Decorrelating (DECOR) Transformations for Low-Power Adaptive Filters*

Sumant Ramprasad, Naresh R. Shanbhag, and Ibrahim N. Hajj
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801.
{ramprasa,shanbhag,hajj}@uivlsi.csl.uiuc.edu

Abstract

Presented in this paper are decorrelating transformations (referred to as DECOR transformations) to reduce the power dissipation in adaptive filters. The coefficients generated by the weight update block in an adaptive filter are passed through a decorrelating block such that fewer bits are required to represent the coefficients. Thus, the size of the arithmetic units in the filter (F-block) is reduced thereby reducing the power dissipation. The DECOR transform is well suited for narrow-band filters because there is significant correlation between adjacent coefficients. In addition, the effectiveness of DECOR transforms increases with increase in the order of the filter and decrease in coefficient precision. Simulation results indicate reduction in power dissipation in the F-block ranging from 12% to 38% for filter bandwidths ranging from $0.15f_s$ to $0.025f_s$ (where f_s is the sample rate).

1 INTRODUCTION

The recent proliferation of portable, battery-powered, wireless communication systems has made low power, high performance Digital Signal Processing (DSP) an important research area. A common DSP operation is filtering, where the output, $\widehat{d}(n)$, at time n is given by,

$$\widehat{d}(n) \;=\; \sum_{i=0}^{N-1} w_i(n)x(n-i), \qquad (1)$$

where $w_i(n)$ is the i^{th} coefficient of the adaptive filter and $x(n)$ is the input. The most significant portion of the power dissipation in an adaptive filter occurs in the multipliers in the filter. The power dissipation in the multiplier in turn depends upon the size (i.e. number of bits) of the operands and reducing the operand bit-width will reduce the power dissipation. In this paper, we present decorrelating (**DECOR**) transformations to reduce the number of bits required to represent the filter coefficients. The **DECOR** transforms employ the fact that in most filters, the magnitude of the difference between the absolute values of adjacent coefficients is typically less than the magnitude of the coefficients themselves. Hence, fewer bits are required to represent the differences compared to the actual coefficients. In addition, there is also a reduction in delay and area in certain situations due to smaller bit-widths at the inputs to the multipliers.

The Signal Flow Graph Transformations (SFGT) in [7], which were developed independently, are a special case of the **DECOR** transform. The differences between our work and [7] are as follows. In [8], the **DECOR** transform is applied to infinite-impulse response (IIR) filters, adaptive filters (this paper), filters with rounding after the output of multipliers, and the inputs to a filter in addition to fixed coefficient FIR filters which retain full numerical precision. We study the types of filters that are suitable for **DECOR** transforms and provide gate-level simulations describing the effect of such filter parameters as cutoff frequency and filter order. Power reduction is achieved by reducing the size of the arithmetic units and not by reducing the number of 1's in the coefficients.

Another approach close to **DECOR** in literature is the Differential Coefficients Method (**DCM**) in [10] where differences between adjacent coefficients are employed for fixed coefficient Finite Impulse Response (FIR) filters. The first-order differential coefficients, δ_i^1, are given by,

$$\delta_i^1 \;=\; w_i - w_{i-1}, \qquad (2)$$

where $\{w_i\}_{i=0}^{N-1}$ are the coefficients of the fixed coefficient filter (the time index is omitted because the coefficients do not change with time). Each product term, $w_i x(n-i)$, (except $w_0 x(n)$) in (1) is written as,

$$w_i x(n-i) \;=\; \delta_i^1 x(n-i) + w_{i-1}x(n-i). \qquad (3)$$

The term $\delta_i^1 x(n-i)$ in (3) is computed by a multiplier and added to $w_{i-1}x(n-i)$, which is computed by the previous stage, to give $w_i x(n-i)$. The result of applying **DCM** to the direct form (DF) filter in Figure 1 is shown in Figure 2. It is possible to employ second-order differences, δ_i^2, by repeating the above procedure on the first-order differential coefficients δ_i^1. The advantage of **DCM** is that the bit-width of the coefficients are reduced at the expense of $N-1$ additional adders and delays for an N tap filter. The

*This work was supported by DARPA contract DABT63-97-C-0025, NSF CAREER award MIP-9623737, and NSF award MIP 97-10235.

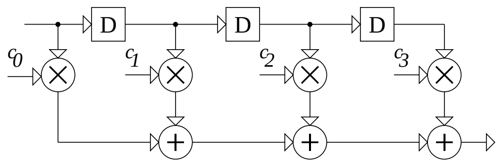

Figure 1: Direct Form (DF) Filter

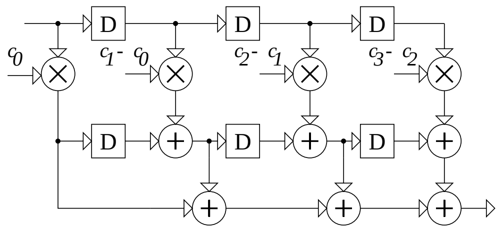

Figure 2: Differential Coefficients Method (**DCM**)

method described in this paper also converts a DF filter into one employing coefficient differences. However, we employ a different formulation of this problem, which results in the following advantages over **DCM**: 1) lower overhead for a given filter order, 2) overhead is independent of the filter order, 3) energy savings over a wider range of bandwidths, 4) easily and efficiently implementable in software, and 5) applicable to adaptive filters.

In other work on low-power adaptive filters, in [5] the total switched capacitance is reduced by dynamically varying the filter order based on signal statistics. In [4], power reduction is achieved by a combination of powering down filter taps and modifying the coefficients. In [11], the strength reduction transformation is applied at the algorithmic level to reduce power dissipation in complex adaptive filters. The techniques in [4, 5, 11] can be applied in addition to **DECOR**.

The rest of this paper is organized as follows. In section 2, the application of **DECOR** transforms to fixed coefficient filters [8] is summarized. In section 3, the **DECOR** transform is applied to adaptive filters and in section 4, simulation results for the reduction in power dissipation are presented.

2 PRELIMINARIES

In this section, we summarize the application of the **DECOR** transform to fixed coefficient filters [8]. In **DECOR**, the transfer function, $H(z)$, is multiplied and divided by the polynomial $f(z)$ given below,

$$f(z) \quad = \quad (1+\alpha z^{-\beta})^m, \qquad (4)$$

where α and β depend upon the type of filter (low-pass, high-pass, band-pass, band-stop) as shown in Table 1. Therefore, the z-transform, $\widehat{D}(z)$, of the output is given by,

$$\widehat{D}(z) \quad = \quad H(z)\frac{(1+\alpha z^{-\beta})^m}{(1+\alpha z^{-\beta})^m}X(z), \qquad (5)$$

where $X(z)$ is the z-transform of the input. In (4) and (5) α, β, and m are integers chosen such that the magnitude of the impulse response of $H(z)(1+\alpha z^{-\beta})^m$ is minimized. The derivation of the optimum values in Table 1 is presented in [8]. In Table 1, the parameter, α, is either 1 or -1 and determines if coefficients spaced β sample delays apart are either added or subtracted respectively. The parameter m is the order of difference and determines the number of times

Table 1: Optimum α and β for different types of FIR filters

Filter Type	α	β	$f(z)$
Low-pass	-1	1	$(1-z^{-1})^m$
High-pass	1	1	$(1+z^{-1})^m$
Band-pass (center = f_c)	1	$\frac{1}{2f_c}$	$(1+z^{-\frac{1}{2f_c}})^m$
Band-stop	-1	2	$(1-z^{-2})^m$

the coefficients are added or subtracted. The filter obtained after applying **DECOR** with $\alpha=-1$, $\beta=1$, and $m=1$ (i.e., $f(z)=1-z^{-1}$) to the DF filter in Figure 1 is shown in Figure 3. In Figure 3, all coefficients, except for the left-most and right-most, are differences of adjacent coefficients in the original filter. Note that the left-most coefficients in Figure 1 and Figure 3 are identical, while the right-most coefficients have opposite signs.

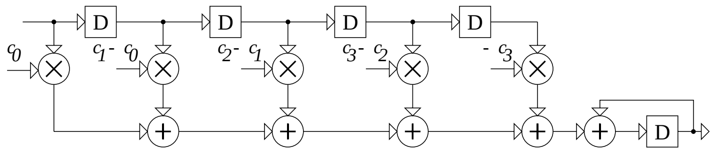

Figure 3: **DECOR** Filter ($\alpha=-1$, $\beta=1$, $m=1$)

For **DECOR** to be useful there must be a reduction in the bit-width of the coefficients, which in turn implies that the maximum magnitude of the impulse response of $H(z)(1+\alpha z^{-\beta})^m$ must be less than half the maximum magnitude of the impulse response of $H(z)$. This typically implies that the filter must have a pass-band width less than $0.1925 f_s$ (where f_s is the sample rate).

3 LOW-POWER ADAPTIVE FILTERS VIA DECOR TRANSFORMS

In order to apply the **DECOR** transform to adaptive filters, we derive the following from (1),

$$\widehat{d}(n) \quad = \quad -\alpha\widehat{d}(n-\beta)+\sum_{i=0}^{N+\beta-1}\delta_i(n)x(n-i). \qquad (6)$$

The derivation of (6) is presented in Appendix A. The $\delta_i(n)$ in (6) are given as follows,

$$\delta_i(n) \quad = \quad \begin{cases} w_i(n) & 0\le i<\beta \\ w_i(n)+\alpha w_{i-\beta}(n-\beta) & \beta\le i<N \\ \alpha w_{i-\beta}(n-\beta) & N\le i<N+\beta \end{cases} \qquad (7)$$

From (7), we see that the first β coefficients are identical to the first original β coefficients and the last β coefficients are the last original β coefficients scaled by α. The center $N-\beta$ coefficients are sums or differences (depending on whether α is 1 or -1) of the original coefficients. The size of the multipliers will be reduced if $max(|\delta_i(n)|)$ is less than $\frac{max(|w_i(n)|)}{2}$. Reducing the size of the multipliers reduces the power dissipation and in certain situations, also reduces the delay and the area of the filter.

In (7), we see that the **DECOR** filter has an overhead of β additional multipliers, adders, and delays due to $\delta_i(n)x(n-i)$, $i=N\ldots N+\beta-1$. From Table 1, we note that β is typically 1 or 2 and independent of the filter order. Each of the multipliers will, however, have a smaller size if $max(|\delta_i(n)|)$ is less than $\frac{max(|w_i(n)|)}{2}$. The **DECOR** filter also requires 1 additional adder and β additional delays to add $-\alpha\widehat{d}(n-\beta))$.

The standard adaptive filter, shown in Figure 4, has 2 blocks,

1. Weight update (**WUD**) block: This block uses the inputs and the error to compute the new coefficients. The weight-update equation for an LMS filter is,
$$w_i(n+1) \quad = \quad w_i(n) + \mu e(n)x(n-i), \qquad (8)$$
where μ is the step size and $e(n)$ is the adaptation error given by,
$$e(n) \quad = \quad d(n) - \widehat{d}(n). \qquad (9)$$
In (9), $d(n)$ is the desired response of the filter.
2. Filter (**F**) block: This block filters the input employing the coefficients computed by the **WUD** block according to (1).

The **DECOR** adaptive filter, shown in Figure 5, has 3 blocks,

1. Weight update (**WUD**) block: This block is identical to the **WUD** block in Figure 4.
2. Decorrelating block: The inputs to this block are the coefficients computed by the **WUD** block. The output consists of decorrelated coefficients.
3. Filter ($\mathbf{F}'$) block: This block filters the input employing the coefficients computed by the decorrelating block according to (6). As can be seen from Figure 3, this block will be different from the **F**-block in Figure 4.

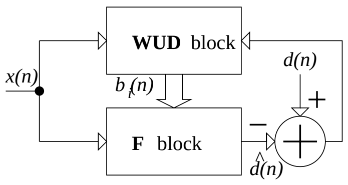

Figure 4: Adaptive Filter

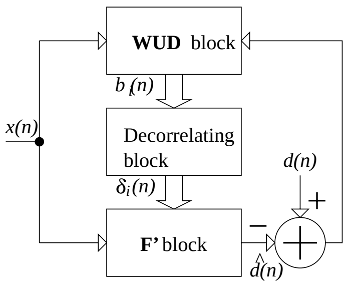

Figure 5: **DECOR** Adaptive Filter

The inputs to the decorrelating block are the N coefficients $\{w_i(n)\}_{i=0}^{N-1}$. The outputs of the decorrelating block are the $N+\beta$ coefficients $\{\delta_i(n)\}_{i=0}^{N+\beta-1}$. The parameters α and β are chosen as in Table 1 depending on the type of filter to be implemented. In (7), the parameter α, which is either 1 or -1, determines if coefficients spaced β taps apart are either added or subtracted, respectively. The proof that the output is identical to the standard adaptive filter as long as the computations in the $\mathbf{F}'$-block are exact (i.e., no rounding or truncation) is presented in Appendix A. Hence the finite precision analysis for the original adaptive filter holds for the **DECOR** adaptive filter as well.

It is possible to employ higher order differences corresponding to $m > 1$ in (5) by cascading more than one decorrelating block as shown in Figure 6. In section 4, we will see that, typically, a single decorrelating block provides the most reduction in power dissipation and multiple decorrelating blocks increase the delay and consume more area.

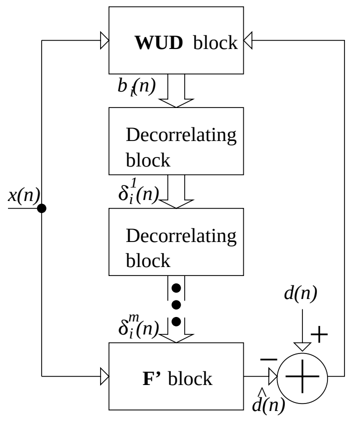

Figure 6: **DECOR** adaptive filter using multiple decorrelators

After the filter has converged the power dissipation in the **WUD** and decorrelating blocks can be reduced substantially by powering them down. Hence, after convergence, only the $\mathbf{F}'$-block will consume power. The block diagrams for an adaptive Least-Mean Squares (LMS) filter and a **DECOR** adaptive LMS filter are shown in Figure 7 and Figure 8, respectively.

Dynamic algorithm transformations [4] can be applied in addition to the **DECOR** transform. In that case, the **WUD** block would send a zero coefficient for the taps to be turned off. The decorrelating block would, as always, generate a new set of coefficients using the coefficients from the **WUD** block.

It may be possible to combine the **WUD** block with the decorrelating block by deriving update equations for $\delta_i(n)$. We can employ (7) and (8) to calculate $\delta_i(n)$ directly from its previous values. The weight update equation for $\delta_i(n)$ is,

$$\delta_i(n+1) = \begin{cases} \delta_i(n) + \mu e(n)x(n-i) & 0 \le i < \beta \\ \delta_i(n-\beta+1) + \mu \sum_{k=0}^{\beta-1}(e(n-k) & \\ +\alpha e(n-k-\beta))x(n-i-k) & \beta \le i < N \\ \delta_i(n-\beta+1) + \alpha\mu & \\ \sum_{k=0}^{\beta-1} e(n-k-\beta)x(n-k-i) & N \le i < N+\beta \end{cases} \qquad (10)$$

Hence it is possible to combine the decorrelating block and the **WUD** block by modifying the **WUD** block to calculate $\delta_i(n)$ directly instead of the **WUD** block computing $w_i(n)$ and the decorrelating block computing $\delta_i(n)$. One potential problem with combining the **WUD** block and the decorrelating block using (10) is that a quantizer is often used in the **WUD** block to reduce the precision of the coefficients. The output of the **DECOR** LMS filter will not be identical to the original filter, in general, because of the quantizer.

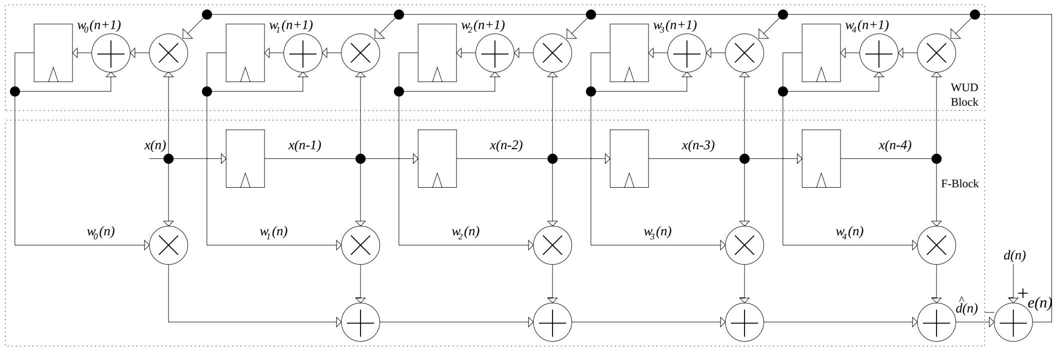

Figure 7: Adaptive LMS Filter

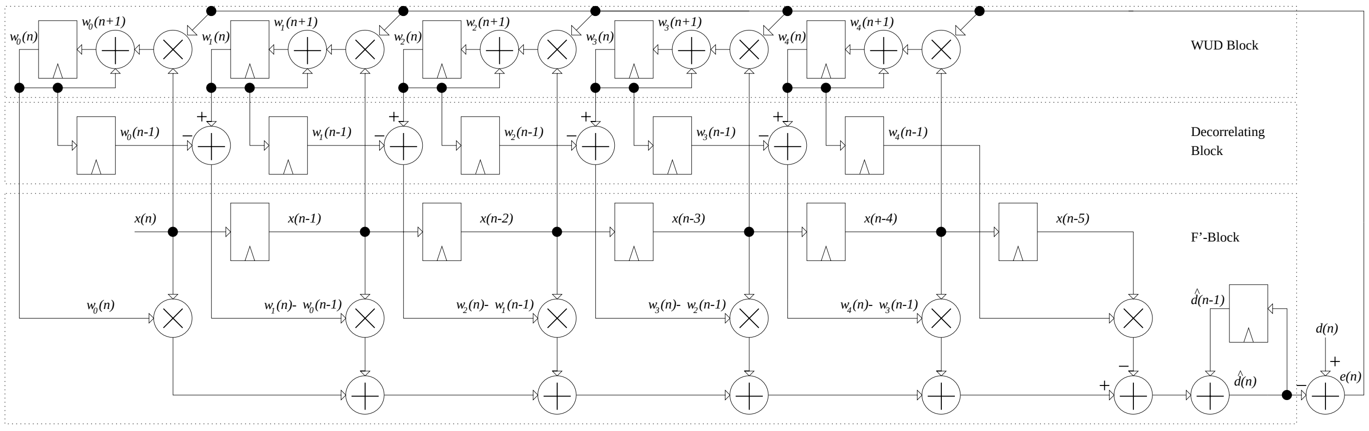

Figure 8: **DECOR** Adaptive LMS Filter

However, the effect of the quantizer will be small if enough bits are used for $w_i(n)$, in which case, the **WUD** and decorrelating blocks can be combined.

In this section, we have so far described the application of the **DECOR** transform to the coefficients. For an LMS filter, we can also apply **DECOR** to the inputs. For instance, we can derive the following from (1),

$$\widehat{d}(n) = -\alpha\widehat{d}(n-\beta) + \sum_{i=0}^{N-1} w_i(n-\beta)(x(n-i) + \alpha x(n-i-\beta))$$

$$+\mu \sum_{i=0}^{N-1} \sum_{j=0}^{\beta-1} e(n-j)x(n-i)x(n-i-j). \qquad (11)$$

The derivation of (11) is similar to that of (6) shown in Appendix A. In (11), we see that one of the inputs to the multiplier is the difference between successive input samples instead of the input samples themselves. We can reduce the size of the multipliers if the input is correlated and the maximum of the difference is less than half the maximum of the original samples. This implies that the input must have its energy concentrated in a narrow band of the frequency spectrum. There is no restriction on the transfer function of the filter when **DECOR** is applied to the input. There is an overhead in the form of the final double summation in (11). After convergence, this term will be small because the errors will be small. Thus we do not need to compute the final term if the **DECOR** transform is applied to the input only after convergence.

4 SIMULATION RESULTS

In this section, we present the results of zero-delay, gate-level simulations of serial LMS filters and **DECOR** LMS filters. In order to perform gate-level simulations, a zero-delay, gate-level model of a ripple-carry adder and a two's complement array multiplier [6] was developed in C. The array multiplier was employed in the simulations because of its simplicity and regularity. The simulation assumed all the multiplications and additions were performed on separate units. All multipliers and adders were assumed to have operands of the same bit-width. In our simulations, we measured the following 3 quantities,

1. The total number of transitions at the inputs to the gates. This is a measure of the power dissipation in CMOS circuits because power is dissipated predominantly during signal transitions.

2. The maximum number of gates between two latches. This is a measure of the critical path or delay.

3. The total number of transistors, which provides a measure of the area.

Only inverters and 2 and 3 input NAND gates were employed to construct the adders and multipliers since the aim is to estimate power dissipation in CMOS circuits. The adder and multiplier models were employed to construct filters. The transitions in a latch were assumed to be equal to the transitions at its inputs. The gates were assumed to have zero-delay since zero-delay simulations are fast. For instance, a zero-delay gate-level simulation of a 40 tap fixed coefficient low-pass FIR filter with 4096 samples of 16 bit input data required 100 seconds on a Sparc Ultra-2, whereas a unit-delay simulation of the same filter required 9840 seconds. One set of simulations was run with unit-delay to examine the effect of changing the delay model on the reduction in power dissipation. The savings in power dissipation are around 10 percentage points lower for a fixed coefficient filter with the unit-delay model compared to the zero-delay model. This is because, in an array multiplier, the number of transitions is higher under the unit-delay model due to glitching caused by reconvergent fanout. Hence, the overhead due to **DECOR** is higher under the unit-delay model since **DECOR** introduces additional multipliers. The overhead can be reduced by employing a different multiplier (ex.: Booth multiplier) with less glitching, or by modifying the array multiplier to reduce glitching (ex.: introducing latches). The correctness of simulations of FIR filters was verified by comparing the outputs of the gate-level simulations and RTL simulations.

The adaptive filter was used for system identification of low-pass FIR filters. The input was 4096 samples of uniform white noise. The desired response was obtained by applying the input to a filter generated using MATLAB's fir1 command. The step-size, μ, was set at 0.0078125. In order to easily compare the output of the serial LMS filter and the **DECOR** LMS filter, all computations in the $\mathbf{F}'$-block were exact (i.e., without any rounding or truncation).

In Figures 9, 10, 11, 12, and 13 we report the impact of pass-band width, filter order, coefficient precision, order of difference, and data precision respectively on power, delay, and area. The base-line filter was a low-pass FIR filter with cutoff of $0.05 f_s$, data precision of 17 bits, coefficient precision of 8 bits, and filter order of 40. We measured the power dissipation in the $\mathbf{F}'$-block separately from the rest of the filter because, after the coefficients have converged, all blocks other than the $\mathbf{F}'$-block can be turned off reducing the power dissipation in those blocks to zero. The coefficient width was determined experimentally by running RTL simulations and noting the maximum magnitude of the coefficients.

From Figure 9, we see that, in general, the percentage reduction in power dissipation in the $\mathbf{F}'$-block is increased as the width of the pass-band is reduced. This is because the differences between adjacent coefficients is smaller for narrow-band filters leading to a greater reduction in bit-width of coefficients. There is little change in delay and area. From Figure 10, we see that the percentage reduction in power dissipation in the $\mathbf{F}'$-block is increased as the filter order is increased. This is because the relative overhead due to the **DECOR** transform is decreased as the filter order is increased. There is also a small reduction in delay and area. From Figure 11, we see that the percentage reduction in power dissipation in the $\mathbf{F}'$-block is increased as the precision of the coefficients is decreased. This is because the reduction in the number of bits is generally independent of the precision due to which the fractional savings is higher for lower precision. There is a small percentage reduction in delay and area which is higher for lower precision. From Figure 12, we see that the percentage reduction in power dissipation is maximized for the first order of difference only. This is because of the overhead for higher order of differences. There is a small reduction in delay and area for the first order of difference. From Figure 13, we see that the percentage reduction in power dissipation in the $\mathbf{F}'$-block is nearly independent of data precision. In all the experiments, the power dissipation in blocks other than the $\mathbf{F}'$-block (i.e., **WUD** and decorrelating blocks) changed by less than $\pm 2\%$.

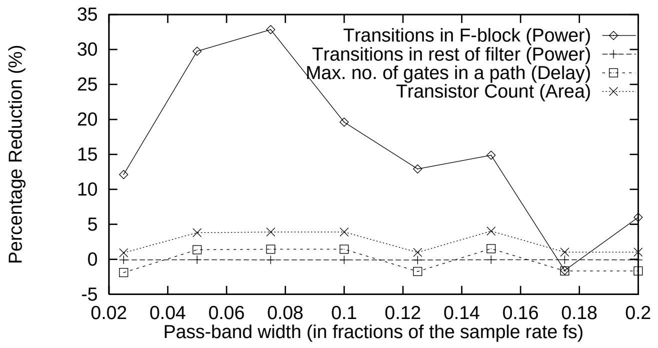

Figure 9: Effect of passband width (filter order = 40, coefficient precision = 8 bits, data precision = 17 bits, $\alpha = -1$, $\beta = 1$, $m = 1$)

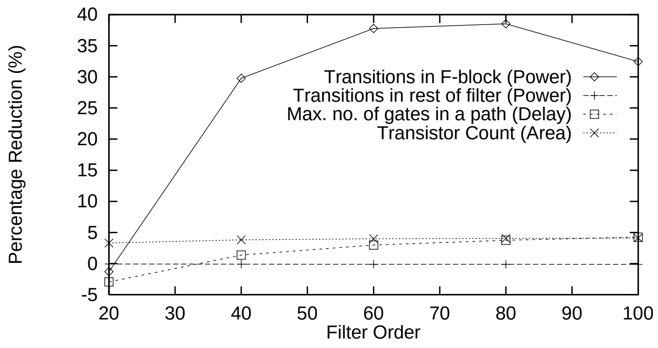

Figure 10: Effect of filter order (cutoff = $0.05 f_s$, coefficient precision = 8 bits, data precision = 17 bits, $\alpha = -1$, $\beta = 1$, $m = 1$)

5 CONCLUSION

In this paper, we presented **DECOR** transforms and applied it to reduce power dissipation in adaptive filters. The coefficients generated by the weight update block in an adaptive filter are passed through a decorrelating block such that fewer bits are required to represent the decorrelated coefficients. Thus the size of the arithmetic units in the filter (**F**-block) is reduced thereby reducing the power dissipation. The **DECOR** transform is suited for narrow-band filters because there is significant correlation between adjacent coefficients. The effectiveness of **DECOR** transforms increases with increase in the order of the filter and decrease in co-

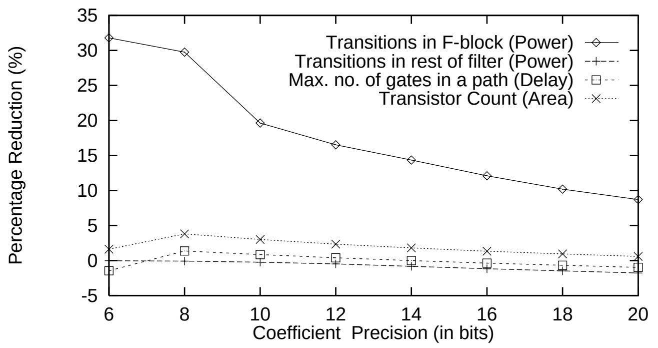

Figure 11: Effect of coefficient precision (cutoff = $0.05 f_s$, filter order = 40, data precision = 17 bits, $\alpha = -1$, $\beta = 1$, $m = 1$)

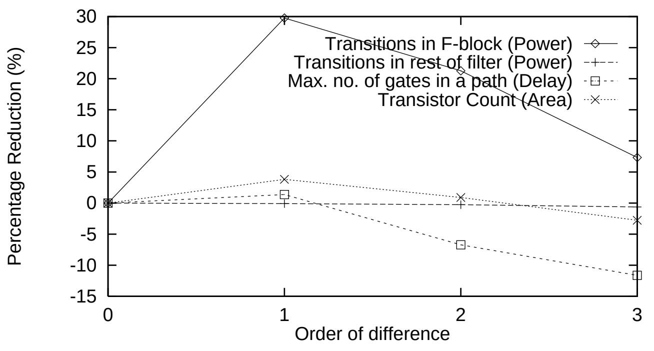

Figure 12: Effect of order of difference (cutoff = $0.05f_s$, filter order = 40, coefficient precision = 8 bits, data precision = 17 bits, $\alpha = -1$, $\beta = 1$)

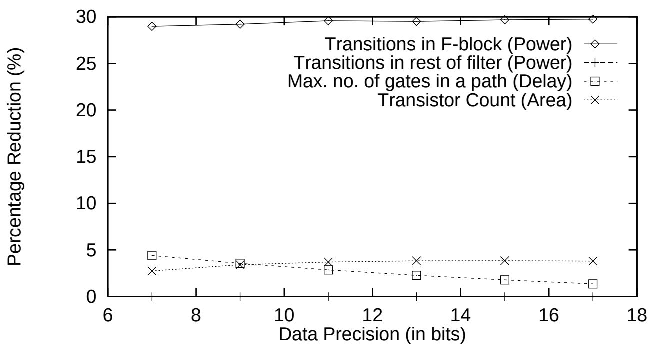

Figure 13: Effect of data precision (cutoff = $0.05f_s$, filter order = 40, coefficient precision = 8 bits, $\alpha = -1$, $\beta = 1$, $m = 1$)

efficient precision. Simulation results indicate reduction in power dissipation in the **F**-block ranging from 12% to 38% for filter bandwidths ranging from $0.15f_s$ to $0.025f_s$.

APPENDIX A

DERIVATION OF (6)

We present the derivation of (6) in this appendix.

$\widehat{d}(n-\beta) = \sum_{i=0}^{N-1} w_i(n-\beta)x(n-\beta-i)$ [Replace n with $n-\beta$ in (1)]

$\widehat{d}(n)+\alpha\widehat{d}(n-\beta) = \sum_{i=0}^{N-1} w_i(n)x(n-i)+\alpha\sum_{i=0}^{N-1} w_i(n-\beta)x(n-\beta-i)$ [Add (1) and above equation]

$= \sum_{i=0}^{\beta-1} w_i(n)x(n-i)+\sum_{i=\beta}^{N-1} w_i(n)x(n-i)+\alpha\sum_{i=0}^{N-\beta-1} w_i(n-\beta)x(n-\beta-i)+\alpha\sum_{i=N-\beta}^{N-1} w_i(n-\beta)x(n-\beta-i)$

$= \sum_{i=0}^{\beta-1} w_i(n)x(n-i)+\sum_{i=\beta}^{N-1} w_i(n)x(n-i)+\alpha\sum_{i=\beta}^{N-1} w_{i-\beta}(n-\beta)x(n-i)+\alpha\sum_{i=N}^{N+\beta-1} w_i(n-\beta)x(n-i)$ [Replace i with $i+\beta$ in the second and third sums]

$= \sum_{i=0}^{\beta-1} w_i(n)x(n-i)+\sum_{i=\beta}^{N-1}(w_i(n)+\alpha w_{i-\beta}(n-\beta))x(n-i)+\alpha\sum_{i=N}^{N+\beta-1} w_{i-\beta}(n-\beta)x(n-i)$

$\Rightarrow \widehat{d}(n) = -\alpha\widehat{d}(n-\beta)+\sum_{i=0}^{\beta-1} w_i(n)x(n-i)+\sum_{i=\beta}^{N-1}(w_i(n)+\alpha w_{i-\beta}(n-\beta))x(n-i)+\alpha\sum_{i=N}^{N+\beta-1} w_{i-\beta}(n-\beta)x(n-i)$

$= -\alpha\widehat{d}(n-\beta)+\sum_{i=0}^{N+\beta-1}\delta_i(n)x(n-i)$, [Using (7)]

which is the desired equation. ¶

References

[1] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Broderson, "Optimizing Power Using Transformations," *IEEE Transactions on Computer-Aided Design of ICs*, vol. 14, no. 1, pp. 12–31, January 1995.

[2] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 498–523, April 1995.

[3] A. Chatterjee and R. K. Roy, "Synthesis of low power linear DSP circuits using activity metrics," *7th International Conference on VLSI Design*, pp. 265–270, Calcutta India, January 1994.

[4] M. Goel and N. R. Shanbhag, "Dynamic Algorithm Transformations (DAT) for Low-Power Adaptive Signal Processing," *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 161–166, Monterey CA, August 18–20 1997.

[5] J. T. Ludwig, S. H. Nawab, and A. P. Chandrakasan, "Low-Power digital filtering using approximate processing," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp. 395–400, March 1996.

[6] M. Hatamian and G. L. Cash, "Parallel Pipelined Multiplier," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 4, pp. 505–513, August 1986.

[7] M. Mehendale, S. B. Roy, S. D. Sherlekar, and G. Venkatesh, "Coefficient Transformations for Area-Efficient Implementation of Multiplier-less FIR Filters," *11th International Conference on VLSI Design*, pp. 110–115, Chennai India, January 4–7 1998.

[8] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, " Decorrelating (DECOR) Transformations For Low-Power Digital Filters," *submitted to 1998 International Conference on Computer-Aided Design.*

[9] N. Sankarayya, K. Roy, and D. Bhattacharya, "Optimizing Computations in a Transposed Direct Form Realization Of Floating-Point LTI-FIR Systems," *ACM/IEEE International Conference on Computer-Aided Design*, pp. 120–125, San Jose CA, November 9–13 1997.

[10] N. Sankarayya, K. Roy, and D. Bhattacharya, "Algorithms for Low Power and High Speed FIR Filter Realization Using Differential Coefficients," *IEEE Transactions on Circuits and Systems – II*, vol. 44, no. 6, pp. 488–497, June 1997.

[11] N. R. Shanbhag and M. Goel, "Low-Power Adaptive Filter Architectures and their Application to 51.84 Mb/s ATM-LAN," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1276–1290, May 1997.