Software/Hardware Implementation of an Adaptive Noise Cancellation System

Dr. Wagdy H Mahmoud, University of the District of Columbia
Dr. Nian Zhang, University of the District of Columbia

Dr. Zhang’s research expertise and interests are neural networks, fuzzy logic, and computational intelligence methods on autonomous robot navigation, pattern recognition, signal and image processing, time series prediction, and renewable energy.

Dr. Zhang received her B.S. in Electrical Engineering at the Wuhan University of Technology, M.S. in Electrical Engineering from Huazhong University of Science and Technology, and Ph.D. in Computer Engineering from Missouri University of Science and Technology. She is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of the District of Columbia (UDC), Washington D.C.

Dr. Zhang is an Associate Editor of IEEE Transactions on Neural Networks. She has served regularly on the organizing committees and program committees of many international conferences. Her research projects are supported from National Science Foundation, US Geological Survey (USGS), and UDC.
Software/Hardware Implementation of an Adaptive Noise Cancellation System

Wagdy H Mahmoud, Nian Zhang
University of the District of Columbia
Washington DC, 20008

Abstract

This paper provides details of our electrical engineering program efforts to introduce software/hardware design concepts and tools in senior-level and senior-design courses. The paper provides details of laboratory exercises and a senior project to implement adaptive filters using variations of the least mean square (LMS) and the recursive least squares (RLS) algorithms and the use of adaptive filters designed using these algorithms in the design of adaptive noise cancellation system. The paper also details the implementation of the adaptive noise cancellation system on an FPGA board. The paper will also detail the challenges involved in teaching continually-evolving software/hardware design tools and the efforts made to reduce their learning times.

Keywords: adaptive filters, adaptive noise cancellation, FPGA, Xilinx system generator, capstone design.

1. Introduction

For the past many years, adaptive filters design has been an active area of scholarly research and innovative implementations. An adaptive filter is a filter that self-adjusts its coefficients according to an optimizing algorithm. Adaptive filters are essential components in a wide range of signal processing, control, and communications including: 1) signal detection; 2) echo cancellation; 3) noise cancellation and/or suppression; 3) channel equalization; 4) system identification and inverse modeling of unknown systems; 5) forward and backward predictions and adaptive tracking; and 6) spectral analysis. In many of these applications, the hardware implementations are essential whenever real-time execution is needed [1]. Technological advances in VLSI fabrications have made Field-Programmable Gate arrays (FPGA) the platform of choice for hardware implementations especially where timing requirement is strict. Such hardware implementations can be realized using hardware description languages such as VHDL or Verilog. Modern FPGA chips contain many resources that are essential for DSP applications such as embedded multipliers, multiply-accumulate units, soft and hard processor cores, and embedded memory blocks [2]. The availability of powerful integrated software and hardware design and synthesis packages, advanced FPGA boards, Intellectual property (IP) designs, and tools that allow seamless integration between these software and hardware design packages has made software/hardware co-design the methodology of choice for the implementations of many of real-time applications.

Due to the wide-range of their applications, Adaptive filters can be used as an excellent vehicle to teach student the concepts of software/hardware co-design and to train them on the use of a wide variety of software and hardware tools that are widely-used by industry. Efforts to introduce adaptive filters to undergraduate students through practical applications, and to create
basic and advanced laboratory exercises and projects suitable for undergraduate students has been reported in [3 - 4].

This paper details our effort to incorporate the teaching of software/hardware design tools through some of the practical applications of adaptive filters. The main goals of such effort is: i) to prepare our undergraduate senior students for professional careers in industry or graduate studies; ii) familiarize our students with state-of-the-art software/hardware design tools, Intellectual property (IP) component and implementation platforms; iii) sharpening students’ abilities to design and implement complex systems using these tools; iv) provide students with skills that can help them long-life learners and successful professionals.

The organization of this paper is as follows. Section 2 provides brief descriptions of the least-mean-square (LMS) algorithm and the Recursive Least Squares (RLS) algorithm. A listing of the most-used variations of LMS and the RLS algorithms is also provided along with brief statements of their advantages and disadvantages. Section 3 provides a list of resources available to our students and details our plan to introduce adaptive filtering concepts and implementations to our undergraduate students. Section 4 provides the details of an active noise cancellation system designed and implemented using available software/hardware resources. Section 5 detail the challenges involved in teaching continually-evolving software/hardware design tools and the efforts made to reduce their learning times.

2. Adaptive algorithms

This section briefly describes two of the most recognized adaptive filter design algorithm; namely the LMS and the RLS.

2.a The LMS algorithm

The most commonly-used algorithm to design adaptive linear filter is the least-mean-square (LMS) algorithm originally developed by Widrow and Hoff [5]. The LMS algorithm is based on the principle of Minimum Mean square error and the steepest descent algorithms [6]. However, unlike the steepest descent algorithm, the LMS does not require exact measurements of the gradient vector, nor does it requires matrix inversion. It is often referred to as the Widrow-Hoff rule. The LMS algorithm is used to search for the solving the Wiener-Hoff equation and find the optimal coefficients $W_{opt}$ for an adaptive filter. The main advantages of the LMS algorithm is its computational simplicity, ease of implementation, unbiased convergence, and the existence of a proof in stationary environment [1]. A block diagram of a typical adaptive noise cancellation system is shown in Figure 1.
Figure 1. Block diagram of a typical adaptive noise cancellation system

The signals shown in Figure 1 are described as follows:

a. The vector \( \mathbf{X}(k) \) represents the is the input vector of time delayed input values and \( x(k) \) is the input at time \( k \).

\[
\mathbf{X}(k) = [x(k) \ x(k-1) \ x(k-2) \ \cdots \ x(k-N+2) \ x(k-N+1)]^T
\]  

b. The vector \( \mathbf{W}(k) \) is used to represent the weights applied to the filter coefficients at time \( k \).

\[
\mathbf{W}(k) = [W_0(k) \ W_1(k) \ W_2(k) \ \cdots \ W_{N-2}(k) \ W_{N-1}(k)]^T
\]  
c. The parameter \( \mu \) is the step size of the adaptive filter.
d. \( e(k) \) is the error between the desired response \( d(k) \) and the output of the filter \( y(k) \), i.e., the filtered signal, at time \( k \).

The LMS algorithm performs the following operations to update the coefficients of the FIR filter:

1. Calculate the output signal \( y(k) \) of the FIR filter. The output of the filter represents an estimate of the desired response. \( y(k) \) is calculated as the convolution of the weight vector and the input vector:

\[
y(k) = \sum_{n=0}^{N-1} W_n(k)x(k-n) = \mathbf{W}^T(k)\mathbf{x}(k)
\]  

2. The error signal \( e(k) \) is estimation error defined as the difference between the estimated response and the desired response.

\[
e(k) = d(k) - y(k)
\]  

3. The error signal and the input signal are applied to the weight update algorithm to updates the filter coefficients.

The LMS algorithm updates its coefficients through the minimization of the mean (expectation) of the instantaneous squared error denoted by \( E[e^2(k)] \). The LMS algorithm assume that \( x(k) \) and \( d(k) \) are wide-sense stationary ergodic processes, therefore, their means and variances are constant. \( \mathbf{X}(k) \) and \( \mathbf{W}(k) \) are assumed to be independent. As explained in detail in [2], the LMS iterative weight update algorithm follows the following equation.

\[
\mathbf{W}(k + 1) = \mathbf{W}(k) + 2\mu e(k)\mathbf{X}(k)
\]  

The step size parameter \( \mu \) is a small positive constant. The selection of value of the step size highly influences the updating of the filter coefficients and has a major impact on the performance of the LMS algorithm. The smaller the selected value for \( \mu \), the larger the time it takes for the adaptive filter to converge to the optimal solution. However, selecting a large value for \( \mu \), may cause the algorithm to be unstable and the output to diverge. For stable behavior and convergence of the LMS algorithm, the step size must be a small positive value (\( \mu \ll 1 \)) and
Where $N$ is the number of taps of the filter and $R$ is the input signal covariance matrix defined as 
\[ R = E(X(n)X^T(n)) \] 
(7)

Recent publications have further limited the bounds of $\mu$ [7], or required the value of $\mu$ to be a power of 2 in order to be executed by shifting instead of multiplication [8].

2.b The LMS Algorithm Variations

Many variations for the LMS algorithm have been reported in the literature. Some of these variations were developed to overcome the shortcoming of technology at their time of development such as long multiplication times. Other variations were developed to improve the tracking ability and the speed of convergence of the algorithm. These variations include:

1. The Normalized Least-Mean-Squares (NLMS) algorithm implements a normalized variation of the LMS algorithm. In this algorithm, the step size is time-varying parameter that increases or decreases as the mean-square error increases or decreases. In each iteration the step size is modified as follows
\[ \mu = \frac{\mu}{x^T(n)x(n) + \epsilon} \] 
(8)

Where $\epsilon$ is a small constant used to prevent the divergence of the algorithm when $x^T(n)x(n)$ is very small. This allows the adaptive filter to better track changes in the system as well as produce small misadjustment error. In many cases, the NLMS algorithm has better stability and accurate tracking capabilities than the LMS and converges more quickly with fewer iteration and fewer samples [9].

2. The sign-Data Least-Mean-Squares (SDLMS) modifies the weights applied to the filter coefficients at each iteration based on the sign of the input data $x(n)$. In vector form, the SDLMS algorithm can be defined as
\[ w(k + 1) = w(k) + \mu e(k) \text{sgn}[x(k)], \text{sgn}[x(k)] = \begin{cases} 1 & x(k) > 0 \\ 0 & x(k) = 0 \\ -1 & x(k) < 0 \end{cases} \] 
(9)

The SDLMS algorithm reduces the amount of computation needed for convergence. However, poorly selected initial conditions may result in unbounded error signal, thus causing the algorithm to be unstable. To ensure the SDLMS algorithm stability and thus convergence, the initial conditions must be set to nonzero values, and step size $\mu$ must be small ($\mu \ll 1$). The practical bounds for $\mu$ are
\[ 0 < \mu < \frac{1}{N[\text{Input Signal Power}]} \] 
(10)

The sign data (or regressor) LMS was first developed to reduce the number of multiplications required by the LMS. The step size, $\mu$, is carefully chosen to be a power of two and only bit shifting multiplies are required and the sign of the error only is used.

3. The Sign-Error Least-Mean-Squares (SELMS) algorithm modifies the weights applied to the filter coefficients at each iteration based on the sign of the error, $e(n)$. In vector form, the SELMS algorithm can be defined as
The \textit{sign error LMS} was first developed to reduce the number of multiplications required by the LMS. The step size, \( \mu \), should be chosen to be a power of two so that only bit shifting multiplies are required and the sign of the data only is used.

4. The Delayed-LMS (DLMS) algorithm: a pipelined variation of the LMS algorithm. It uses registers in the filter and error feedback paths [10 - 11]. The DLMS updates its coefficients according to the following equation:

\[
W(k + 1) = W(k) + 2\mu e(k - D)X(k - D)
\]

Where \( D \) represent the delays in the pipelined architecture. The DLMS-based adaptive filter is faster than the LMS-based filter. However, it needs more logic resources to implement its pipelined architecture.

5. \textbf{Filtered-X LMS}: The filtered-X-LMS is used in active noise control systems where a secondary path from the output of the adaptive filter to the error signal is needed. Details of the algorithm can be found in [12].

6. \textbf{The Fast LMS algorithm}: This algorithm reduces the number of multipliers needed for the hardware implementation of the weight adaption algorithm, and thus significantly reduces the logic resources needed for hardware implementation. It follows the following equation:

\[
W(k + 1) = W(k) + e(k) \cdot \text{sign}(x(k)) \gg n
\]

The Fast LMS algorithm applies the sign bit of the input signal to the error signal and shifts the result by \( n \). The inaccuracy caused by such simplification reduces the accuracy of the adaptation process.

2.c \textbf{The RLS algorithm}

The RLS algorithm has a faster convergence times than the LMS algorithm. However, it requires more computational resources than the LMS algorithm. The RLS algorithm can be implemented using transversal filter structure used to for the implementation of the LMS algorithm and uses the equations (1-4) to represent the input signal \( X(k) \), the filter coefficients \( W(k) \), the computation of the output signal \( y(k) \), and the error signal \( e(k) \) [13]. However, it updates the filter coefficients based on the following equation:

\[
W(k + 1) = W(k) + e(k)m(k)
\]

Where \( m(k) \) is the gain vector that is defined by the following equations

\[
m(k) = \frac{P(k-1)X(k)}{\lambda + X^T(k)P(k-1)X(k)}
\]

Where \( \lambda \) denotes the forgetting factor (0 \( \leq \lambda < 1 \)) and

\[
P(k) = P(k - 1) - m(k)X^T(k)P(k - 1)
\]

For each sample of input data, the RLS algorithm performs an exact minimization of the sum of the squares of the desired signal estimation error. Therefore, for each sample, the RLS algorithm requires the computations of the vector \( m(k) \) and the matrix \( P(k) \). The filter weights are then
updated using the equation (13). The matrix $P(k)$ is the inverse correlation matrix of the input signal $X(k)$. The RLS algorithm suffers from the numerical integrity problem and diverges when the inverse correlation matrix $P(k)$ loses the property of positive definiteness.

2.d The RLS Algorithm Variations

The MATLAB package provides seven variations of the RLS algorithm. However, the most recognized variation of the RLS is the QR Decomposition-Based RLS. The QRRLS solves the instability problem of the standard RLS algorithm by performing QR decomposition on the correlation matrix of the input signal. Therefore, the QRRLS algorithm guarantees the property of positive definiteness. The main drawback of the QRRLS is that it requires more resources than the standard RLS algorithm.

3. Available Resources and Teaching of Adaptive Filtering

The following software and hardware resources, available at the advanced Digital Design lab, will be used to intro adaptive filtering to our senior undergraduate students:

a. The latest versions of the system edition of the Xilinx ISE® design suite including the Xilinx system generator and the embedded design kit (EDK) tools.
b. The latest compatible version of the MathWorks® design tool packages including the signal processing, DSP, fixed-point, communications, image acquisition tool boxes and the HDL coder, HDL verifier, and the filter design HDL coder.
c. The latest version of ModelSim®
d. A large assortment of FPGA boards including the Xilinx University Program XUPV5-LX110T Development System, Spartan 6 FPGA boards, and the latest Xilinx Kintex-7 FPGA KC705 Evaluation Kit, and Xilinx Kintex-7 FPGA Embedded Kit.

3.a Detailed Teaching Plan

The introduction to adaptive filtering will include:

a. A formal study of the steepest descent, the LMS and the RLS algorithm. Reference books [14 -16] provide excellent theoretical review and laboratory exercises for the design of adaptive filters. More information can be found in the user guide of the DSP toolbox.
c. Review and implement the demonstrations and examples given in the DSP System Toolbox User’s Guide, on-line help in both the MATLAB and Simulink implementations of the LMS and RLS algorithm for various applications. In these implementations, students will explore: i) The 10 variations of the MATLAB’s adaptfilt algorithm implementations for the LMS-based algorithm; ii) The seven variations of the adaptfilt algorithm implementations for the RLS-based algorithm; iii) The Simulink blocks for the implementations of both the LMS- and the RLS-based filters. Students will evaluate the performances of these implementations, in terms of convergence, stability and robustness for different applications, step sizes, and number of filter taps. The outcomes of such review and explorations are: i) provide student with a solid understanding of the design of adaptive filters; and b) improve their programming skills so they can code the various versions of the algorithm and compare performances of their programs with the corresponding built-in MATLAB functions and Simulink blocks for the same set of input data and noise.
d. Explore the using of the HDL coder toolbox for generating hardware description language (HDL) code that can be synthesized using the Xilinx ISE design suite and simulated using the ModelSim® software package.

e. Review various published software/hardware implementations of both the LMS- and the RLS-based filters using the Xilinx System Generator® (SysGen). Such a review will help students: i) Explore the capabilities of the SysGen toolbox and how it can be used to design applications; ii) Use the M-block to create user-defined Xilinx blocks; iii) Use the tools of the SysGen to generate synthesizable HDL code for a specific FPGA board (the System Generator token), to view generated signals (the WaveScope) and to estimate the logic resources needed to implement the design on a specific FPGA board (the Resource Estimator); iv) to export the generated HDL code to the ISE suite, synthesize the code and implement the design on the FPGA board; and v) to learn how to use hardware co-simulation capabilities of the SysGen to verify the performance of the generated hardware design and to compare it to the simulated software design.

f. For advanced users: Graduate students will also learn how to use the EDK tool to create: i) input and output interfaces (UART) to the hardware design; ii) to include applications written in the C-language to the designed system. Advanced users should also learn how to use embedded hard core processors (available in the Virtex 5 board) or the downloadable soft core processors available to all boards to optimize the performance of the designed application.

The skills gained from performing the above steps will enable students to: i) develop professional capstone design projects and Master-level theses; ii) improve their research capabilities and their long-life learning skills; iii) enable student to design entire system (speech coding, adaptive noise canceller, echo suppressor, etc.) from the built-in and designed blocks; iv) improve their marketability and enhance their professional careers.

4. Adaptive Noise Cancellation system

In adaptive noise cancellation, the adaptive filter is usually designed as a transversal FIR filter structure [17 - 18]. The transversal filter consists of three basic elements; unit delay elements, multipliers and adders as shown in Figure 2. The unit delay elements are designed using the unit-delay operator \( z^{-1} \). The output of the unit-delay operator is a delayed copy of its input. The number of delay elements represents the filter order.

The adaptive noise cancellation filter employing the LMS algorithm can be implemented as shown in Figure 3.
4.a The Simulink Implementation

The following implementation is a modified version of the design provided in [19]. The design implement the FIR filter with 7 taps and $\mu = 0.002$. The Simulink implementation of the noise cancellation system is shown in Figure 4.

4.b Simulation results

The simulation result of the error signal for the developed design is shown in Figure 6.
4.c The hardware Implementation

The developed system is implemented on the XUPV5-LX110T FPGA board using Xilinx ISE 14.3. The device utilization summary of the ANC system is given in Figure 7. The device was tested using the hardware co-simulation capability of the Simulink and the result was identical to that of the software implemented one.

![Device utilization summary of the ANC system](image)

Figure 7. Device utilization summary of the ANC system
5. Major Challenge and Results

Introducing adaptive filtering concepts, applications, and implementations using software/hardware design methodology into undergraduate senior-level course including the capstone design creates major challenges to the instructors of these courses. These challenges include:

a) Number and functionalities of these tools: A large number of system-level and embedded processing tools and environments are used. These tools are used for design creation, software simulation, HDL code generation, hardware implementations, and simulations.

b) Documentation sizes: Most of these tools have lengthy user guides, references, and getting started manuals.

c) Frequency of tools updates and modifications: currently, most of the software tools are being updated twice a year. FPGA chips and boards become obsolete in a few years. Many of Xilinx and Simulink function blocks becomes obsolete with each new revision requiring major updates for designed demonstrations and examples prepared by the instructor.

The authors have been working on developing system-level design projects for the past few years. So far, students have an extremely positive responses to the materials introduced in pervious capstone and system-level design courses. Analysis of student responses to these newly-developed materials will be the subject of another paper.

Acknowledgement

This study was supported by a grant from the University of the District of Columbia STEM Center (NSF/HBCU-UP / HRD-0928444), Washington, D.C. 20008

6. References


