Circuits Syst Signal Process (2011) 30:1521–1548
DOI 10.1007/s00034-011-9310-0

FPGA-Implementation of Parallel and Sequential Architectures for Adaptive Noise Cancelation

Mohammed Bahoura · Hassan Ezzaidi

Received: 20 April 2010 / Revised: 15 April 2011 / Published online: 7 May 2011

Abstract This paper presents a FPGA-based rapid prototyping of an adaptive noise canceller (ANC) using XUP Virtex-II Pro development board and Xilinx System Generator. New parallel and sequential architectures of the ANC are proposed and successfully applied to remove noise from electrocardiogram and speech signals. The pipelined architecture were evaluated and compared to existing high-speed systems using objective measurement tests. By providing comparable filtering performances that of the parallel architectures, the proposed sequential system required fewer material resources.

Keywords Adaptive noise canceller · Sequential adaptive filter · Delayed LMS algorithm · Pipelined architecture · FPGA · ECG

1 Introduction

Adaptive noise canceller (ANC) is an interesting application of adaptive filter that has been used in a wide range of signal processing systems. They include power-line interference (50/60Hz) elimination from electrocardiogram (ECG) signals [16, 18], fetal ECG extraction [18, 21], echo cancelation in long-distance telephone and satellite communications [14], adaptive noise cancelation for hearing aids [4, 19], ambient noise reduction from breath sound measurements [17], etc.

M. Bahoura (✉)
Department of Engineering, Université du Québec à Rimouski, 300, allée des Ursulines, Rimouski, Qc, Canada, G5L 3A1
e-mail: Mohammed_Bahoura@uqar.qc.ca

H. Ezzaidi
Department of Applied Sciences, Université du Québec à Chicoutimi, 550, boul. de l'Université, Chicoutimi, Qc, Canada, G7H 2B1
e-mail: hezzaidi@uqac.ca

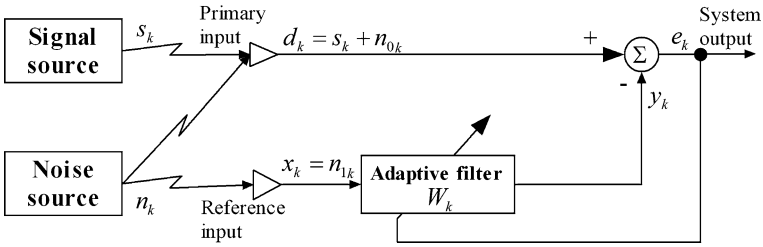


Fig. 1 Adaptive noise canceller block diagram

The adaptive noise canceller is based on a finite impulse response (FIR) filter whose coefficients are automatically adjusted using a gradient descendant algorithm. The least mean square (LMS) algorithm is commonly used to minimize the coast function (error).

In this paper, we propose new pipelined parallel and sequential architectures of the LMS-based adaptive FIR filter. Compared to existing high-speed systems [20], the proposed pipelined architectures ensure the same filtering performances but they presents advantage to require less hardware resources because they need less delay and shifter elements. In addition, the sequential pipelined structure needs only two multipliers regardless of the filter size. The parallel architectures is well suited for small-size filters, while the sequential one is more appropriate for large-size filters. Unlike the previous FPGA-based solutions that need knowledge of a hardware description language (HDL) [16, 20], the proposed solution is implemented on MATLAB/Simulink environment using Xilinx System Generator blockset, without writing a single line of HDL code.

2 Adaptive Noise Cancelation

Figure 1 shows a block diagram of an adaptive noise canceller that consists of two input signals and an adaptive filter. The primary input contains the wanted signal s_k plus the uncorrelated noise n_{0k}. The reference input is the noise n_{1k} which is uncorrelated with the signal s_k but correlated in some unknown way with the corrupting noise n_{0k} [16]. The reference input is processed with the adaptive filter that iteratively adjusts its impulse response to minimize the mean square error (MSE) between its output and the primary input. The adaptive filter is optimized to produce an output y_k that must be as close as possible to n_{0k}. This output is subtracted from the primary input to produce the system output $e_k = s_k + n_{0k} - y_k$.

Assume that signal and noise are uncorrelated, the mean square error (MSE) is given by

$$E\left[e_k^2\right] = E\left[s_k^2\right] + E\left[(n_{0k} - y_k)^2\right]. \tag{1}$$

2.1 Least Mean Square (LMS)

The least mean square (LMS) is the most used algorithm to iteratively minimizing this error (MSE). For an $(N-1)$th order adaptive filter, the LMS algorithm can be

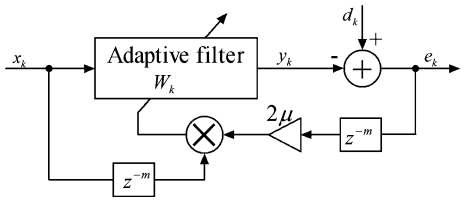


Fig. 2 Block diagram of DLMS algorithm, where m is the number of the delay elements

expressed as

$$y_k = \sum_{i=0}^{N-1} w_{i,k} x_{k-i}, \tag{2}$$

$$e_k = d_k - y_k, \tag{3}$$

$$\mathrm{W}_{k+1} = \mathrm{W}_k + 2\mu e_k \mathrm{X}_k, \tag{4}$$

where $\mathrm{W}_k = [w_{0,k}, w_{1,k}, \ldots, w_{N-1,k}]^T$ are the filter coefficients at time n, $\mathrm{X}_k = [x_k, x_{k-1}, \ldots, x_{k-N+1}]^T$ are the last N samples of the reference input at time n, and μ is a positive parameter controlling the stability and the convergence speed. A larger value for μ can increase the convergence speed, but a smaller value can ensure better stability.

2.2 Delayed Least Mean Square (DLMS)

In some practical applications, the LMS algorithm can be implemented only with delayed coefficient adaptation [11]. As shown in Fig. 2, the so-called delayed least mean square (DLMS) algorithm can be obtained by inserting the delay (z^{-m}) into the error feedback loop of the LMS algorithm [10, 12]. The DLMS algorithm is then expressed by the following equations [11]:

$$e_{k-m} = d_{k-m} - y_{k-m}, \tag{5}$$

$$\mathrm{W}_{k+1} = \mathrm{W}_k + 2\mu e_{k-m} \mathrm{X}_{k-m}, \tag{6}$$

where m is the number of the delay elements and y_{k-m} is a delayed version of that defined in (2).

It is shown that the delay in the coefficient adaptation has only a minor effect on the steady-state behavior if the step size (μ) is within certain bonds [11, 20]. The major penalties with the DLMS algorithm are a reduced convergence speed for stationary signals and a poorer tracking capability for non-stationary signals [11, 20].

3 Architectures

The FIR filters can be implemented by several different architectures: sequential, parallel and semi-parallel. The type of architecture chosen is typically determined by

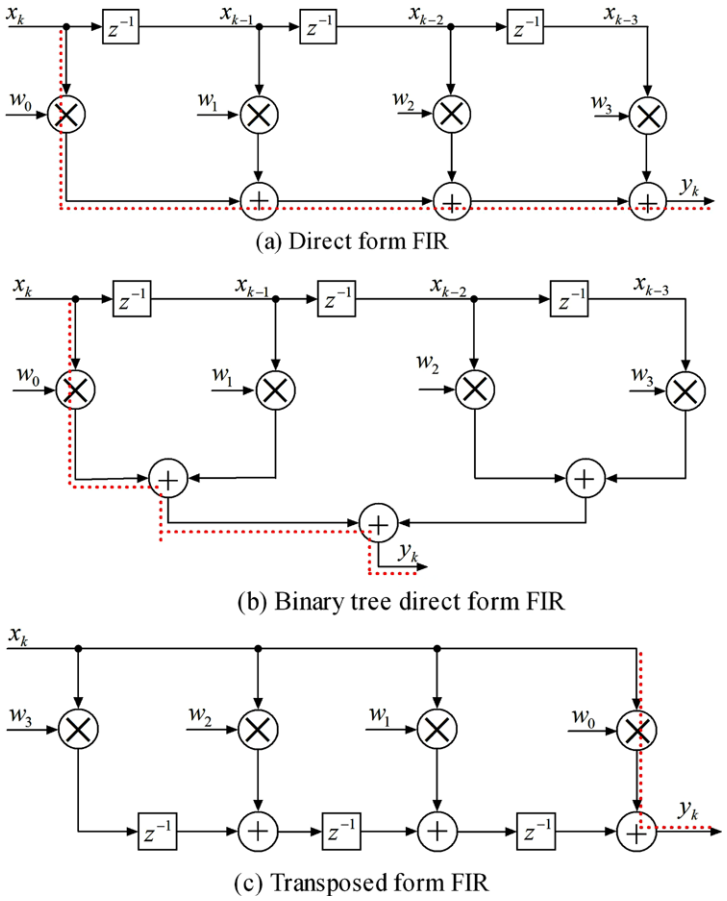


Fig. 3 Canonical implementation of a four-tap FIR filter. The critical path is illustrated by the *dashed line*

the amount of the required processing [9]. The two most important factors are the sampling rate and the number of filter coefficients.

The parallel architecture is well suited for a high sampling rate requirement and a small number of coefficients. However, the sequential architecture is more suited for a low sampling rate requirement and a large number of coefficients. The semi-parallel architecture is a good compromise that enables implementing filters having a large number of coefficients and requiring a high sampling rate.

3.1 Parallel Architectures

3.1.1 Parallel FIR Filters

The finite impulse response (FIR) filters can be implemented using two well-known canonical forms, called the *direct* and *transposed* forms [1]. Figure 3 presents four-tap FIR filters that are functionally equivalent, when the coefficients w_j have a fixed

value. In real-world applications, both forms suffer from important drawbacks. The direct form (Fig. 3(a)) is limited by the critical path corresponding to the longest computation time among all paths that contain zero delays. The critical path is an increasing function of the number of taps, and at the same time needs to be less than a clock period [1]. It is defined by $\tau_{\mathrm{cp}} = \tau_m + (N-1)\tau_a$, where τ_m and τ_a are the time needed for one multiplication and one addition, respectively, and N is the number of taps. For the binary tree direct form (Fig. 3(b)), the critical path is reduced to $\tau_{\mathrm{cp}} = \tau_m + \log_2(N)\tau_a$. The transposed form (Fig. 3(c)) overcomes this limitation by retiming delays into the adder chain. In this case, the critical path is reduced to a single multiply–accumulate operation, $\tau_{\mathrm{cp}} = \tau_m + \tau_a$. However, this form suffers from significant fan-in to apply simultaneously the input data signal to all taps of the filter [1].

3.1.2 Parallel LMS-Based Adaptive FIR Filters

Figure 4 presents the canonical realizations of a four-tap adaptive FIR filter based on the LMS algorithm to adjust its coefficients [13]. The direct form of the LMS adaptive FIR filter (DF-LMS) can be described by the following equations:

$$y_k = \sum_{i=0}^{N-1} w_{i,k} x_{k-i} = w_{0,k} x_k + w_{1,k} x_{k-1} + \cdots + w_{N-1,k} x_{k-N+1}, \tag{7}$$

$$w_{i,k+1} = w_{i,k} + 2\mu e_k x_{k-i}, \tag{8}$$

where e_k is defined by (3). The binary tree direct form of the LMS adaptive FIR filter (TDF-LMS) is defined by the same equations as DF-LMS.

However, the transposed form of the LMS adaptive FIR filter (TF-LMS) can be described by the following equation:

$$y_k = \sum_{i=0}^{N-1} w_{i,k-i} x_{k-i} = w_{0,k} x_k + w_{1,k-1} x_{k-1} + \cdots + w_{N-1,k-N+1} x_{k-N+1}, \tag{9}$$

where $w_{i,k+1}$ is defined by (8).

It is obvious that DF-LMS, TDF-LMS and TF-LMS filters are not performing the same operation when the coefficients vary [1]. In the DF-LMS and TDF-LMS filters, the coefficients contribute to the output signal at the same time (see (7)), while in the TF-LMS filter, they are delayed and contribute at different times (see (9)).

3.1.3 Parallel Delayed LMS-Based Adaptive FIR Filter

Figure 5 presents the canonical realizations of an adaptive FIR filter by inserting delays before the adaptation block, where the shaded boxes represent the inserted

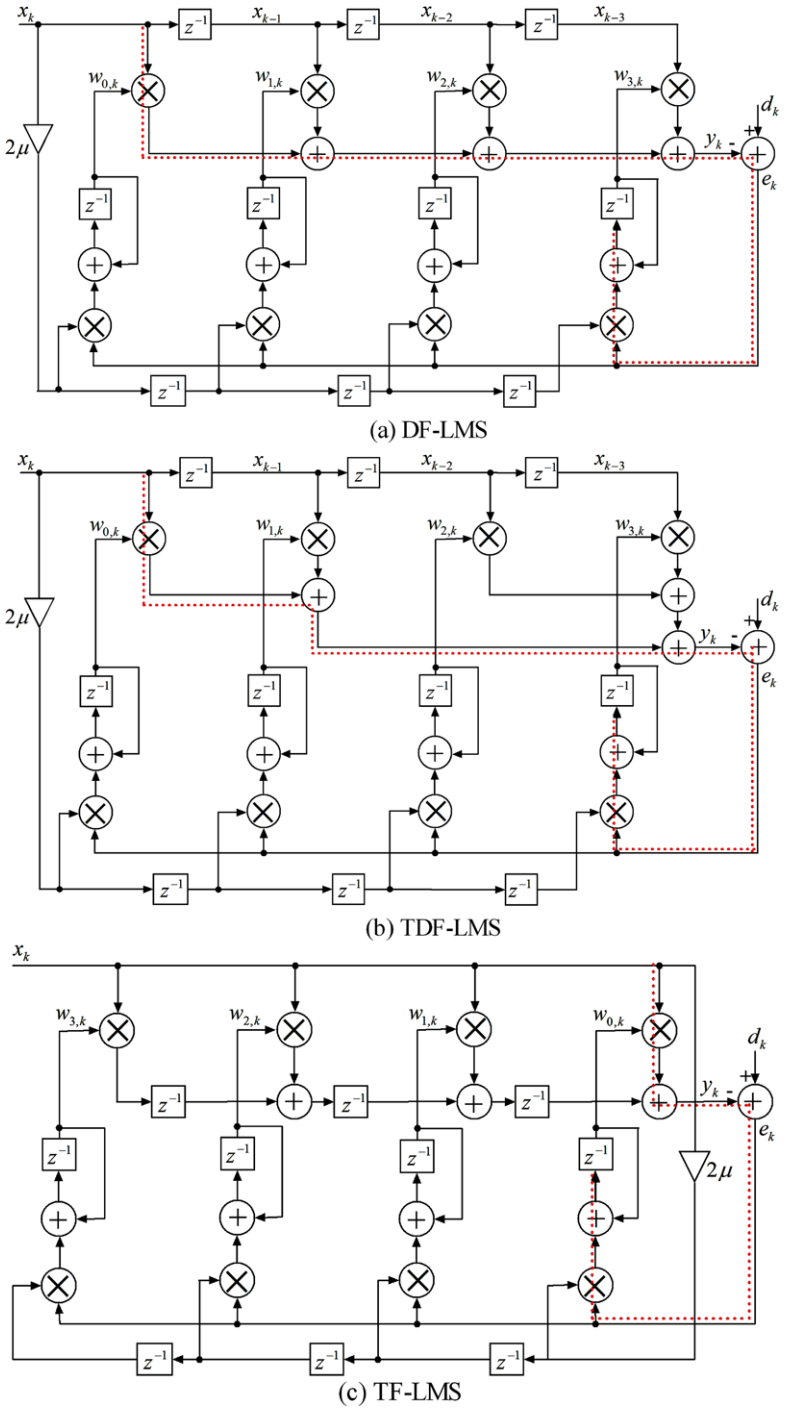


Fig. 4 Canonical implementation of a four-tap adaptive FIR filter based on the LMS algorithm. The critical path is illustrated by the *dashed line*

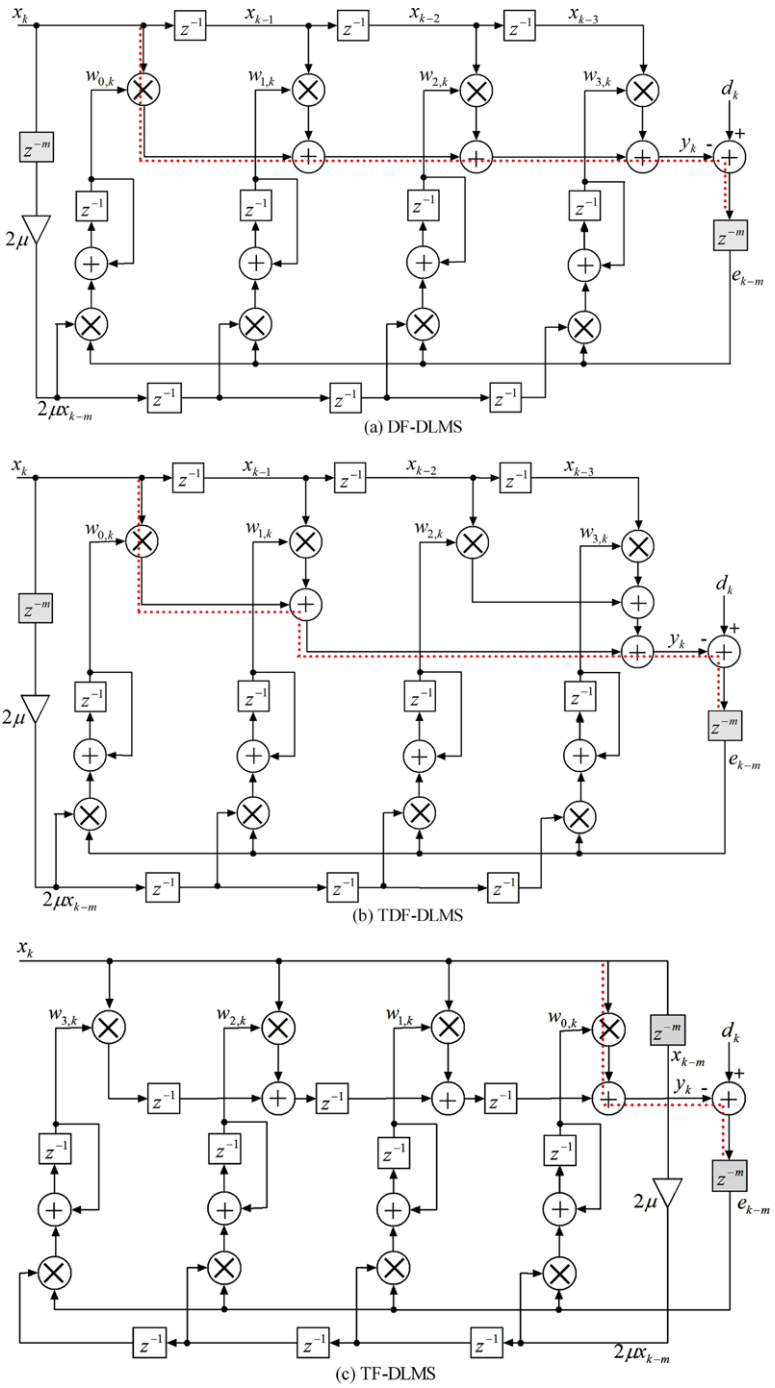


Fig. 5 Canonical implementation of a four-tap adaptive FIR filter based on the DLMS algorithm

delays. The direct form of the DLMS algorithm (DF-DLMS) can be described by the three following equations [20]:

$$y_k = \sum_{i=0}^{N-1} w_{i,k} x_{k-i} = w_{0,k} x_k + w_{1,k} x_{k-1} + \cdots + w_{N-1,k} x_{k-N+1}, \tag{10}$$

$$w_{i,k+1} = w_{i,k} + 2\mu e_{k-m} x_{k-m-i}, \tag{11}$$

where e_{k-m} is defined by (5). The binary tree direct form of the DLMS adaptive FIR filter (TDF-DLMS) is defined by the same equations as DF-DLMS.

However, the transposed form of the LMS algorithm (TF-DLMS) can be described by the following equation:

$$y_k = \sum_{i=0}^{N-1} w_{i,k-i} x_{k-i} = w_{0,k} x_k + w_{1,k-1} x_{k-1} + \cdots + w_{N-1,k-N+1} x_{k-N+1}, \tag{12}$$

where $w_{i,k+1}$ is defined by (11).

In addition to the difference related to the timing of the contribution of the coefficients to the output signal previously observed with the DF-LMS, TDF-LMS and TF-LMS filters, the DF-DLMS, TDF-DLMS and TF-DLMS filters differ also in the way that their respective coefficients are updated.

3.1.4 Parallel Pipelined LMS-Based Adaptive FIR Filter

The retimed delayed least mean square (RDLMS) algorithm re-distributes the inserted delays throughout the circuit architecture in order to reduce the critical path [15]. This process involves the determination of the delays needed to achieve a fully pipelined version of the circuit [20]. In fact, a too small delay value leads to a low-speed circuit while a large value produces a slower convergence rate and a poorer tracking capability [20].

Figure 6 presents the canonical realizations of a four-tap pipelined adaptive filter. The DF-RDLMS, TDF-RDLMS and TF-RDLMS forms are obtained by retiming the DF-DLMS, TDF-DLMS and TF-DMLS, respectively. The coefficient update path is also reduced by inserting delay after the multiplier. A full-pipelining is obtained with $m = N + 2$ for DF-RDLMS, $m = 3 + \log_2 N$ for TDF-RDLMS and $m = 4$ for TF-RDLMS.

3.2 Sequential Architectures

We have recently proposed a sequential pipelined architecture for adaptive noise canceller that have been successfully applied to remove the power-line interference from ECG signals [2]. In this paper, the proposed architecture is described in more details and optimized by reducing the number of the inserted delays. The various path delays are now synchronized.

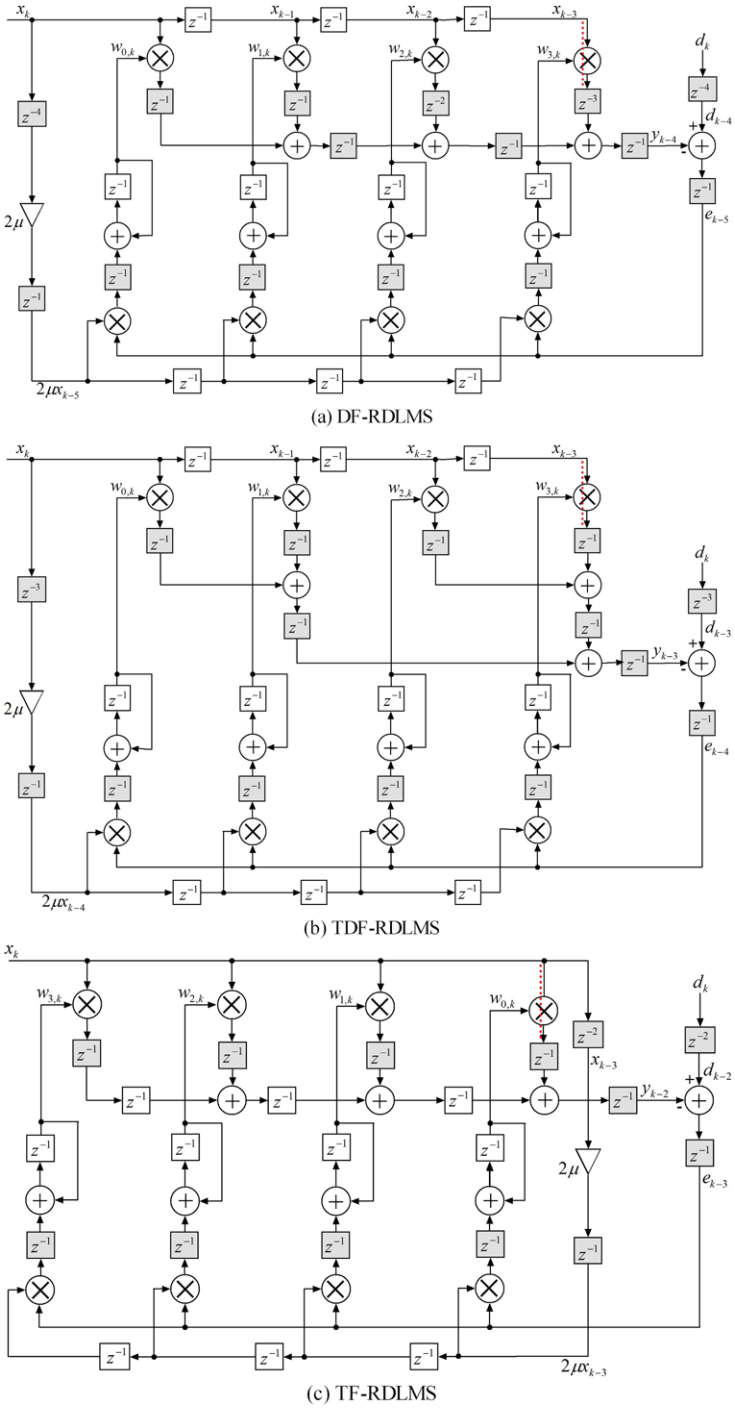


Fig. 6 Canonical implementation of a four-tap adaptive FIR filter based on the RDLMS algorithm. The inserted delay is chosen $m = N + 2$ for the DF-RDLMS, $m = 3 + \log_2 N$ for the TDF-RDLMS, and $m = 4$ for the TF-RDLMS

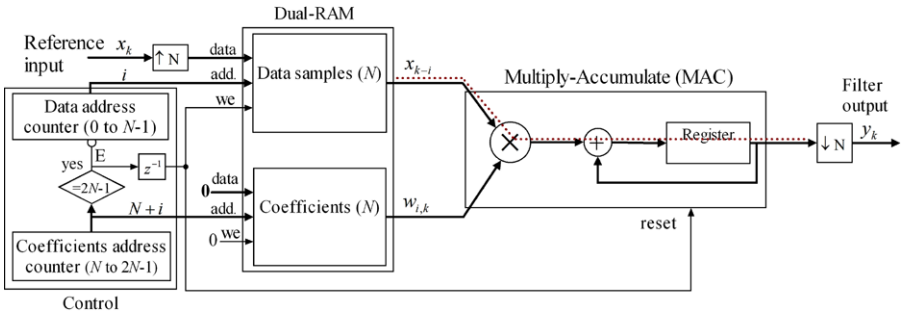


Fig. 7 Sequential FIR filter

3.2.1 Sequential FIR Filter

The multiply–accumulate unit (MAC) is a simple architecture that uses a single multiplier with an accumulator to implement a FIR filter sequentially. As shown in Fig. 7, the input data x_k are up-sampled to allow the sequential computation of the filter output y_k, which is obtained by down-sampling. This architecture reduces hardware (multipliers and adders) at the expense of the filter throughput [9]. The input data and filter coefficients are stored using a dual-port block RAM. The dual-port RAM will be used in a mixed-mode configuration, with the data written and read from port A (RAM mode) and the coefficients read from port B (ROM mode). For a given data address i, respectively, a coefficient address $N+i$, the MAC multiply by $w_{i,k}$ the corresponding delayed data sample x_{k-i} and accumulating the result. The filter output y_k is available after N address clocks (see (2)).

3.2.2 Sequential LMS-Based Adaptive FIR Filter

The proposed sequential LMS-based adaptive FIR filter (SF-LMS) is obtained by updating the filter coefficients of the previous architecture using the LMS algorithm (Fig. 8). The memory addresses are clocked N time faster than the input samples, where N is the number of filter taps. Both RAMs are configured in read after write. For a given data address i, the filter coefficient $w_{i,k}$ is updated using the LMS algorithm (see (13)) before multiplying it by the corresponding delayed data sample x_{k-i} and accumulating the result (see (2)).

$$w_{i,k} = w_{i,k-1} + 2\mu e_k x_{k-i}. \tag{13}$$

After N address clocks, a new input sample is stored in the data RAM and the accumulator is re-initialized. In practice, the implementation of the SF-LMS filter (Fig. 8) is not feasible because some components of Xilinx system generator, such as the MAC and the down-sampler, already include delays that affect the convergence of the LMS algorithm.

3.2.3 Sequential Delayed LMS-Based Adaptive FIR Filter

This sequential delayed LMS-based adaptive FIR filter (SF-DLMS) is an architecture proposed to explain the fully pipelined architecture. As shown in Fig. 9, it is obtained

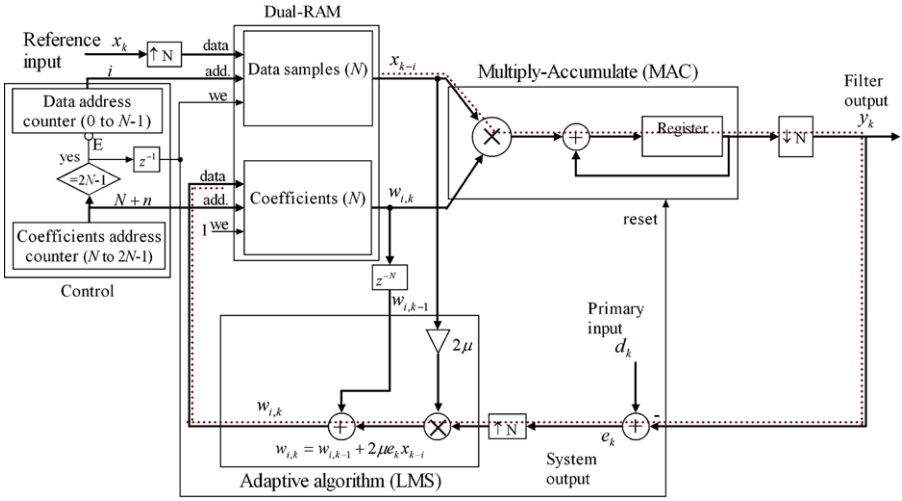


Fig. 8 Sequential LMS-based adaptive FIR filter (SF-LMS)

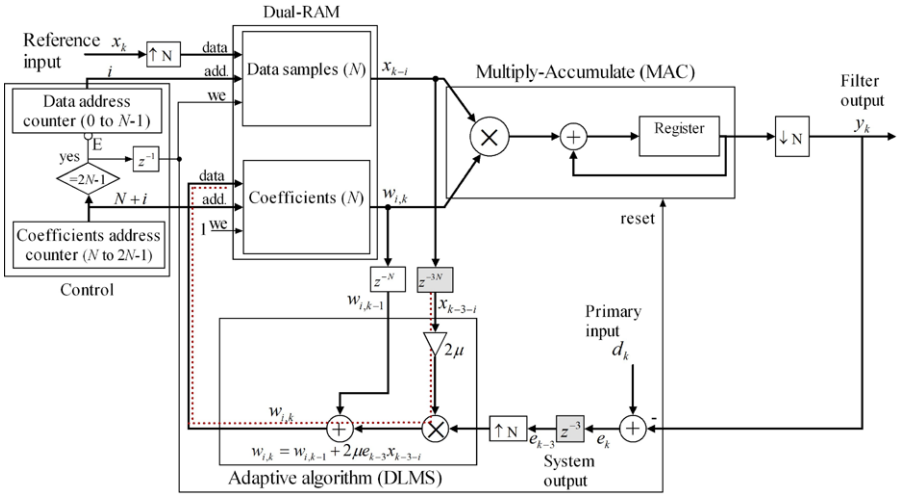


Fig. 9 Sequential delayed LMS-based adaptive FIR filter (SF-DLMS)

by inserting an appropriate number of delays ($m = 3$, which corresponds to $3N$ in the up-sampled signal). The DLMS algorithm is implemented with delayed coefficient adaption (see (14)).

$$w_{i,k} = w_{i,k-1} + 2\mu e_{k-3} x_{k-3-i}. \tag{14}$$

As for the SF-LMS, the implementation of the SF-DLMS filter is not possible because delays of the MAC and the down-sampler are not yet compensated for. These delays affect the convergence of the LMS algorithm.

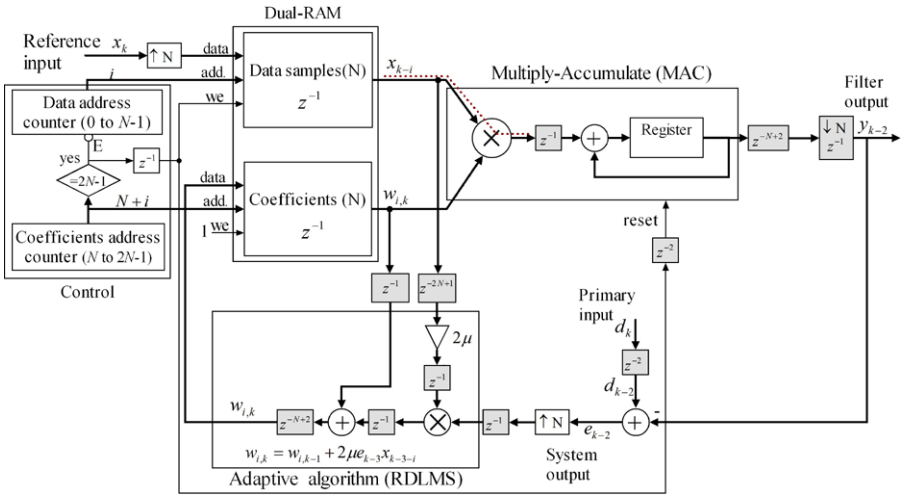


Fig. 10 Sequential pipelined LMS-based adaptive FIR filter (SF-RDLMS)

3.2.4 Sequential Pipelined LMS-Based Adaptive FIR Filter

The sequential retimed delayed LMS-based adaptive FIR filter (SF-RDLMS), as shown in Fig. 10, is obtained by retiming the inserted delays in order to compensate for delays of the MAC and the up-sampler. This architecture enables the increasing the operating frequency by minimizing the critical path [11, 12, 20]. Compared to our previous work [2], this architecture is optimized in terms of the delay elements. It can be noted that one delay (z^{-1}) in the original signal is equivalent to N delays (z^{-N}) in the up-sampled signal.

3.3 Characteristics of Different Architectures

The characteristics of the proposed architectures are compared to those of the high-speed pipelined systems (DF-RDLMS-R, TDF-RDLMS-R, DF-RDLMS-R) published the last years [20]. They are also compared to the characteristics of the single-frequency adaptive noise canceller (SFANC) [18] proposed recently to remove power-line interference from ECG signals [16]. In fact, the SFANC is a notch filter that allows one to cancel only *a priori* known single-frequency interference. Therefore, it is inadequate to remove narrow or large band interferences.

Table 1 compares different LMS filter architectures in terms of the critical path, latency, and overhead hardware. It is shown that the retimed delayed filters (DF-RDLMS, DF-RDLMS, TF-RDLMS, SF-RDLMS, DF-RDLMS-R, DF-RDLMS-R and TF-RDLMS-R) reduce the critical path to the time needed for one multiplication (τ_m). For the pipelined (retimed delayed) architectures, the sequential (SF-RDLMS) and the transposed (TF-RDLMS and TF-RDLMS-R) forms lead to a very small latency and employ less delay elements, compared to the direct (DF-RDLMS, TDF-RDLMS, DF-RDLMS-R and TDF-RDLMS-R) forms. However, the SF-RDLMS

Table 1 Characteristics of different filter architectures. It is assumed that $\tau_a \leq \tau_m$. It can be noted that the shifters and control block were not taken into account

Architecture	Critical path	Latency	Number of hardware elements				
			Adder	Multiplier	Delay	Shifter	RAM (bits)
DF-LMS (Fig. 4a)	$2\tau_m + (N+1)\tau_a$	0	$2N$	$2N$	$3N-2$	1	–
TDF-LMS (Fig. 4b)	$2\tau_m + (2+\log_2 N)\tau_a$	0	$2N$	$2N$	$3N-2$	1	–
TF-LMS (Fig. 4c)	$2\tau_m + 3\tau_a$	0	$2N$	$2N$	$3N-2$	1	–
DF-DLMS (Fig. 5a)	$\tau_m + N\tau_a$	0	$2N$	$2N$	$5N$	1	–
TDF-DLMS (Fig. 5b)	$\tau_m + (1+\log_2 N)\tau_a$	0	$2N$	$2N$	$3N+2+2\log_2 N$	1	–
TF-DLMS (Fig. 5c)	$\tau_m + 2\tau_a$	0	$2N$	$2N$	$3N+4$	1	–
DF-RDLMS (Fig. 6a)	τ_m	N	$2N$	$2N$	$(N^2+13N)/2$	1	–
TDF-RDLMS (Fig. 6b)	τ_m	$1+\log_2 N$	$2N$	$2N$	$6N+1+2\log_2 N$	1	–
TF-RDLMS (Fig. 6c)	τ_m	2	$2N$	$2N$	$5N+5$	1	–
SF-LMS (Fig. 8)	$2\tau_m + 3\tau_a$	0	3	2	N	1	$2N \times 16$
SF-DLMS (Fig. 9)	$\tau_m + \tau_a$	0	3	2	$3N+3$	1	$2N \times 16$
SF-RDLMS (Fig. 10)	τ_m	2	3	2	$4N+4$	1	$2N \times 16$
DF-RDLMS-R [20]	τ_m	$N+1$	$2N+1$	$2N$	$(N^2+15N+8)/2$	N	–
TDF-RDLMS-R [20]	τ_m	$2+\log_2 N$	$2N$	$2N$	$6N+4+2\log_2 N$	N	–
TF-RDLMS-R [20]	τ_m	2	$2N+1$	$2N$	$7N+4$	N	–
SFANC[a] [16]	$2\tau_m + 3\tau_a$	0	4	4	2	2	–

[a]The SFANC is a notch filter that allows one to cancel only *a priori* known single-frequency interference

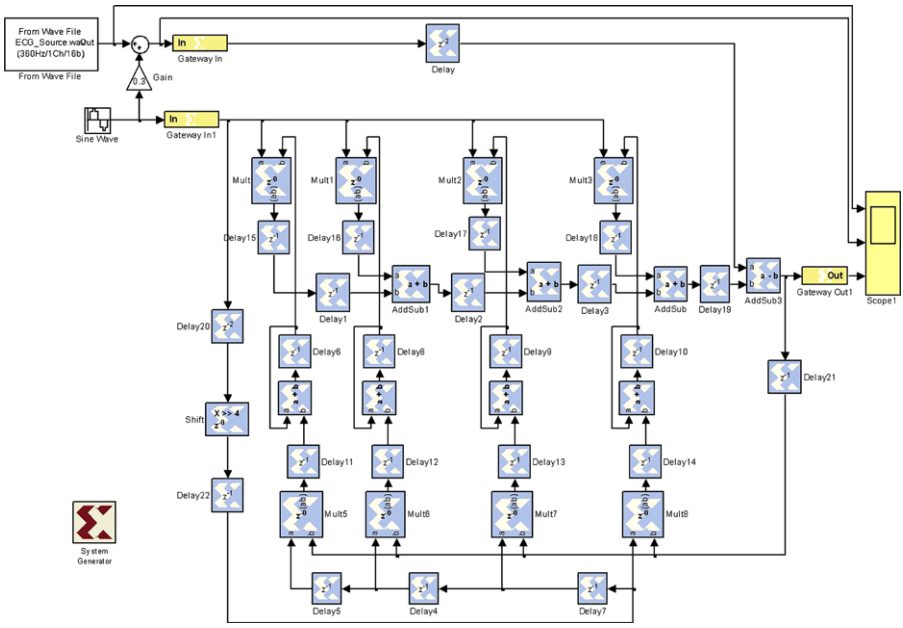


Fig. 11 Simulink block diagram of a four-tap pipelined transposed form (TF-RDLMS) using Xilinx System Generator blockset

uses only two multipliers and three adders, compared to the TF-RDLMS that uses $2N$ multipliers and $2N$ adders.

By choosing a step size value of power-of-2, the multiplication by 2μ can be replaced by a simple shift operation. Thus, each of the proposed architectures needs only one shifter unlike N shifters used by the reference methods [20].

4 Hardware Implementation

4.1 Xilinx System Generator

Xilinx System Generator (XSG) is high-level software tool that enables the use of MATLAB/Simulink environment to create and verify hardware designs for Xilinx FPGAs quickly and easily. It provides a library of Simulink blocks bit and cycle accurate modeling for arithmetic and logic functions, memories, and DSP functions. It also includes a code generator that automatically generates HDL code from the created model. Generated HDL code can be synthesized and implemented in the Xilinx FPGAs. The XSG blocks are like standard Simulink blocks except that they can operate only in discrete-time and fixed-point format.

4.2 FPGA Implementation

In this paper, different architectures of the LMS-based adaptive noise canceller are implemented using Xilinx System Generator under MATLAB/Simulink. Figure 11

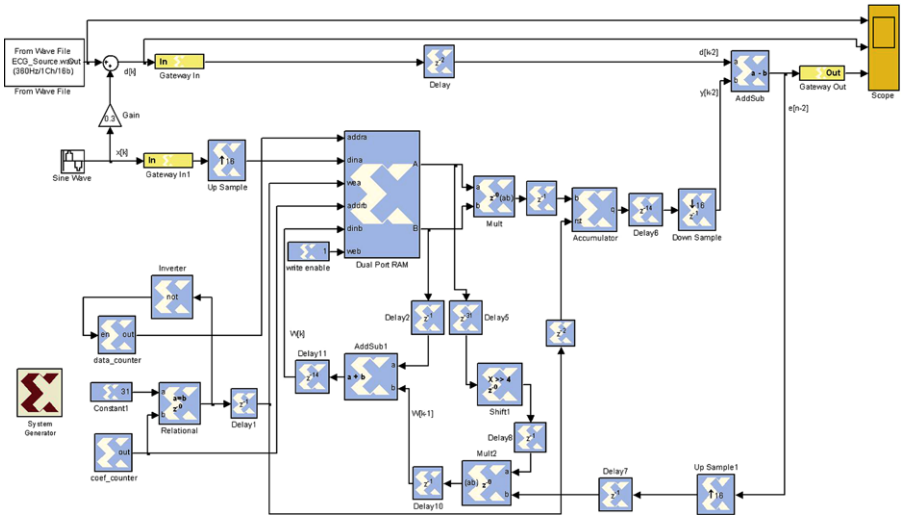


Fig. 12 Simulink block diagram of a sixteen-tap pipelined sequential form (SF-RDLMS) using Xilinx System Generator blockset

represents the parallel architecture (TF-RDLMS) of a pipelined LMS-based adaptive FIR filter, while Fig. 12 represents the sequential one (SF-RDLMS). The fixed-point data is a 2's complement signed 16-bit number having 14 fractional bits.

Xilinx University Program Virtex-II Pro Development System is used to implement this adaptive noise canceller. After successful simulation, the hardware co-simulation compilation automatically creates bitstream file and associates it with a JTAG co-simulation block. The hardware-in-the-loop co-simulation enables incorporating the design running in an FPGA directly into a Simulink simulation. When the design is simulated, the compiled portion (JTAG co-simulation block) is actually running on the hardware and data are transferred between computer and FPGA board.

4.2.1 Conventional Architectures

Table 2 gives the resource requirement and the characteristics of the conventional architectures for various filter lengths, as reported by the Xilinx tools. It can be shown that parallel forms (DF-LMS, TDF-LMS and TF-LMS) require the same number of resources, which increases with the filter length. This can be explained by the number of hardware elements listed in Table 1. It is obvious that the size of the implemented filter is mainly limited by the number of multipliers available on the FPGA. The highest operating frequency and the maximum combinational path delay of these architectures are also presented. These parameters are, respectively, a decreasing and an increasing function of the taps for the direct forms (DF-LMS and TDF-LMS) but they remain constant for the transposed one (TF-LMS). These characteristics can be explained by the critical path that is an increasing function of the taps for the DF-LMS and TDF-LMS filters and remains constant for the TF-LMS (see Table 1 for more details). Finally, these architectures consume approximately the same power that increases slightly with the filter size.

Table 2 Resource utilization, maximum operating frequency, combinational path delay and power consumption for the conventional parallel adaptive filter architectures. Resource availability of Xilinx Virtex-II Pro XC2VP30 FPGA are given between brackets

Architecture	Conventional direct form (DF-LMS)					Conventional binary tree direct form (TDF-LMS)					Conventional transposed form (TF-LMS)				
Filter length	4	8	16	32	64	4	8	16	32	64	4	8	16	32	64
Slices (13,696)	81	177	369	753	1,521	81	177	369	753	1,521	81	177	369	753	1,521
Flip-flops (27,392)	162	354	738	1,506	3,042	162	354	738	1,506	3,042	162	354	738	1,506	3,042
4-LUTs (27,392)	136	272	544	1,088	2,176	136	272	544	1,088	2,176	136	272	544	1,088	2,176
MULT18x18s (136)	8	16	32	64	128	8	16	32	64	128	8	16	32	64	128
GClocks (16)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bonded IOBs (556)	49	49	49	49	49	49	49	49	49	49	49	49	49	49	49
Equivalent gates	34,795	69,843	139,939	280,131	560,515	34,795	69,843	139,939	280,131	560,515	34,795	69,843	139,939	280,131	560,515
Maximum operating frequency (MHz)	59.109	42.241	27.403	17.907	10.576	65.352	58.366	53.024	48.497	44.547	73.477	72.732	72.552	72.195	71.491
Maximum combinational path delay (ns)	14.760	21.509	34.168	53.520	92.224	13.170	15.002	16.738	18.488	20.317	11.494	11.676	11.745	11.972	12.427
Total power consumption (mW)	104.32	104.84	105.61	109.19	113.12	104.42	104.75	105.94	109.76	113.20	104.03	104.71	105.97	108.16	111.33

4.2.2 Pipelined Architectures

Table 3 gives the resource requirement and the characteristics of the pipelined architectures for various filter lengths. It can be shown that the DF-RDLMS architecture requires significantly more resources (slices, flip-flops and 4-LUTs) than the TDF-RDLMS and TF-RDLMS ones because it needs more delays (Table 1). It can also be seen that the size of the implemented filter is mainly limited by the number of multipliers available on the FPGA. The maximum operating frequency is approximately constant for various filter lengths of these architectures. There is no combinational path because the DF-RDLM, TDF-RDLM and TF-RDLMS are fully pipelined. Finally, the TDF-RDLMS and TF-RDLMS architectures consume approximately the same power that increases slightly with the filter size. However, the power consumption of the DF-RDLMS system increases substantially with the filter size. This can be explained by the increasing number of delays. For example, a 64-tap pipelined adaptive filter needs 2,464 delays for the DF-RDLMS but requires only 397 and 325 delays for the TDF-RDLMS and TF-RDLMS, respectively.

For the proposed sequential pipelined architecture (SF-RDLMS), Table 4 gives the required resources for various filter lengths. As can be observed, only two multipliers are used regardless of the filter length (Figs. 10 and 12). However, the number of other resources used (slices, flip-flops and 4-LUTs) increases with the filter length. The maximum operating frequency is approximately constant for various filter lengths. This frequency of course corresponds to the up-sampled signal. However, there is no combinational path because the SF-RDLMS architecture is fully pipelined. Finally, the power consumption increases slightly with the filter size.

Compared to the previously proposed SF-RDLMS [2], the present architecture (Table 4) requires fewer material (Slices, Flip-flops, LUTs) and operates at higher frequency. Tables 3 and 4 also confirm that the sequential architecture needs fewer material resources than the parallel ones. As can be seen, the Virtex-II Pro XC2VP30 FPGA enables implementation of a very high filter size with the sequential architecture (SF-RDLMS) compared to the parallel ones (DF-RDLMS, TDF-RDLMS and TF-RDLMS) that is limited to 64.

5 Experiments and Results

5.1 Database

The proposed architectures are evaluated on actual electrocardiogram (ECG) and speech signals. The ECG signals were taken from MIT-BIH arrhythmia database [8] sampled at 360 Hz. The speech signals were taken from TIMIT database [7] and down-sampled to 8,000 Hz.

5.2 Objective Evaluation Tests

To compare the performance of these architectures, we use three objective evaluation tests: the signal-to-noise ratio (SNR), the normalized mean square error (NMSE) and the cross-correlation (CC) measure.

Table 3 As in Table 2 but with the pipelined parallel adaptive filter architectures

Architecture	Pipelined direct form (DF-RDLMS)					Pipelined binary tree direct form (TDF-RDLMS)					Pipelined transposed form (TF-RDLMS)				
Filter length	4	8	16	32	64	4	8	16	32	64	4	8	16	32	64
Slices (13,696)	294	575	1,138	2,378	5,365	294	575	1,138	2,264	4,516	276	526	1,026	2,017	4,013
Flip-flops (27,392)	402	786	1,554	3,122	6,258	402	786	1,554	3,090	6,162	370	690	1,330	2,594	5,154
4-LUTs (27,392)	200	400	800	1,856	4,736	168	304	576	1,120	2,208	168	304	576	1,120	2,276
MULT18x18s (136)	8	16	32	64	128	8	16	32	64	128	8	16	32	64	128
GClocks (16)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bonded IOBs (556)	49	49	49	49	49	49	49	49	49	49	49	49	49	49	49
Equivalent gates	40,379	81,059	162,275	341,059	747,779	38,331	74,915	147,939	294,275	586,947	38,219	74,291	146,147	290,307	578,883
Maximum operating frequency (MHz)	199.043	197.410	195.104	192.542	187.614	201.016	197.410	195.104	192.542	187.614	201.016	197.410	195.104	192.542	187.614
Maximum combinational path delay (ns)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Total power consumption (mW)	109.28	114.86	124.41	164.64	294.15	106.90	107.74	109.39	112.48	117.54	106.92	107.59	108.26	111.15	116.70

Table 4 As in Table 2 but with the pipelined sequential architecture

Architecture	Pipelined sequential form (SF-RDLMS)				
Filter length	4	8	16	32	64
Slices (13,696)	123	126	135	169	234
Flip-flops (27,392)	194	197	216	283	414
4-LUTs (27,392)	127	130	151	219	352
BRAMs (136)	1	1	1	1	1
MULT18x18s (136)	2	2	2	2	2
GClocks (16)	1	1	1	1	1
Bonded IOBs (556)	49	49	49	49	49
Equivalent gates	79,599	79,656	80,583	84,681	92,805
Maximum operating frequency (MHz)	207.106	207.106	207.106	207.106	207.106
Maximum combinational path delay (ns)	–	–	–	–	–
Total power consumption (mW)	108.67	108.82	109.80	113.85	121.77

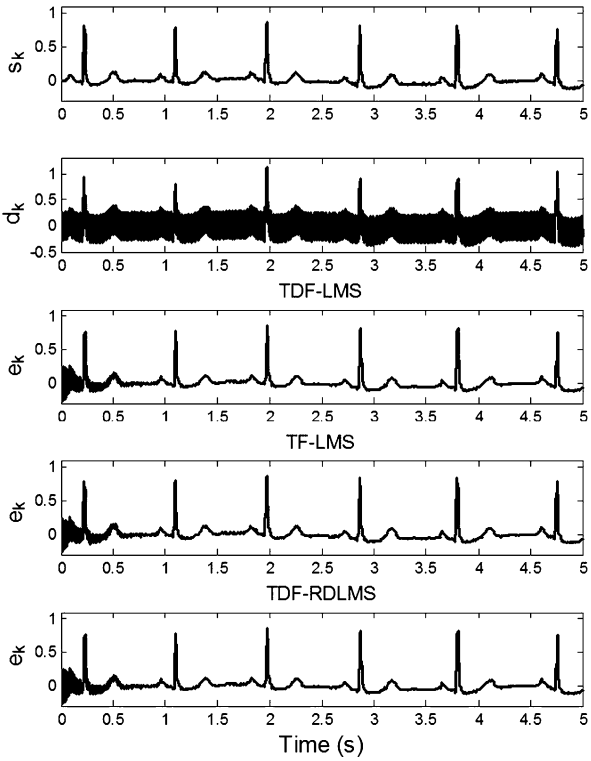
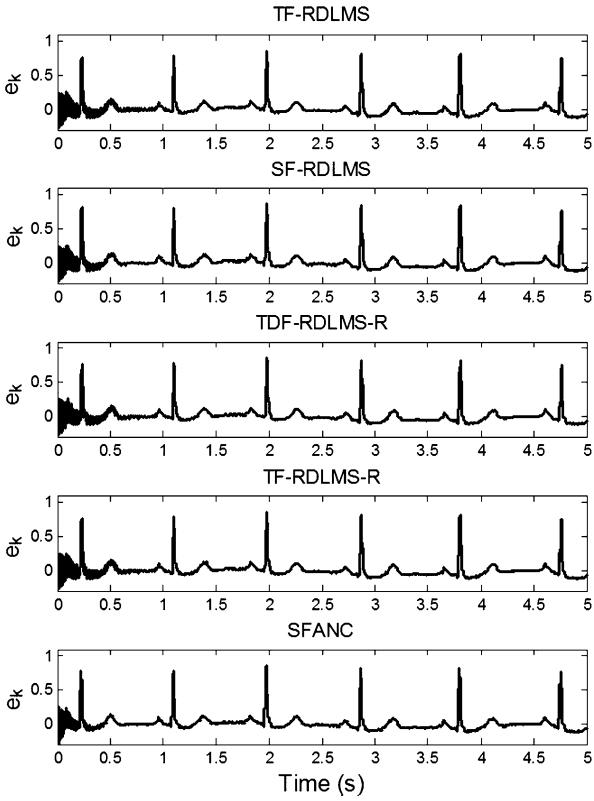


Fig. 13 Results obtained with the first 5 s of the BIH-MIT ECG tape 101 record. The clean signal s_k is given on the top-left, followed by the noised version d_k and the filtered output e_k for various architectures

The SNR evaluates the noise level in the filtered signal. It is defined as [3]

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\sum_{k=0}^{M-1}(s_k)^2}{\sum_{k=0}^{M-1}(s_k-\widehat{s_k})^2}\right), \tag{15}$$

Fig. 13 (*Continued*)



where s_k and $\widehat{s_k}$ are the original signal and the reconstructed signal, respectively. M is the number of samples.

The NMSE evaluates the distortion introduced by the filtering. It is given by [5]

$$\text{NMSE} = \frac{\sum_{k=0}^{M-1}(s_k - \widehat{s_k})^2}{\sum_{k=0}^{M-1}(s_k)^2}. \tag{16}$$

The CC measure is employed to evaluate the similarity between the original signal and the reconstructed signal. It is defined as [6]

$$\text{CC} = \frac{\sum_{k=0}^{M-1}[(s_k - \mu_o)(\widehat{s_k} - \mu_r)]}{\sqrt{\sum_{k=0}^{M-1}(s_k - \mu_o)^2 \sum_{k=0}^{M-1}(\widehat{s_k} - \mu_r)^2}}, \tag{17}$$

where μ_o and μ_r are the average values of the original and reconstructed signals, respectively.

Table 5 Performances and characteristics of various 2-tap adaptive filters used for ECG denoising ($\mu = 2^{-6}$)

Architecture	Noisy	TDF-LMS	TF-LMS	TDF-RDLMS	TF-RDLMS	SF-RDLMS	TDF-RDLMS-R [20]	TF-RDLMS-R [20]	SFANC [16]
Performance									
SNR	1.2436	29.4457	29.4457	27.3340	28.5670	31.1770	27.3347	29.3700	29.4761
NMSE	2.5017	0.0011	0.0011	0.0019	0.0014	0.0007	0.0019	0.0012	0.0011
CC	0.5010	0.9994	0.9994	0.9991	0.9993	0.9997	0.9991	0.9994	0.9994
Characteristic									
Slices (13,696)	–	34	34	153	152	106	161	185	42
Flip-flops (27,392)	–	66	68	210	210	159	226	242	50
4-LUTs (27,392)	–	68	68	100	100	91	100	133	84
BRAMs (136)	–	0	0	0	0	1	0	0	0
MULT18x18s (136)	–	4	4	4	4	2	4	4	4
Equivalent gates	–	17,271	17,271	20,111	20,111	77,163	20,519	21,732	18,167
Max. operating frequency (MHz)	–	73.785	73.785	205.048	205.048	207.106	205.048	205.802	72.971
Max. comb. path delay (ns)	–	11.358	11.4218	–	–	–	–	–	11.364
Total power consumption (mW)	–	103.93	104.08	106.62	106.64	105.92	107.35	108.10	104.62

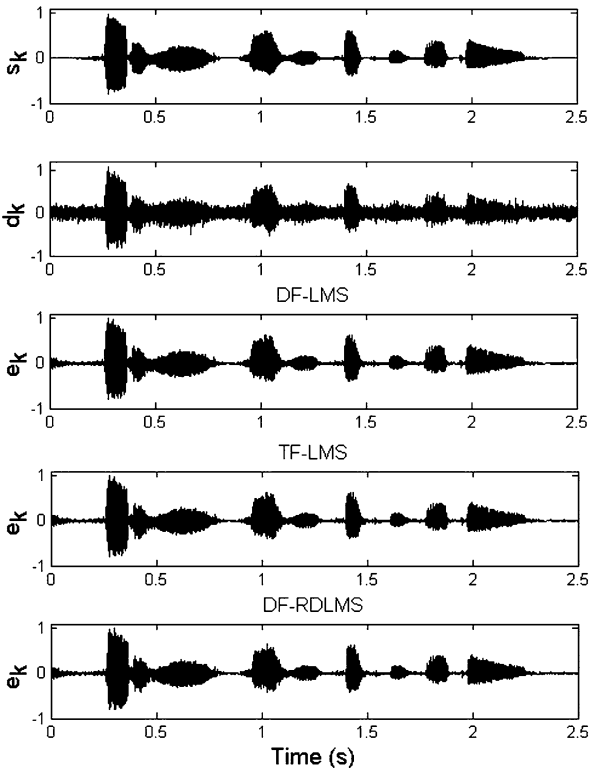


Fig. 14 Results obtained with speech sentence obtained from TIMIT database. The clean signal s_k is given on the top-left, followed by the noised version d_k and the filtered output e_k for various architectures

5.3 Results and Discussion

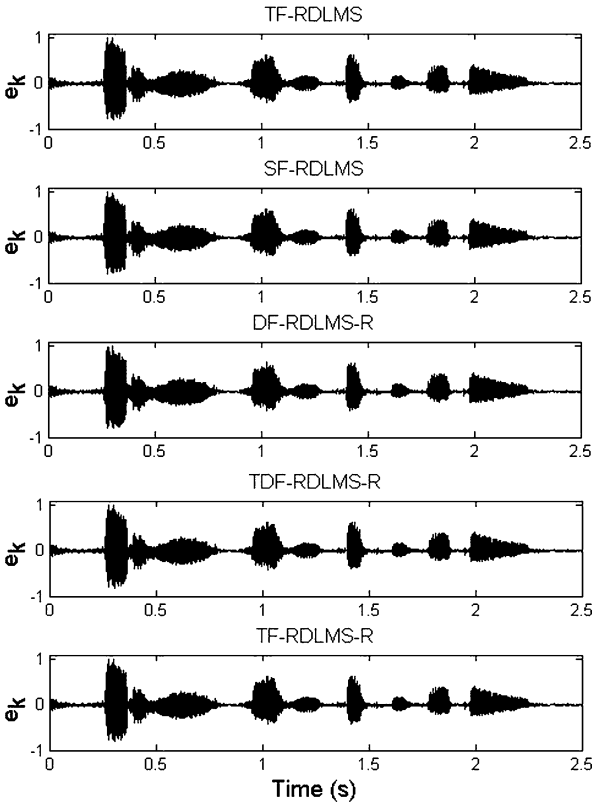
5.3.1 ECG Signal Corrupted by Power-Line Interference

The ability of the ANC architectures to remove power-line interference form actual ECG signals was evaluated. The noised version is obtained by adding a single-frequency interference (60 Hz) to the clean ECG signal. The filter length and the step size of the tested architectures are, respectively, $N = 2$ and $\mu = 2^{-6}$. Only for the SFANC filter [16], the ECG signals must be up-sampled to 1,200 Hz in order to insert the same number of delays that corresponds to the phase shift of 90°.

Figure 13 shows the simulation results obtained with an ECG signal for various adaptive filter architectures (TDF-LMS, TF-LMS, TDF-RDLMS, TF-RDLMS, SF-RDLMS, TDF-RDLMS-R, TF-RDLMS-R and SFANC). The clean signal s_k is given on the top-left, followed by the noised version d_k and the filtered output e_k for various architectures. In this case, the filtered output e_k corresponds to the reconstructed version of the signal $\widehat{s}_k$ used in the last three equations. For the same filter length, these architecture give comparable results, where the unwanted interference is efficiently eliminated. The convergence speed can be improved by increasing the step size μ but a large value can leads to instability.

Table 5 presents the characteristics and performances of theses architectures. As can be seen, conventional parallel systems (TDF-LMS and TF-LMS) gives the same

Fig. 14 (*Continued*)



performance and need approximately the same resource than the SFANC, but they present the advantage to operate at lower sampling frequency (360 Hz) instead of 1,200 Hz needed by SFANC. The pipelined parallel systems improve the maximum frequency at the cost of hardware resources. The best results (SNR, NMSE and CC) are obtained with the pipelined sequential architecture (SF-RDLMS).

5.3.2 Speech Signal Corrupted by White Noise

The proposed architectures are also evaluated on actual speech signal. The corrupted version is obtained by adding an artificial white noise to the clean speech signal. The filter length and the step size of the tested architectures are, respectively, $N = 16$ and $\mu = 2^{-5}$. As indicated previously, the SFANC cannot be applied to remove the additive white noise because it contains more than one frequency.

Figure 14 shows the simulation results obtained with a signal for various adaptive filter architectures (DF-LMS, TF-LMS, DF-RDLMS, TF-RDLMS, SF-RDLMS, DF-RDLMS-R, TDF-RDLMS-R and TF-RDLMS-R). The clean signal s_k is given on the top-left, followed by the noised version d_k and the filtered output e_k for various architectures. Once converged, these adaptive filters remove efficiently the additional noise without disturbing the original speech signal.

Table 6 Performances and characteristics of various 16-tap adaptive filters used for speech denoising ($\mu = 2^{-5}$)

Architecture	Noisy	DF-LMS	TF-LMS	DF-RDLMS	TF-RDLMS	SF-RDLMS	DF-RDLMS-R [20]	TDF-RDLMS-R [20]	TF-RDLMS-R [20]
Performance									
SNR	6.5713	15.4191	15.4191	15.2202	15.3024	15.8047	15.2179	15.3345	15.2165
NMSE	0.2717	0.0281	0.0281	0.0301	0.0293	0.0258	0.0301	0.0290	0.0299
CC	0.8832	0.9859	0.9859	0.9850	0.9853	0.9870	0.9850	0.9854	0.9850
Characteristic									
Slices (13,696)	–	369	369	1,138	1,026	135	1,155	1,147	1,172
Flip-flops (27,392)	–	738	738	1,554	1,330	216	1,586	1,570	1,586
4-LUTs (27,392)	–	544	544	800	576	151	833	576	833
BRAMs (136)	–	0	0	0	0	1	0	0	0
MULT18x18s (136)	–	32	32	32	32	2	32	32	32
Equivalent gates	–	139,999	139,999	162,275	146,147	80,583	163,680	148,131	163,608
Max. operating frequency (MHz)	–	27.403	72.552	195.104	195.104	207.106	195.104	195.104	205.802
Max. comb. path delay (ns)	–	34.168	11.745	–	–	–	–	–	–
Total power consumption (mW)	–	105.61	105.97	124.41	108.26	109.80	126.71	109.88	126.23

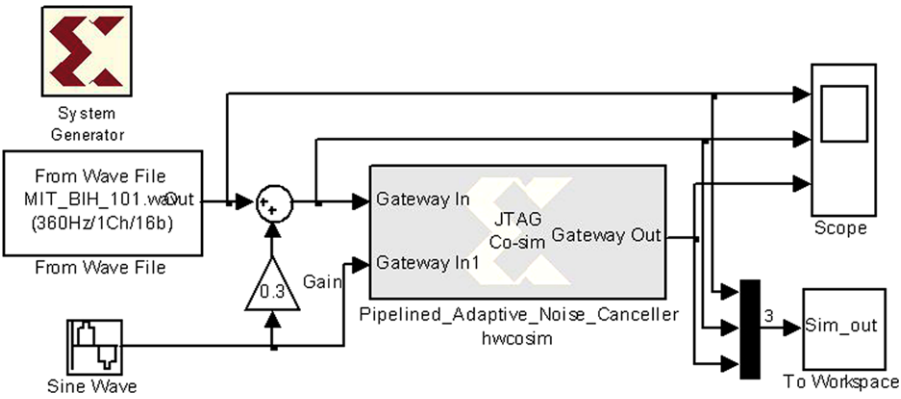


Fig. 15 Hardware-in-the-loop co-simulation of the adaptive power-line interference cancelation from ECG signal

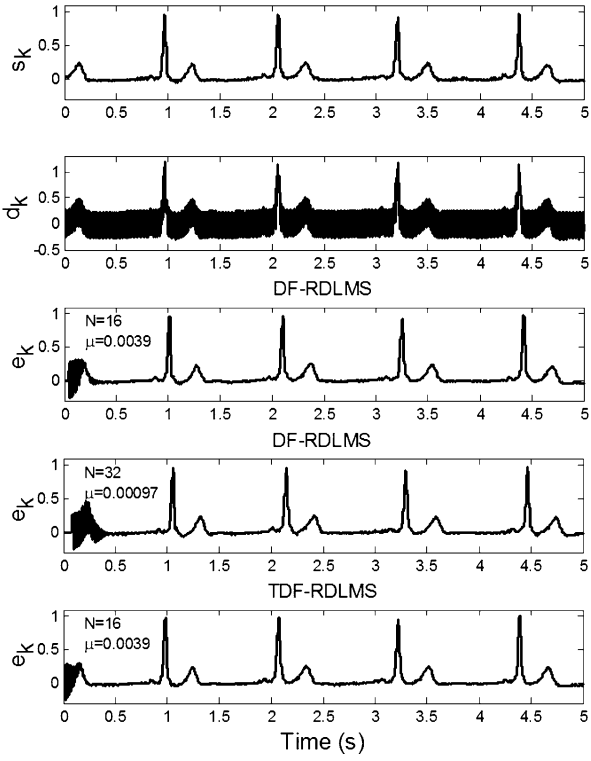
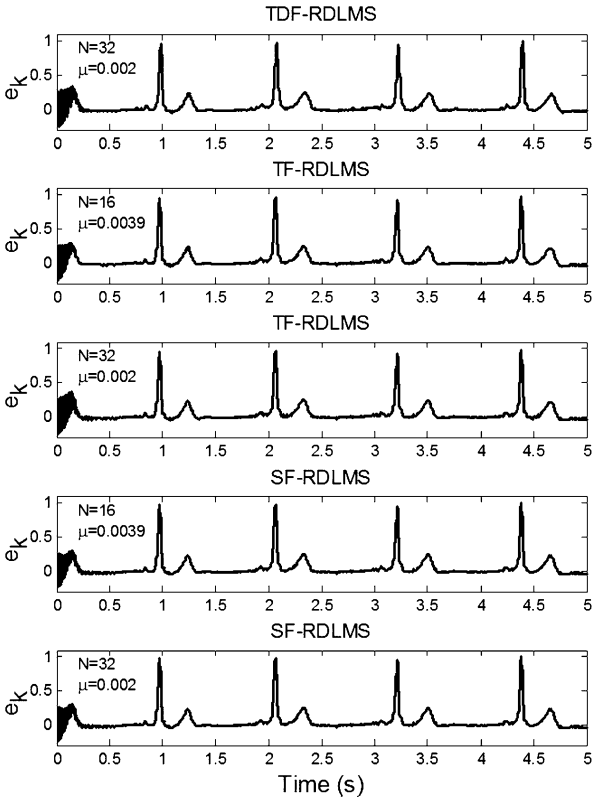


Fig. 16 Results of hardware-in-the-loop co-simulation obtained with the first 5 s of the MIT-BIH ECG tape 202 record. The clean signal s_k is given on the top-left, followed by the noised version d_k and the filtered output e_k for various architectures

Table 6 presents the characteristics and performances of theses architectures. When the filter size increases, the characteristics of the proposed pipelined parallel architectures (DF-RDLMS, TF-RDLMS and SF-RDLMS) become more interesting than those of the reference systems (DF-RDLMS-R, TDF-RDLMS-R and TF-RDLMS-R). As can be seen in this table, the proposed sequential architecture

Fig. 16 (*Continued*)



(SF-RDLMS) presents the advantage of using fewer components (slices, flip-flops, 4-LUTs and multipliers). In fact, this architecture uses only two multipliers regardless of the filter size. Also, it presents the lowest latency at the filter output.

5.3.3 Hardware Co-simulation

The hardware-in-the-loop co-simulation can be accomplished only for the pipelined architectures because the maximum of their operating frequencies (Tables 3 and 4) is greater than the system clock frequency (100 MHz) of the used board. Figure 15 shows the diagram of the hardware-in-the-loop co-simulation that enables running the compiled model (pipelined ANC) on the FPGA.

The hardware co-simulation results obtained with fully pipelined architectures are shown in Fig. 16. It is obvious that the power-line interference is efficiently suppressed without disturbing the ECG signals. Due to the inserted delays in the coefficient adaptation, increasing the size (N) of a pipelined filter requires a reduction of the size step (μ) to ensure a stable response. Also, the pipelining process produces latency at the filter output, which is negligible for TF-RDLMS SF-RDLMS but it can be considerable for DF-RDLMS (Fig. 16). As shown in Table 1, the latency of the DF-RDLMS, TDF-RDLMS, TF-RDLMS and SF-RDLMS architectures is, respectively, equal to N, $1 + \log_2 N$, 2 and 2.

6 Conclusion

In this paper, parallel and sequential LMS-based adaptive FIR filters are proposed and successfully applied to remove power-line interference from ECG signal and white noise from speech signal. Compared to existing systems, the proposed architectures give the same performances, but they present the advantage to require less material resources. The parallel architectures is well suited for small size filters, while the sequential one is more appropriate for a large-size filter. The proposed architectures are implemented in FPGA using Xilinx System Generator. This high-level design tool allows simulation, implementation and verification within MATLAB/Simulink environment, usually without programming in HDL. The obtained results demonstrate also that FPGAs can easily be used to implement high performance DSP functions.

Acknowledgements The authors would like to thank Xilinx Inc for the donation of software through the Xilinx University Program.

References

1. K. Azadet, C.J. Nicole, Low-power equalizer architectures for high-speed modems. IEEE Commun. Mag. **36**(10), 118–126 (1998)
2. M. Bahoura, H. Ezzaidi, FPGA-implementation of a sequential adaptive noise canceller using Xilinx system generator, in *Proceedings of the 21th IEEE International Conference on Microelectronics (ICM'09)*, 19–22 Dec. 2009, pp. 213–216
3. M. Bahoura, J. Rouat, Wavelet speech enhancement based on time-scale adaptation. Speech Commun. **48**(12), 1620–1637 (2006)
4. J.V. Berghe, J. Wouters, An adaptive noise canceller for hearing aids using two nearby microphones. J. Acoust. Soc. Am. **103**(6), 3621–3626 (1998)
5. E. Berti, F. Chiaraluce, N.E. Evans, J.J. McKee, Reduction of Walsh-transformed electrocardiograms by double logarithmic coding. IEEE Trans. Biomed. Eng. **47**(11), 1543–1547 (2000)
6. J. Chen, S. Itoh, A wavelet transform-based ECG compression method guaranteeing desired signal quality. IEEE Trans. Biomed. Eng. **45**(12), 1414–1419 (1998)
7. J.S. Garofolo, L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, N.L. Dahlgren, *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus*, CD-ROM, NTIS edition, 1993
8. A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.Ch. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation **101**(23), e215–e220 (2000)
9. G.C. Hawkes, *DSP: Designing for Optimal Results*. Advanced Design Guide. Xilinx Inc, 1.0 edition, 2005
10. T. Kimijima, K. Nishikawa, H. Kiya, An effective architecture of the pipelined LMS adaptive filters. IEICE Trans. Fundam. **E82-A**(8), 1428–1434 (1999)
11. G. Long, F. Ling, J.G. Proakis, LMS algorithm with delayed coefficient adaptation. IEEE Trans. Acoust. Speech Signal Process. **37**(9), 1397–1405 (1989)
12. K. Matsubara, K. Nishikawa, H. Kiya, Pipelined adaptive filters based on look-ahead-based delayed LMS algorithm. Electron. Commun. Jpn., Part II, Electron. **82**(1), 55–62 (1999)
13. U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd edn. (Springer, Berlin, 2007)
14. K. Murano, S. Unagami, F. Amano, Echo cancellation and applications. IEEE Commun. Mag. **28**(1), 49–55 (1990)
15. K. Nishikawa, H. Kiya, Pipeline implementation of gradient-type adaptive filters. Electron. Commun. Jpn., Part III, Fundam. Electron. Sci. **84**(5), 33–42 (2001)
16. R. Ramos, A. Mànuel-Làzaro, J. Del Río, G. Olivar, FPGA-based implementation of an adaptive canceller for 50/60-Hz interference in electrocardiography. IEEE Trans. Instrum. Meas. **56**(6), 2633–2640 (2007)

17. A. Suzuki, C. Sumi, K. Nakayama, M. Mori, Real-time adaptive cancelling of ambient noise in lung sound measurement. Med. Biol. Eng. Comput. **33**(5), 704–708 (1995)
18. B. Widrow, J.R. Glover Jr., J.M. McCool, Adaptive noise cancelling: principles and applications. Proc. IEEE **63**(12), 1692–1716 (1975)
19. J. Wouters, J.V. Berghe, J.B. Maj, Adaptive noise suppression for a dual-microphone hearing aid. Int. J. Audiol. **41**(7), 401–407 (2002)
20. Y. Yi, R. Woods, L.K. Ting, C.F.N. Cowan, High speed FPGA-based implementations of delayed-LMS filters. J. VLSI Signal Process. **39**(1–2), 113–131 (2005)
21. V. Zarzoso, A.K. Nandi, Noninvasive fetal electrocardiogram extraction: Blind separation versus adaptive noise cancellation. IEEE Trans. Biomed. Eng. **48**(1), 12–18 (2001)