Low-power Adaptive Filter Architectures via Strength Reduction

Manish Goel and Naresh R. Shanbhag
Coordinated Science Lab./ECE Department,
Univ. of Illinois at Urbana-Champaign, Urbana IL-61801
E-mail : [mgoel,shanbhag]@uivlsi.csl.uiuc.edu

Abstract

Low-power and high-speed algorithms and architectures for complex adaptive filters are presented in this paper. These architectures have been derived via the application of algebraic and algorithm transformations. The strength reduction transformation when applied at the *algorithmic level* results in a power reduction by 21% as compared to the traditional cross-coupled structure. A fine-grain pipelined architecture is then developed via the relaxed look-ahead transformation. The pipelined architecture allows high-speed operation with minimum overhead and when combined with power-supply reduction enables additional power-savings of 40-69%. Thus, an overall power-saving of 60-90% over the traditional cross-coupled architecture is achieved.

1 Introduction

Power-reduction techniques [2, 3, 4] have been proposed at all levels of design hierarchy begining with algorithms and architectures and ending with circuits and technological innovations. It is now well recognized that an astute algorithmic and architectural design can have a large impact on the final power dissipation characteristics of the fabricated VLSI solution. In this paper, we will investigate algorithms and architectures for low-power and high-speed adaptive filters.

Algorithm transformation techniques [3] such as look-ahead [6], relaxed look-ahead [8], block-processing, associativity [7] have been employed to design high-speed algorithms and architectures. Low-power operation is then achieved by trading off excess speed with power. Of particular interest is a class of transformations known as algebraic transformations [7]. Strength reduction [2] is an algebraic transformation, which has been applied at the architectural level to trade-off expensive multipliers with adders. This results in an overall savings in area and power. A key contribution of this paper is the application of the strength reduction transformation at the *algorithmic level* (instead of architectural level) to obtain low-power adaptive filter algorithms. An algorithmic level application of strength reduction is shown to be more effective in achieving power reduction as compared to an architectural level application.

The application of strength reduction increases the critical path computation time. This results in a throughput limitation, which is undesirable in high- bit rate applications. We address this problem with relaxed look-ahead [8] transformation. This transformation results in a fine-grain pipelined architecture, which is an approximation of the architecture obtained by look-ahead technique. The relaxed look-ahead technique maintains the functionality of the algorithm rather than the input-output behaviour. Furthermore, it is possible to trade off some of the increased throughput for reduced power dissipation via power supply reduction as indicated in [3].

2 Preliminaries

In this section, we will review some of the basics of strength reduction and relaxed look-ahead pipelining.

2.1 Algebraic Transformation

Algebraic transformations are an important class of architectural level transformations, which have been proposed for low-power [2] and high-speed [7] DSP algorithms. The strength reduction transformation trades off high-complexity multiply operations with low-complexity add operations thus achieving low-power.

Consider the problem of computing the product of two complex numbers $(a + jb)$ and $(c + jd)$ as shown below

$$(a + jb)(c + jd) = (ac - bd) + j(ad + bc) \qquad (2.1)$$

We observe that a total of four real multiplications and two real additions are needed for computing a complex product. However, it is possible to reduce this complexity via strength reduction [1, 2]. This requires reformulating (2.1) as follows

$$(a - b)d + a(c - d) = ac - bd \qquad (2.2)$$

$$(a - b)d + b(c + d) = ad + bc \qquad (2.3)$$

As can be seen from (2.2)-(2.3), the number of real multiplications is three and the number of additions is five. Therefore, this form of strength reduction transformation reduces the number of multipliers by one at the expense of three additional adders.

If we assume that the effective capacitance of a two-operand multiplier is K_C times that of a two-operand adder, it can be seen that strength reduction results in a power savings factor PS given by

$$PS = \frac{P_{D,o} - P_{D,sr}}{P_{D,o}} = \frac{K_C - 3}{2(2K_C + 1)} \qquad (2.4)$$

where $P_{D,o}$ and $P_{D,sr}$ are the dynamic power dissipation of the original and strength-reduced algorithms. From (2.4), it is clear that for $K_C > 3$, we will achieve power savings. Asymptotically, the power savings approach 25% as K_C increases.

It can be easily seen that the strength reduction transformation increases the critical path computation time, which can be a limitation in the high speed applications. This problem is solved by throughput enhancement techniques such as pipelining as described next.

2.2 Relaxed Look-ahead Pipelining

In this sub-section, we describe relaxed look-ahead [8] technique, which is an approximation to the look-ahead [6] technique. Consider an LMS adaptive filter with a first-order weight-update recursion given by

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00 © 1996

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mu e(n)\mathbf{X}(n) \qquad (2.5)$$
$$e(n) = d(n) - \mathbf{W}^T(n-1)\mathbf{X}(n) \qquad (2.6)$$

where $\mathbf{W}(n)$ is a $N \times 1$ vector of filter coefficients, μ is the adaptation step-size, $e(n)$ is the estimation error, $\mathbf{X}(n)$ is the $N \times 1$ input vector, and $d(n)$ is the desired signal.

A pipelined LMS algorithm can be obtained via relaxed look-ahead transformations described in [8]. The transformed equations are

$$\mathbf{W}(n) = \mathbf{W}(n-D_2) + \mu \sum_{i=0}^{LA-1} e(n-D_1-i) \cdot \mathbf{X}(n-D_1-i) \qquad (2.7)$$
$$e(n) = d(n) - \mathbf{W}^T(n-D_2)\mathbf{X}(n) \qquad (2.8)$$

where LA is the look-ahead factor and D_1 and D_2 are delays introduced via the *delay relaxation* and *sum-relaxation*. These delays can be employed to pipeline the hardware operators in an actual implementation.

In this paper, we will employ the relaxed look-ahead pipelined LMS filter to obtain the pipelined filter architectures. Furthermore, the increased throughput due to pipelining can be employed to achieve high-speed and low-power (in combination with power-supply scaling).

3 Low-Power Adaptive Filter Architecture

In this section, we will develop a low power adaptive filter via strength reduction transformation. We will assume that a passband digital communication system such as quadrature amplitude modulation (QAM) or carrierless amplitude/phase (CAP) modulation [5] is being employed. In this situation, the receiver processes a two-dimensional signal using a two-dimensional filter. This results in the traditional cross-coupled equalizer structure.

3.1 Traditional Cross-coupled Equalizer Architecture

Assume the filter input to be a complex signal $\tilde{\mathbf{X}}$(n) given by

$$\tilde{\mathbf{X}}(n) = \mathbf{X}_r(n) + j\mathbf{X}_i(n) \qquad (3.1)$$

where $\mathbf{X}_r(n)$ and $\mathbf{X}_i(n)$ are real and imaginary parts, respectively. Furthermore, if the filter $\mathbf{W}(n)$ is also complex ($\tilde{\mathbf{W}}(n) = \mathbf{c}(n) + j\mathbf{d}(n)$), then its output $\tilde{y}$(n) can be obtained as follows

$$\begin{aligned}\tilde{y}(n) &= \tilde{\mathbf{W}}^H(n-1)\tilde{\mathbf{X}}(n) \\ &= \left[\mathbf{c}^T(n-1)\mathbf{X}_r(n) + \mathbf{d}^T(n-1)\mathbf{X}_i(n)\right] + \\ &\quad j\left[\mathbf{c}^T(n-1)\mathbf{X}_i(n) - \mathbf{d}^T(n-1)\mathbf{X}_r(n)\right] \\ &= y_r(n) + jy_i(n) \qquad (3.2)\end{aligned}$$

where $\tilde{\mathbf{W}}^H$ represents the Hermitian (transpose and complex conjugate) of the matrix $\tilde{\mathbf{W}}$. A direct implementation of (3.2) results in the traditional cross-coupled structure shown in Fig. 1, which requires $4N-2$ adders and $4N$ multipliers. In the adaptive case, a **WUD**-block would be needed to automatically compute the coefficients of the filter. This can be done as follows

$$\tilde{\mathbf{W}}(n) = \tilde{\mathbf{W}}(n-1) + \mu\tilde{e}^*(n)\tilde{\mathbf{X}}(n) \qquad (3.3)$$

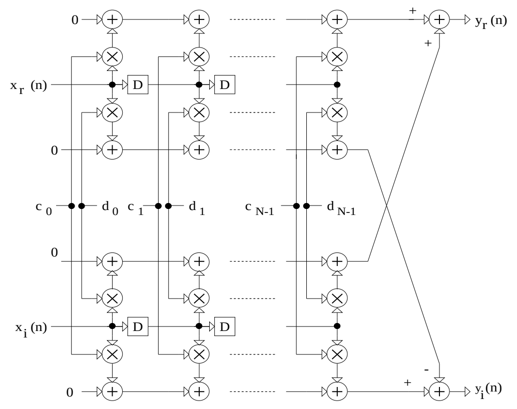

Figure 1: Traditional cross-coupled **F**-block implementation

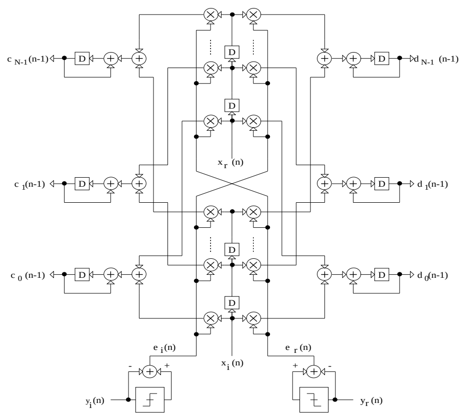

Figure 2: Traditional **WUD**-block implementation

where $\tilde{e}(n) = e_r(n) + je_i(n)$, $e_r(n) = Q[y_r(n)] - y_r(n)$, $e_i(n) = Q[y_i(n)] - y_i(n)$, $Q[\cdot]$ is the output of the slicer, and $\tilde{e}^*$ is the complex conjugate of $\tilde{e}$. Therefore, to implement **WUD**-block, we need the following real equations

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mu\left[e_r(n)\mathbf{X}_r(n) + e_i(n)\mathbf{X}_i(n)\right] \qquad (3.4)$$
$$\mathbf{d}(n) = \mathbf{d}(n-1) + \mu\left[e_r(n)\mathbf{X}_i(n) - e_i(n)\mathbf{X}_r(n)\right] \qquad (3.5)$$

From the **WUD**-block architecture in Fig. 2, it is clear that we require $4N+2$ adders and $4N$ multipliers for an N-tap complex filter. In the next subsection, we will present a low-power adaptive filter using strength reduction.

3.2 Low-Power Adaptive Filter Architecture

It can be easily seen that (3.2) involves multiplication of two complex polynomials. So strength reduction transformation presented in the previous section can be applied to (3.2). Applying the transformation, we obtain

$$\tilde{y}(n) = [y_1(n) + y_3(n)] + j[y_2(n) + y_3(n)] \qquad (3.6)$$

where

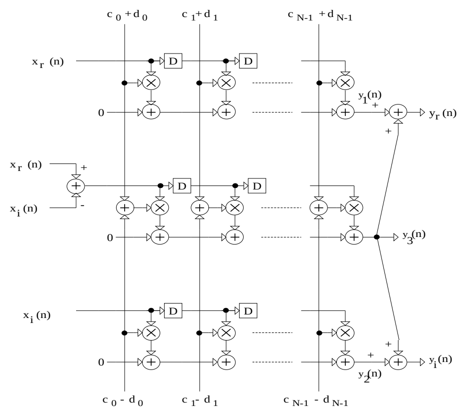

Figure 3: Strength reduced **F**-block implementation

$$y_1(n) = \mathbf{c}_1^T(n-1)\mathbf{X}_r(n) \quad (3.7)$$
$$y_2(n) = \mathbf{d}_1^T(n-1)\mathbf{X}_i(n) \quad (3.8)$$
$$y_3(n) = -\mathbf{d}^T(n-1)\mathbf{X}_1(n) \quad (3.9)$$

and $\mathbf{X}_1(n) = \mathbf{X}_r(n) - \mathbf{X}_i(n)$, $\mathbf{c}_1(n) = \mathbf{c}(n) + \mathbf{d}(n)$, and $\mathbf{d}_1(n) = \mathbf{c}(n) - \mathbf{d}(n)$. The proposed architecture (see Fig. 3) requires three filters and two output adders which corresponds to $3N$ multipliers and $4N$ adders. We now consider the adaptive version and specifically analyze the **WUD**- block. From (3.7)-(3.9) and Fig. 3, it seems an efficient architecture would result if $\mathbf{c}_1$(n) and $\mathbf{d}_1$(n) are adapted instead of $\mathbf{c}$(n) and $\mathbf{d}$(n).

Applying strength reduction transformation to the update equations for $\mathbf{c}_1(n)$ and $\mathbf{d}_1(n)$, we obtain

$$\mathbf{c}_1(n) = \mathbf{c}_1(n-1) + \mu[\mathbf{eX}_1(n) + \mathbf{eX}_3(n)] \quad (3.10)$$
$$\mathbf{d}_1(n) = \mathbf{d}_1(n-1) + \mu[\mathbf{eX}_2(n) + \mathbf{eX}_3(n)] \quad (3.11)$$

where
$$\mathbf{eX}_1(n) = 2e_r(n)\mathbf{X}_i(n) \quad (3.12)$$
$$\mathbf{eX}_2(n) = 2e_i(n)\mathbf{X}_r(n) \quad (3.13)$$
$$\mathbf{eX}_3(n) = e_1(n)\mathbf{X}_1(n) \quad (3.14)$$

and $e_1(n) = e_r(n) - e_i(n)$, $\mathbf{X}_1(n) = \mathbf{X}_r(n) - \mathbf{X}_i(n)$. It can be seen from the architecture of **WUD**-block (Fig. 4) that it requires only $3N$ multipliers and $4N+3$ adders. Combining the architecture for the **F**-block (Fig. 3) and **WUD**-block (Fig. 4), we obtain the proposed strength reduced low-power adaptive filter architecture in Fig. 5.

3.3 Power Savings

Using the definition of PS in (2.4), it can be easily seen that the power savings PS due to the proposed filter architecture is given by

$$PS = \frac{(2NK_C - 3)}{4(2NK_C + 3N)} \quad (3.15)$$

where K_C is the ratio of the effective capacitance of a two-operand multiplier to that of a two-operand **F**-block adder. It can be seen from (3.15) that for the large values of N and K_C, the power savings approach 25%. Even for the typical values of $N = 32$ and $K_C = 8$, the power savings are 21%. It is worth mentioning here that the same strength reduction transformation applied at the architectural level would result in power savings of 9.2% for $K_C = 8$.

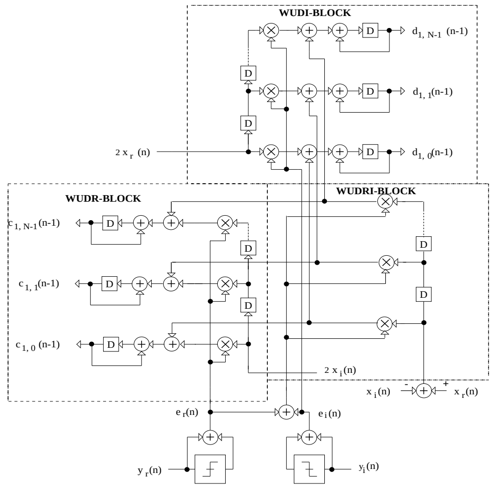

Figure 4: Strength reduced **WUD**-block implementation

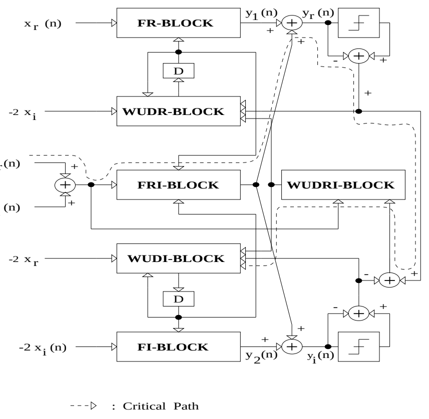

Figure 5: Low-power strength reduced adaptive filter architecture

4 Relaxed Look-ahead Pipelined Equalizer Architecture

In the adaptive case, both cross-coupled and strength reduced architectures have throughput limitation due to the error feedback path. The dotted line in Fig. 5 indicates the critical path. By calculating the computation time for the **F**-blocks from Fig. 3 and for **WUD**-blocks from Fig. 4, we get the lower limit on the clock time for serial strength reduced equalizer architecture (**SEA**)

$$T_{\mathbf{SEA}} \geq 2T_m + (N+7)T_a \quad (4.1)$$

where T_m and T_a are two-operand multiply and single precision add times, respectively. For application that require large values of N, the lower bounds on T_c may prevent a feasible implementation. In this section, we propose a solution to the problem by pipelining the **SEA** and therefore achieving high-speed. Some of the speed will be traded-off with power and thus achieving additional power savings.

4.1 Pipelined Equalizer Architecture (PEA)

In order to derive the **PEA**, we start with **SEA** equations (3.6-3.9, 3.10-3.14) and then apply realxed look-

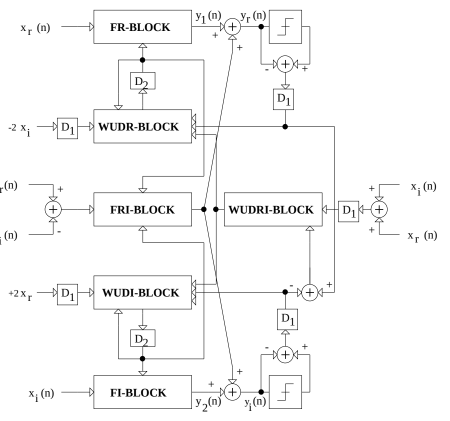

Figure 6: Pipelined strength reduced adaptive filter architecture

ahead. Observe that the equations are similar to the that of the traditional LMS described by (2.5-2.6). By inspection of the relaxed look-ahead pipelined LMS algorithm given by (2.7-2.8), we get the following equations which describe the **PEA**,

$$\tilde{y}(n) = [y_1(n) + y_3(n)] + j[y_2(n) + y_3(n)] \tag{4.2}$$

where

$$\begin{aligned} y_1(n) &= \mathbf{c}_1^T(n - D_2)\mathbf{X}_r(n) && (4.3) \\ y_2(n) &= \mathbf{d}_1^T(n - D_2)\mathbf{X}_i(n) && (4.4) \\ y_3(n) &= -\mathbf{d}^T(n - D_2)\mathbf{X}_1(n) && (4.5) \end{aligned}$$

$$\begin{aligned} \mathbf{c}_1(n) &= \mathbf{c}_1(n - D_2) + \mu \sum_{i=0}^{LA-1} [\mathbf{eX}_1(n - D_1 - i) + \\ &\quad \mathbf{eX}_3(n - D_1 - i)] && (4.6) \\ \mathbf{d}_1(n) &= \mathbf{d}_1(n - D_2) + \mu \sum_{i=0}^{LA-1} [\mathbf{eX}_2(n - D_1 - i) + \\ &\quad \mathbf{eX}_3(n - D_1 - i)] && (4.7) \end{aligned}$$

where $\mathbf{eX}_1(n)$, $\mathbf{eX}_2(n)$ and $\mathbf{eX}_3(n)$ are defined in (3.12-3.14), The block-diagram of **PEA** is shown in Fig. 6. In a practical implementation, D_1 and D_2 delays will be employed to pipeline the **F** and **WUD**-blocks. Thus, all the operations in the **PEA** can be pipelined at a fine-grain level.

Assuming that the algorithmic delays have been retimed in a uniform fashion (i.e., all stages have the same delay), the lower bound on the input sample period $T_{\mathbf{PEA}}$ is given by

$$T_{\mathbf{PEA}} \geq \frac{2T_m + (N + 2LA + 5)T_a}{D_1 + D_2} \tag{4.8}$$

Higher values of D_1 and D_2 imply higher speed-ups. Practical maximum values of D_1 and D_2 are a function of the desired algorithmic performance (i.e. BER and/or SNR at the slicer).

4.2 Power Savings

As mentioned before, the pipelining along with power-supply reduction [2] has been proposed as a technique for reducing the power dissipation. As previously done, we get the power saving PS with respect to the cross-coupled architecture as shown below

$$PS = \frac{2NK_C(4K^2 - 3) + 2N(6K^2 - 2LA - 4) - 3}{4K^2(2NK_C + 3N)} \tag{4.9}$$

where $K > 1$ is the factor by which power supply is scaled down. Employing typical values of $K = 5V/3.3V$, $K_C = 8$, $N = 32$ and $LA = 3$, we obtain a total power savings of approximately 61% over the traditional cross-coupled architecture.

Based upon the transistor threshold voltages, it has been shown [2] that value of $K = 3$ are possible with present CMOS technology. With this value of K, (4.9) predicts a power savings of 90%, which is a significant reduction.

5 Conclusions

Application of strength reduction transformation [1, 2] at the *algorithmic level* (as opposed to the architectural level) has resulted in a low-power complex adaptive filter architecture. Power savings of approximately 21% was shown to be achievable. Relaxed look-ahead [8] pipelined architectures were then developed for achieving high-speed operation. An additional 40-69% power savings was achieved by scaling down the power-supply. Performance evaluation of the proposed architectures for 51.84 Mb/s [5] and 155 Mb/s ATM-LAN is currently underway.

References

[1] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* MA: Addison-Wesley, 1987.

[2] A. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," in *Proceedings of the IEEE*, vol. 83, pp. 498–523, April 1995.

[3] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Minimizing power using transformations," *IEEE Trans. Comp.-Aided Design*, vol. 14, no. 1, pp. 12–31, Jan. 1995.

[4] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *1994 IEEE Symposium on Low Power Electronics*, pp. 8–11, Oct. 1994.

[5] G. H. Im and J. J. Werner, "51.84 Mb/s 16-CAP ATM-LAN standard," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, pp. 620–632, May 1995.

[6] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters - part I : Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 37, pp. 1099–1117, July 1989.

[7] M. Potkonjak and J. Rabaey, "Fast implementation of recursive programs using transformations," in *Proc. ICASSP*, (San Fransisco), pp. 569–572, March 1992.

[8] N. R. Shanbhag and K. K. Parhi, *Pipelined Adaptive Digital Filters.* Kluwer Academic Publishers, 1994.