multiplierless realization of adaptive filters by nonuniform quantization of input signal," *1994 IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 457-459, 1994.

[4] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits and Systems*, Vol. 36, No. 7, pp. 1044-1047, July 1989.

[5] B. R. Horng, H. Samueli and A. N. Wilson, "The Design of Low-complexity in Linear-phase FIR Filter Banks Using Powers-of-Two Coefficien ts with an Application to Subband Image Coding," *IEEE Trans. Circuits Syst. Video Technology*, Vol. 1, No. 4, pp. 318–324, Dec. 1991.

[6] M. Yagyu, A. Nishihara and N. Fujii, "Fast FIR Digital Filter Structures Using Minimal Number of Adders and its Application to Filter Design," *IEICE Trans. Fundamentals*, Vol. E79-A, No. 8, pp. 1120–1128, Aug. 1996.

[7] K. Muhammad and K. Roy, "Very low-complexity Digital Filters Based on Computational Redundancy Reduction," Purdue University, West Lafayette, IN, Technical Report, TR-ECE 99-5, Feb. 1999.

[8] N. Sankarayya, K. Roy and D. Bhattacharya, "Algorithms for Low Power and High Speed FIR Filter Realization Using Differential Coefficients," *IEEE Trans. Circuits and Systems*, Vol. 44, No. 6, pp. 488–497, Jun. 1997.

[9] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms," The MIT Press, 1990.