A SPT treatment to the Bit Serial Realization of the Sign-LMS based Adaptive Filter

Sunav Choudhary, Pritam Mukherjee, Mrityunjoy Chakraborty*
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur, INDIA
E.Mail* : mrityun@ece.iitkgp.ernet.in

***Abstract*—This paper presents a bit serial realization of the sign-LMS based adaptive filter which enjoys multiplier free weight update loop. To reduce the complexity of the multipliers that arise in the filtering process, the filter weights are represented in the so-called canonic SPT form which guarantees presence of at least one zero between every two non-zero power-of-two terms. As the filter weights are not fixed but updated in time, it is essential to ensure that the canonic SPT format is retained in the updated filter coefficients. For this, a bit serial adder is proposed that takes as input two numbers in canonic SPT and produces an output also in canonic SPT. It is further shown how the canonic SPT property of the input can be used to reduce the complexity of the adder. For the filtering part, a bit serial multiplier is developed that takes one input (i.e., data bits) in 2's complement form and the other input (i.e., weight bits) in canonic SPT, producing the result in 2's complement. The multiplication can not, however, be realized using a few *fixed* shift and add operations, since the position of the non-zero SPT terms in the canonic SPT expression of each coefficient changes with time. The proposed multiplier instead multiplies the 2's complement number with pairs of consecutive SPT bits of the other number. The resulting partial products can be realized using simple AND-OR logic.**

I. Introduction

In a digital filter, the complexity of realization, both in terms of silicon area and time, is determined primarily by the multipliers. Consequently, efforts have been made to design filters that are free of multipliers. Ideally, if a multiplier coefficient is replaced by a single signed power of two term, the complexity reduces enormously since multiplication by a power of two amounts to a simple shift operation. However, the coefficient quantization error in such cases can be substantial. A more effective approach for this would be to approximate each multiplier coefficient by a sum of (signed) power of two (SPT) while keeping the number of power of two terms as few as possible. A well known sparse SPT representation in this context is the so-called canonic SPT [1]. Under this, a coefficient, say, w is represented as,

$$w \;=\; \sum_{r=1}^{R} s(r)\; 2^{g(r)}, \tag{1}$$

where $s(r) \in \{1,\, -1,\, 0\}$ is the r-th SPT coefficient, $g(r)$ is an increasing sequence of integers and R is the number of terms specified a priori. In canonic SPT, no two consecutive terms are non-zero (i.e., ± 1) simultaneously, i.e., if for any r, $s(r) \;=\; \pm 1$, then both $s(r+1)$ and $s(r-1)$ must be zero (for example, $11 = 2^4 - 2^2 - 2^0$, $19 = 2^4 + 2^2 - 2^0$ etc.). In other words, the canonic SPT guarantees that at least $\lfloor \frac{R}{2} \rfloor$ SPT coefficients in (1) are zero[1] ($\lfloor . \rfloor$ denotes floor operation). A circuit to convert 2's complement numbers into canonic SPT, both in bit serial and parallel forms, has been presented in [1].

The SPT format has been used widely by researchers over years for efficient realization of fixed coefficient digital filters ([4]-[10]). The proposed algorithms are, however, offline techniques which result in fixed SPT format for each filter coefficient. Noting the position of the non-zero SPT terms, circuitry can then be developed to implement the corresponding shifts. Such a treatment, however, is not possible in the context of adaptive filters, due to the following reasons :

- The coefficients in an adaptive filter are continuously updated and thus offline techniques producing fixed SPT representation of the coefficients is not possible.
- As the coefficients change continuously, the positions of the non-zero SPT terms also change, meaning it is not possible to design fixed circuits that implement the corresponding shifts.

This paper addresses the above issues, in the context of the well known sign-LMS algorithm [11]. In a sign-LMS based adaptive filter that takes $x(l)$ as input and $d(l)$ as the desired response, a p-th order filter weight vector $\mathbf{w}(l) = [w_0(l), w_1(l), \cdots, w_{p-1}(l)]^t$ is updated as, $\mathbf{w}(l+1) = \mathbf{w}(l) + \mu\; sign(\mathbf{x}(l))\; sign(e(l))$, where $\mathbf{x}(l) = [x(l), x(l-1), \cdots, x(l-p+1)]^t$, $e(l) = d(l) - \mathbf{w}^t(l)\mathbf{x}(l)$ is the output error and μ is the algorithm step size. As is easily seen, the sign-LMS algorithm, unlike its general LMS counterpart, is free of the multiplication term $\mathbf{x}(l)e(l)$ in the weight update loop and the only multiplication involved occurs in the filtering process, i.e., in the computation of $\mathbf{w}^t(l)\mathbf{x}(l)$. To reduce the complexity of this multiplication, the proposed algorithm takes each filter coefficient $w_j(l),\; j = 0, 1, \cdots, p-1$ in canonic SPT format (to be adjusted at each index l). Similarly, the step size μ is also represented in canonic SPT format which can be pre-computed offline, as the value of μ is known and fixed a priori. The major problem in the proposed SPT realization is then to ensure that the canonic SPT form for each $w_j(l)$ remains invariant to the time update process, i.e., the addition $w_j(l) \pm \mu\; (= w_j(l+1))$ produces a result also in canonic SPT. Towards this, an algorithm is developed in section II

[1] Alternatively, one can view the canonic SPT as a special case of hybrid-2, redundant arithmetic [3] based representation where no two successive bits are allowed to be non-zero simultaneously

for bit serial addition of two numbers in canonic SPT form. The bit serial architecture is preferred over bit parallel or word serial structures as the former is much more hardware efficient than the latter. It is also shown how the canonic SPT form of the input results in logical blocks with lesser complexity in the proposed SPT adder. The other major problem is the bit serial realization of the filtering part, where the multipliers need to make adjustments to take care of the continuously changing position of the nonzero SPT terms in the canonic SPT expansion of each filter coefficient. This is discussed in section III.

II. An Algorithm for Addition of SPT and 2's Complement Numbers

Let the two numbers which are to be added be $a = a_N a_{N-1} \cdots a_1 a_0 \ (\equiv \sum_{i=0}^{N-1} a_i\, 2^i)$ and $b = b_N b_{N-1} \cdots b_1 b_0 \ (\equiv \sum_{j=0}^{N-1} b_j\, 2^j)$ represented in canonic SPT forms, i.e., $a_i,\ b_j \in \{1, -1, 0\}$, with no two successive a_i's and b_j's taking non-zero values. In the proposed scheme, in the i-th cycle, we add a_i, b_i and the incoming carry c_i generated in the $(i-1)$-th cycle, and produce the new carry c_{i+1} and an intermediate result sp_i, which is to be adjusted to the final value s_i in the $(i+1)$-th clock cycle. In other words, in the proposed scheme, there is a latency of one cycle between the i-th cycle input and the corresponding output. The proposed algorithm is given below where we use the notation 1^* to denote ± 1.

Algorithm : *Given a_i, b_i, c_i and sp_{i-1}, carry out the following steps at the i-th cycle :*
Step 1 (Addition) : Add a_i, b_i, c_i to produce c_{i+1} and sp_i.
Step 2 (Adjustment) : For adjustment, we utilize the following identities : $2^i + 2^{i-1} = 2^{i+1} - 2^{i-1}$ and $2^i - 2^{i-1} = 2^{i-1}$.
• *If $sp_i = 1^*$ and $sp_{i-1} = -1^*$, then adjust sp_i to 0 and take $s_{i-1} = 1^*$ as the output (of the previous cycle).*
• *If $sp_i = sp_{i-1} = 1^*$, then take $s_{i-1} = -1^*$, adjust sp_i to 0 and propagate 1^* to the $(i+1)$-th step as c_{i+1}* [Note that for this case, c_{i+1} from Step 1 can not be 1^*, since this would imply that all the three bits, a_i, b_i and c_i are 1^* each simultaneously, which is, however, not possible, as shown in Lemma 1 below].
• No adjustment needed otherwise, meaning $sp_{i-1} \rightarrow s_{i-1}$.

As seen above, the four bits, a_i, b_i, c_i and sp_{i-1} are used to generate s_{i-1} and c_{i+1}. Theoretically, these four bits can have a total of $3^4 = 81$ combinations. However, as shown by Lemmas 1-3 below, only a fraction of these combinations are feasible while the remaining ones can not come up. *This results in considerable savings in hardware as one can use the so-called "don't care" states for the invalid combinations*.

Lemma 1 : *The three bits, a_i, b_i and c_i can not be non-zero simultaneously.*
Proof : Suppose that the three bits, a_i, b_i and c_i are non-zero simultaneously. From the characteristics of the canonic SPT format, this implies that $a_{i-1} = 0$ and $b_{i-1} = 0$. The only possible way to maintain c_i non-zero in this case is to have $c_{i-1} = 1^*$ and $sp_{i-2} = 1^*$ (in the $(i-1)$-th cycle), which would lead to the following adjustments/assignments, as per the algorithm above : $sp_{i-1} \rightarrow 0$, $sp_{i-2} \rightarrow s_{i-2} = -1^*$ and $c_i = 1^*$. The combination, $c_{i-1} = 1^*$ and $sp_{i-2} = 1^*$ can, however, occur only when $a_{i-2} = 1^*$, $b_{i-2} = 1^*$ and $c_{i-2} = 1^*$, i.e., all the three bits, a_{i-2}, b_{i-2} and c_{i-2} are non-zero. Proceeding recursively, for i even, this would then mean that the bits, a_0, b_0 and c_0 are non-zero simultaneously, which is, however, not possible, since, in the proposed scheme, we always have $c_0 = 0$. Again, for i odd, the above means a_1, b_1 and c_1 are non-zero simultaneously. However, c_1 can not be non-zero, since, from the canonic SPT property, we have, in this case, $a_0 = 0$, $b_0 = 0$ and separately, $c_0 = 0$. Hence proved.

Lemma 2 : *If exactly one of the four bits, a_i, b_i, c_i and sp_{i-1} is zero, then it has to be c_i.*
Proof : Suppose, $a_i = 0$ and $b_i \neq 0$, $c_i \neq 0$ and $sp_{i-1} \neq 0$. From the canonic SPT property, it then follows that $b_i = 0$. To have $sp_{i-1} \neq 0$, one of the two bits, a_{i-1} and c_{i-1} must be non-zero, which, however, implies $c_i = 0$ and thus a contradiction. Same logic applies to the case where $b_i = 0$ and the remaining three bits are non-zero. Again, if $sp_{i-1} = 0$, we have, a_i, b_i and c_i non-zero simultaneously, which is not permitted as per Lemma 1. Hence, the only possibility is $c_i = 0$.

Lemma 3 : *If exactly two of the four bits, a_i, b_i, c_i and sp_{i-1} are zero, then at least one of them has to be c_i or sp_{i-1}.*
Proof : Suppose $a_i = b_i = 0$ and $c_i \neq 0$, $sp_{i-1} \neq 0$. In this case, to maintain $c_i \neq 0$, $sp_{i-1} \neq 0$, all the three bits, a_{i-1}, b_{i-1} and c_{i-1} have to be non-zero simultaneously which is, however, not permissible as per Lemma 1. Hence proved.

Circuit Implementation : As seen above, the algorithm is simply a rule to transform the pair $(c_i + a_i + b_i,\ sp_{i-1})$ to the triplet (c_{i+1}, sp_i, s_{i-1}). Define three functions, $f_c(a_i, b_i, c_i, sp_{i-1}), f_{sp}(a_i, b_i, c_i, sp_{i-1})$ and $f_s(a_i, b_i, c_i, sp_{i-1})$ which generate the quantities $c_{i+1},\ sp_i$ and s_{i-1} respectively at the i-th cycle following the above algorithm. The truth table for each function is easy to generate. However, out of a total of $3^4 = 81$ possible combinations in each truth table, *the Lemmas 1-3 can be used to reduce the number of combinations to just 37.*

The corresponding block diagram for hardware realization of the bit serial adder is shown in Fig. 1. Here f_c, f_{sp} and f_s are combinatorial blocks which implement the respective truth tables. Note that since the SPT representation uses altogether three bits, namely, 1, -1 and 0, in a digital implementation, each SPT bit is represented by two binary bits, as per the following : $(1)_{SPT} = (01)_2$, $(0)_{SPT} = (00)_2$ and $(\bar{1})_{SPT} = (11)_2$, which is consistent with 2's complement representation of signed binary numbers.

III. The proposed Bit Serial Implementation

Assume that the filter weights at the l-th index, $w_j(l),\ \ j = 0, 1, \cdots, p-1$ and the step size μ

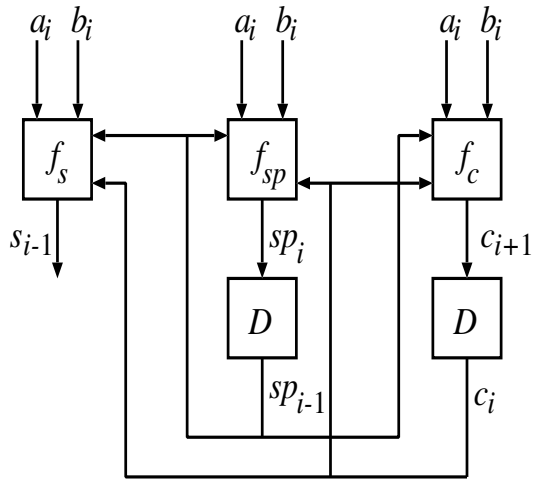

Fig. 1. Block Diagram for Bit-Serial Addition of Numbers in Canonic SPT form.

have the following canonic SPT representations : $w_j(l) = \sum_{i=0}^{N-1} w_{i,j}(l)2^{-i}$ and $\mu = \sum_{i=0}^{N-1} \mu_i 2^{-i}$, with $w_{i,j}(l),\ \mu_i \in \{1, -1, 0\}$. The weight update block can then be implemented by feeding the bit serial adder of Fig. 1 with the inputs $w_{i,j}(l)$ and μ_i serially over $i = 0, 1, \cdots N-1$, producing the sequence $w_{i,j}(l+1)$ at the output with a latency of one bit cycle. For the filtering part, the main job is to compute products like $w_j(l)x(l-j),\ \ j = 0, 1, \cdots, p-1$. Assume that each $x(l-j)$ is given by a N bit word in 2's complement form as, $x(l-j) \ : \ x_{N-1,l-j}\, x_{N-2,l-j} \cdots x_{1,l-j}\, x_{0,l-j}$, with the bits fed to the system serially. Then, from the canonic SPT nature of $w_j(l)$, the product $w_j(l)x(l-j)$ would involve just a few shifts and additions of the word $x_{N-1,l-j}\, x_{N-2,l-j} \cdots x_{1,l-j}\, x_{0,l-j}$, with the amount of shift determined by the position of non-zero SPT terms in $w_j(l)$. However, as the SPT expression of $w_j(l)$ changes from index to index, the shift values also change and it is not possible to design circuits that can implement such time varying shift operations. Instead, a better approach would be to consider bit pairs for each filter weight like $\{w_{0,j}(l), w_{1,j}(l)\},\ \{w_{2,j}(l), w_{3,j}(l)\}$ etc. and observing that at the most one bit in each bit pair can take non-zero values, develop circuits to implement the action of each bit pair on the data word. For this, we propose a bit serial multiplier below which multiplies a canonic SPT number with a 2's complement number, generating the output in 2's complement form.

A Bit Serial Multiplier with One Input in Canonic SPT : We explain the functioning of the proposed multiplier by considering the case of $N = 4$ (the arguments that we use here can be easily extended to the general case). In any l-th word cycle, the four bits of the 2's complement number, say, $a_3a_2a_1a_0$ enter the system at $4l+0,\ 1,\ 2,\ 3$ cycles with LSB, i.e., a_0 first (for convenience, we have dropped the index l from the bits $a_k,\ k = 0, 1, 2, 3$). The other input to the multiplier is a number in a 4 bit canonic SPT format, $b'_3b'_2b'_1b'_0$, where each $b'_k,\ k = 0, 1, 2, 3$ is represented by two binary bits as explained above, namely, $b'_k = s_k\, b_k$. The bits s_k and b_k can be taken as the sign bit and the magnitude bit respectively of b'_k, both of which enter the multiplier serially. We form the bit pairs : $\{b_1,\, b_0\},\ \{b_3,\, b_2\}$ and note that in each bit pair, at the most one bit can be binary 1 and other bit(s) is binary 0. Multiplying the input word $a_3a_2a_1a_0$ by the bit pair $\{b_1,\, b_0\}$, discarding the LSB : b_0a_0 and carrying out the sign extension $a_3b_0 \rightarrow a_3b_0$, the four bit output of this multiplication, $pp_1^3pp_1^2pp_1^1pp_1^0$ is obtained as : $pp_1^i = a_i.b_1 + a_{i+1}.b_0$ for $i = 0,\ 1,\ 2$ and for $i = 3,\ pp_1^3 = a_3.b_1 + a_3.b_0$ [the '.' and '+' operations indicate logical AND and OR operations respectively]. Note that the above result can be negative or positive depending on whether any of the sign bits, s_1 or s_2 is 1 or not. However, if negative, we do not explicitly complement the above output, but instead, achieve the same by using the bits s_1 and s_2 as control inputs to an adder-subtractor, as discussed later. Multiplication of the input word with the bit pair $\{b_3,\, b_2\}$ is also carried out in the same way, producing $pp_2^3pp_2^2pp_2^1pp_2^0$. The two intermediate results are to be added (subtracted) with proper alignment. For this, we discard $pp_1^1pp_1^0$, do the sign extension $pp_1^3 \rightarrow pp_1^3 \rightarrow pp_1^3$, and add $pp_2^3pp_2^2pp_2^1pp_2^0$ with $pp_1^3pp_1^3pp_1^3pp_1^2$. The result $y_3y_2y_1y_0$ comes out serially (y_0 first) at the indices $4l+3, 4, 5, 6$. The proposed multiplier is

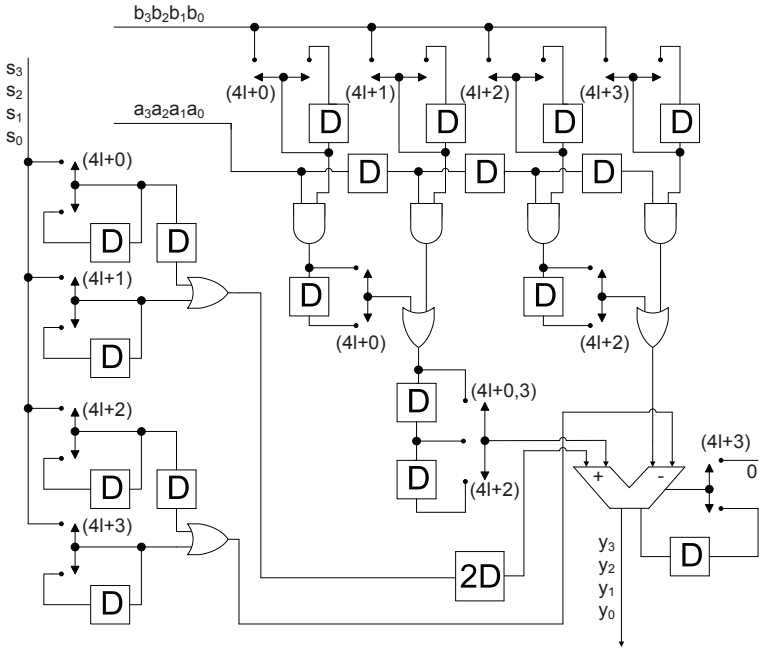

Fig. 2. Proposed bit serial multiplier (for four bit multiplicands).

shown in Fig. 2. Using switches, the four bits $b_0,\ b_1,\ b_2$ and b_3 are held in delay loops for four cycles, starting respectively at the following indices : $4l+0, 1, 2, 3$. Same is done separately to the four sign bits, $s_0,\ s_1,\ s_2$ and s_3. For sign extension, we feed the bit to be extended to a delay and move a switch from input of the delay to the output in the next clock cycle. Also, for convenience, we follow the modular notation for timing instances here, i.e., $4l+0 \equiv 4l+4,\ 4l+1 \equiv 4l+5$ and $4l+2 \equiv 4l+6$. It is then easy to verify that one input

of the adder/subtractor (marked with a '+' sign) receives the bits, pp_1^2, pp_1^3, pp_1^3, pp_1^3 at the following cycles : $4l+3, 0, 1, 2$ respectively. During the same time, the other input (marked with a '-' sign) receives the bits, pp_2^0, pp_2^1, pp_2^2, pp_2^3. It is also easy to verify that the control bit at the '+' input of the adder/subtractor at cycles $4l+3, 0, 1, 2$ is given by $s_0 + s_1$, while the same at the '-' input is given by $s_2 + s_3$.

As seen above, the output is available from the cycle $4l + 3$ onwards. In general, if two N bit numbers are multiplied, the output will be available bit serially over the cycles, $lN+N-1, 0, 1, \cdots, N-2$. Each output will actually provide the result of the multiplication $w_j(l)x(l-j),\ j = 0, 1, \cdots, p-1$. These are to be added and then subtracted from the desired response $d(l)$, all bit serially to yield $e(l)$. However, to maintain correctness, the bit sequence of $d(l)$ is to be delayed by $(N-1)$ cycles. Also note that the sign bit of $e(l)$ which is needed in computing the update term $\mu\ sign(\mathbf{x}(l))\ sign(e(l))$ of the sign-LMS algorithm will be available only at the cycle $(lN + 2N - 2)$ while the next, i.e., $(l+1)$-th data bits enter the system from cycle $(lN + N)$ onwards, meaning the updated weight bits will be required from cycle $(lN + N)$. To overcome this problem, we take recourse to the delayed mode of adaptation [12]. In particular, we delay the update term in the weight update equation by one data cycle, resulting in the update equation, $\mathbf{w}(l+1) = \mathbf{w}(l) + \mu\ sign(\mathbf{x}(l-1))\ sign(e(l-1))$. This means the sign bit of $e(l-1)$, available at cycle $lN+N-2$ will be fed back to the bit serial adder, along with the available/stored sign bits of $x(l-1-j)$, to determine the sign of μ (i.e., $\pm$) in the update term. There is a gap of two cycles between the time when the sign bit of $e(l-1)$ becomes available (i.e., $lN+N-2$) and the time (i.e., $lN+N$) when the updated weight bits are generated. Of these two cycles, one is consumed by the bit serial adder which has a latency of one cycle.

It may be noted that in the above implementation, we have considered the minimum unavoidable delay of one data cycle in the delayed mode of coefficient adaptation. However, by increasing the delay further [12] and noting that one word cycle delay actually amounts to N bit cycle delays, it is possible to pipeline the various operations in the weight update loop (e.g., addition of the multiplier outputs to form $y(n)$, subtraction of $y(n)$ from $d(n)$ to generate $e(n)$ etc.) to reduce the critical path.

IV. Discussions and Conclusion

In this paper, we have presented a bit-serial implementation of the sign-LMS based adaptive filter. The sign-LMS algorithm requires only 50 % of the total number of multipliers as needed in the conventional LMS algorithm. In order to reduce the complexity of these multiplications which arise in the filtering part, the filter weights are represented at each index in the so-called canonic SPT format, which guarantees presence of at least one zero between every two consecutive signed power of two terms. A major concern in choosing this format for the filter weights has been to ensure that the format remains invariant to the weight updating process. Towards this, we have proposed a bit serial adder that takes as input two numbers in canonic SPT form and produces an output also in canonic SPT. Further, it is shown how the canonic SPT property of the input can be exploited to reduce the complexity of the adder. To carry out the filtering part, a bit serial multiplier is developed that takes one input (i.e., data bits) in 2's complement form and the other input (i.e., weight bits) in canonic SPT, producing the result in 2's complement. However, as the filter weights keep changing from index to index, the positions of the non-zero power of two terms in the canonic SPT expression of each coefficient also change with time and obviously, the corresponding time varying shift operations can not be realized by fixed hardware circuits. The proposed multiplier takes care of this by dividing the number into non-overlapping groups of two consecutive SPT bits each and then by observing that in each group, at the most one SPT term can be non-zero (i.e., ± 1). The partial product resulting from the multiplication of the 2's complement number by each group can then be realized using simple AND-OR logic rather than a full fledged full adder. Finally, the bit serial processing has an inherent latency which makes it necessary to employ delayed mode of coefficient adaptation. In the present treatment, we have considered delay by one data cycle only, which is the minimum unavoidable delay. However, the delay can be increased further and the resulting N fold increase in the bit cycle delays can be used to pipeline the weight update loop.

References

[1] Y.C. Lim, J.B. Evans and B. Liu, "Decomposition of Binary Integers into Signed Power-of-Two Terms", *IEEE Trans. Circuits and Systems*, vol. 38, no. 6, pp. 667-672, 1991.

[2] Y.C. Lim, R. Yang, D. Li and J. Song, "Signed power-of-two term allocation scheme for the design of digital filters", *IEEE Trans. Circuits and Systems, part II*, vol. 46, no. 5, pp. 577-584, 1999.

[3] K.K. Parhi, "VLSI Digital Signal Processing Systems - Design and Implementation", John Wiley and Sons Inc., 1999.

[4] H.H. Dam, A. Cantoni, K.L. Teo and S. Nordholm, "FIR Variable Digital Filter With Signed Power-of-Two Coefficients", *IEEE Trans. Circuits and Systems, Part I*, vol. 54, no. 6, pp. 1348-1357, 2007.

[5] S.Y. Park, N.I. Cho, "Design of Multiplierless Lattice QMF: Structure and Algorithm Development", IEEE Trans. Circuits and Systems, part II, vol. 55, no. 2, pp. 173-177, 2008.

[6] Z.G. Feng, K.L. Teo, "A Discrete Filled Function Method for the Design of FIR Filters With Signed-Powers-of-Two Coefficients", *IEEE Trans. Signal Processing*, vol. 56, no. 1, pp. 134-139, 2008.

[7] M. Aktan., A. Yurdakul and G. Dundar, "An Algorithm for the Design of Low-Power Hardware-Efficient FIR Filters", *IEEE Trans. Circuits and Systems part I*, vol. 55, no. 6, pp. 1536-1545, 2008.

[8] Y.J. Yu and Y.C. Lim, "Design of Linear Phase FIR Filters in Subexpression Space Using Mixed Integer Linear Programming", *IEEE Trans. Circuits and Systems part I*, vol. 54, no. 10, pp. 2330-2338, 2007.

[9] F. Xu, C.H. Chang and C.C. Jong, "Design of Low-Complexity FIR Filters Based on Signed-Powers-of-Two Coefficients With Reusable Common Subexpressions", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1898-1907, 2007.

[10] S.Y. Park and N.I. Cho, "Design of signed powers-of-two coefficient perfect reconstruction QMF Bank using CORDIC algorithms", *IEEE Trans. Circuits and Systems part I*, vol. 53, no. 6, pp. 1254-1264, 2007.

[11] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1986

[12] G. Long, F. Ling and J.G. Proakis, "The LMS Algorithm with Delayed Coefficient adaptation", *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 9, pp. 1397-1405, sept., 1989.

[13] S. Choudhary, P. Mukherjee and M. Chakraborty, "An Algorithm for Bit-Serial addition of Numbers in Canonic SPT Format", S. Choudhary, P. Mukherjee and M. Chakraborty, under review for publication in *Electronic Letters.*