Implementation of Fixed-Point LMS Adaptive Filter

RAMAYANAPU VAMSI GOPALA KRISHNA¹, N. VENKATA SATISH²

¹PG Student, Dept of ECE, Aditya College of Engineering and Technology, JNTUK, AP, India, E-mail:vamsiramayanapu@gmail.com.
²Sr Assistant Professor, Dept of ECE, Aditya College of Engineering and Technology, JNTUK, AP, India, E-mail:satish.nitw13@gmail.com.

Abstract: In this project, we present an efficient architecture for the implementation of a delayed least mean square adaptive filter. For achieving lower adaptation-delay and area-delay-power efficient implementation, we use a novel partial product generator and propose a strategy for optimized balanced pipelining across the time-consuming combinational blocks of the structure. From synthesis results, we find that the proposed design offers less area-delay product (ADP) and less energy-delay product (EDP) than the best of the existing systolic structures, on average, for filter lengths N = 8. We propose an efficient fixed-point implementation scheme of the proposed architecture, and derive the expression for steady-state error. We show that the steady-state mean squared error obtained from the analytical result matches with the simulation result. Here we are extending this application towards EEG signaling.

Keywords: Area-Delay Product (ADP) And Less Energy-Delay Product (EDP), EEG Signaling.

I. INTRODUCTION

Adaptive digital filters have been applied to a variety of important problems in recent years. Perhaps one of the most well known adaptive algorithms is the least mean squares (LMS) algorithm, which updates the weights of a transversal filter using an approximate technique of steepest descent. Due to its simplicity, the LMS algorithm has received a great deal of attention, and has been successfully applied in a number of areas including channel equalization, noise and echo cancellation and many others. Least mean squares (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired signal and the actual signal). It is a stochastic gradient descent method in which the filter is adapted based on the current time error. The basic idea behind LMS filter is to update the filter weights to converge to the optimum filter weight. The algorithm starts by assuming a small weights (zero in most cases), and at each step, where the gradient of the mean square error, the weights are found and updated. If the MSE-gradient is positive, the error increases positively, else the same weight is used for further iterations, which means we need to reduce the weights. If the gradient is negative, weight need to be increased. Hence, basic weight update equation during the nth iteration:

\[ W_{n+1} = W_n - \mu \Delta e(n) \] (1)

Where \( \mu \) represents the mean-square error, \( \mu \) is the step size, \( W_n \) is the weight vector. The negative sign indicates that, need to change the weights in a direction opposite to that of the gradient slope. The mean-square error is a function of filter weights is a quadratic function which says that it has only one extreme, which minimizes the mean-square error, is the optimal weight. The LMS thus, approaches towards this optimal weight by ascending/descending down the meansquare error verses filter weight curve.

This paper deals with the implementation of the LMS adaptive algorithm for critical path analysis and low complexity implantation of decoder using Verilog HDL. Section II deals with the related work of the project. Section III deals with the methodology used in the paper. Section IV discusses about the proposed results and its operation. Section V discusses about the conclusion and future scope.

II. RELATED WORK

The block diagram of the conventional DLMS adaptive filter is shown in Fig.1. Here the total adaptation delay of m cycles equals to the delay introduced by the filtering process and the weight-update process.

![Fig.1. Block Diagram of Conventional DLMS Algorithm.](image)

1. Various systolic architectures have been implemented using the DLMS algorithm. They are mainly concerned with the increase the maximum usable frequency. Problem with these architectures was they were involving a large...
adaptation delay. This delay is of ~ N cycles for filter length N, which is quite high for large order filters. Fig. 2 shows Nth order pipelined DLMS adaptive filter implemented by Mayer and Agrawal.

Fig.2. Nth Order Pipelined DLMS Adaptive Filter.

2. Large adaption delay reduces the convergence performance of the adaptive filter. Thus to reduce the adaptation delay, Visvanathan et al. [6] have proposed a modified systolic architecture. This architecture has minimal adaption delay and input/output latency. Fig. 3 shows folded systolic array for DLMS algorithm with Boundary Processor Module (BPM) and Folded Processor Module (FPM). The complex FPM circuitry of previous designs has been replaced by simple 2:1 mux along with registers. Appropriate numbers of registers are moved from BPM to FPM. Processors are pipelined. The key transformations used are slow down and folding to reduce adaption delay. With the use of carry-save arithmetic, the systolic folded architecture can support very high sampling rates but are limited by the delay of a full adder.

Fig. 3. Folded Systolic Array for DLMS Algorithm.

3. Tree methods enhance the performance of adaptive filter but they lack in modularity, local connection. Also with the increase in tree stages critical period also increases. In order to achieve a lower adaption delay again, Van and Feng have proposed a systolic architecture, where they have used relatively large processing elements (PEs). The PE combines the systolic architecture and tree structure to reduce adaption delay. But it involves the critical path of one MAC operation.

4. Ting et al. [4] have proposed a fine-grained pipelined design. Pipelining is applied to multipliers to reduce the critical path. Rich register architecture of FPGA can allow pipelining at CLB level, i.e., fine grained pipelining. Thus Virtex FPGA technology is used. Each CLB acts as a 1 bit adder. Various sized ripple carry adders are allowed by dedicated arry logic. This design limits the critical path to the maximum of one addition time and hence supports high sampling frequency. But as large numbers of pipeline latches are being used it involves a lot of area overhead for pipelining and higher power consumption. Also the routing of FPGA adds very large delay.

5. Meher and Maheshwari modified the conventional DLMS algorithm to an efficient architecture with inner product computation unit and pipelined weight update unit[3]. Adaption delay of DLMS adaptive filter can be divided into two parts: one part is delay introduced in pipelining stages of filtering and second part is delay introduced in pipelining stages of weight updation. Based on these parts DLMS adaptive filter can be implemented as shown in Fig.4. The weight-update equation of the DLMS algorithm is given by

\[ W_{n+1} = W_n + \mu \cdot \hat{e}_n \cdot x_{n+1} \tag{1} \]

Where \( \hat{e}_n = d_n - y_n \) and \( y_n = W_n^T x_n \).

Computations of the error-computation block and the weight-updation block are decoupled by the modified DLMS algorithm. This allows optimal pipelining by feedforward cut-set retiming of both these sections separately to minimize the number of pipeline stages and adaptation delay.

6. Further Meher and Maheshwari have proposed a 2-bit multiplication cell along with an efficient adder tree for pipelined inner product computation. This architecture minimizes the critical path and silicon area without increasing the number of adaptation delays. Both error computation and weight update block is implemented using 2-bit multiplication cell.

Fig.4. 2-Bit Multiplication Cell.

The \( L/2 \) number of AND/OR cell (AOC) and 2-to-3 decoders makes the complete multiplication cell which multiples word A and input x each of 1 bits. Each AOC contains 3 AND cells and an OR-tree of 2 OR cells. Each AND gate takes \( (W+2) \)-bit input D and a single bit input b. Each OR gate takes \((W+2)\) input pair words. The 2-to-3
decoder takes a 2-bit input and produces three output b0, b1 and b2. The AOC perform a multiplication of input operand A with two-digit (Xl, xo), such that the 2-bit multiplication cell.

III. METHODOLOGY

Least squares means (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time as shown in Fig.5. It was invented in 1960 by Stanford University professor Bernard Widrow and his first Ph.D. student, Ted Hoff.

![Fig.5. LMS adaptive filter.](image)

**Relationship To The Least Mean Squares Filter:** The realization of the causal Wiener filter looks a lot like the solution to the least squares estimate, except in the signal processing domain. The least squares solution, for input matrix $X$ and output vector $y$ is

$$\hat{\beta} = (X^TX)^{-1}X^Ty.$$  \hspace{1cm} (2)

The FIR least mean squares filter is related to the Wiener filter, but minimizing the error criterion of the former does not rely on cross-correlations or auto-correlations. Its solution converges to the Wiener filter solution. Most linear adaptive filtering problems can be formulated using the block diagram above. That is, an unknown system $h(n)$ is to be identified and the adaptive filter attempts to adapt the filter $h(n)$ to make it as close as possible to $h(n)$, while using only observable signals $x(n)$, $y(n)$, $d(n)$ and $e(n)$, but $v(n)$ and $h(n)$ are not directly observable. Its solution is closely related to the Wiener filter.

**Disadvantage of LMS Adaptive Filter, Due To Which We Are Going For Delayed LMS Adaptive Filter:** The Least Mean Square (LMS) adaptive filter is the most popular and most widely used adaptive filter, not only because of its simplicity but also because of its satisfactory convergence performance. The direct-form LMS adaptive filter involves a long critical path due to an inner-product computation to obtain the filter output. The critical path is required to be reduced by pipelined implementation when it exceeds the desired sample period. Since the conventional LMS algorithm does not support pipelined implementation because of its recursive behavior, it is modified to a form called the delayed LMS (DLMS) algorithm, which allows pipelined implementation of the filter.

**Pipelining is an important technique used in several applications such as digital signal processing (DSP) systems, microprocessors, etc. It originates from the idea of a water pipe with continuous water sent in without waiting for the water in the pipe to come out. Accordingly, it results in speed enhancement for the critical path in most DSP systems. For example, it can either increase the clock speed or reduce the power consumption at the same speed in a DSP system. Pipelining allows different functional units of a system to run concurrently. Consider an informal example in the following fig.6. A system includes three sub-function units (F0, F1 and F2). Assume that there are three independent tasks (T0, T1 and T2) being performed by these three function units. The time for each function unit to complete a task is the same and will occupy a slot in the schedule. If we put these three units and tasks in a sequential order, the required time to complete them is five slots.

![Fig.6. pipelining of five slots.](image)

However, if we pipeline T0 to T2 concurrently, the aggregate time is reduced to three slots as shown in Fig.7.

![Fig.7. Pipelining of three slots.](image)

Therefore, it is possible for an adequate pipelined design to achieve significant enhancement on speed. In this project, we are giving the adequate emphasis for the fixed point implementation issues, which are not discussed in the existing work due to the recursive behavior of the LMS algorithm. We present here the optimization design to reduce the number of pipeline delays along with the area, sampling period, and energy consumption. The proposed design is found to be more efficient in terms of the power-delay product (PDP) compared to the existing structures.
weights of LMS adaptive filter during the nth iteration are updated according to the following equations

\[ w_{n+1} = w_n + \mu \cdot e_n \cdot x_n \]  

(3)

where

\[ e_n = d_n - y_n \]

\[ y_n = w_n^T \cdot x_n \]

and the input vector \( x_n \), and the weight vector \( w_n \) at the nth iteration are, respectively, given by

\[ x_n = [x_{n-1}, x_{n-2}, \ldots, x_{n-N+1}]^T \]

\[ w_n = [w_n(0), w_n(1), \ldots, w_n(N-1)]^T \]  

(4)

A. Block Diagram

The proposed DLMS adaptive filter structural block diagram is as shown in Fig.8. \( d_n \) is the desired response, \( y_n \) is the filter output, and \( e_n \) denotes the error computed during the nth iteration. \( \mu \) is the step-size, and \( N \) is the number of weights used in the LMS adaptive filter.

![Fig.8. The proposed DLMS adaptive filter structural block diagram.](image)

In pipelined designs with \( m \) pipeline stages, the error \( e_n \) becomes available after \( m \) cycles, where \( m \) is called the “adaptation delay.” The DLMS algorithm therefore uses the delayed error \( e_{n-m} \), i.e., the error corresponding to the \((n-m)\)th iteration for updating the current weight instead of the recent-most error. The weight-update equation of DLMS adaptive filter is given by

\[ w_{n+1} = w_n + \mu \cdot e_{n-m} \cdot x_{n-m} \]  

(5)

The modified DLMS algorithm decouples computations of the error-computation block to minimize the number of pipeline stages and adaptation delay. As shown in Fig.9, there are two main computing blocks in the adaptive filter architecture: 1) the error-computation block, and 2) weight-update block.

The error computation block consists of \( N \) number of 2-b partial product generators (PPG) corresponding to \( N \) multipliers and a cluster of \( L/2 \) binary adder trees, followed by a single shift–add tree. If we introduce pipeline latches after every addition, it would require \( L(N-1)/2 + L-1 \) latches. The adaptation delay is decomposed into \( n_1 \) and \( n_2 \). The error-computation block generates the delayed error by \( n_1 \) cycles, which is fed to the weight-update block shown in Fig.10 after scaling by \( \mu \); then the input is delayed by 1 cycle before the PPG to make the total delay introduced by FIR filtering be \( n_1 \).

IV. SIMULATION AND SYNTHESIS RESULTS

A. General

Snapshot is nothing but every moment of the application while running. It gives the clear elaborated of application. It will be useful for the new user to understand for the future steps as shown in Figs.11 and 12.
A. Simulation Results

Fig.11. Proposed DLMS adaptive filter design simulation result.

B. Synthesis Results

TABLE I: Device Utilization Summary

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
<th>Note(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>118</td>
<td>7,108</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>770</td>
<td>7,108</td>
<td>10%</td>
<td></td>
</tr>
</tbody>
</table>

Logic Distribution:

- Number of occupied Slices: 417 / 3,564 (11%)
- Number of Slices containing only related logic: 417 / 417 (100%)
- Number of Slices containing unrelated logic: 0 / 417 (0%)

Total Number of 4 input LUTs: 770 / 7,108 (10%)

- Number used in logic: 770
- Number used as a router: 0
- Number of bonded IOBs: 59 / 141 (37%)
- IOB Flip Flops: 10
- Number of GCLKs: 1 / 8 (12%)

Total equivalent gate count for design: 7,014

Additional I/O gate count for IOBs: 2,844

TABLE II: Modification Part Design Summary:

<table>
<thead>
<tr>
<th>LMS/VIEX Partition Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>No partition information found.</td>
</tr>
</tbody>
</table>

V. CONCLUSION & DISCUSSION

We have proposed an efficient addition scheme for inner-product computation to reduce the adaptation delay significantly in order to achieve faster convergence performance and to reduce the critical path to support high input-sampling rates. Aside from this, we proposed a strategy for optimized balanced pipelining across the time-consuming blocks of the structure to reduce the adaptation delay and power consumption, as well. The proposed structure involved significantly less adaptation delay and provided significant saving of ADP and EDP compared to the existing structures. We proposed a fixed-point implementation of the proposed architecture, and derived the expression for steady-state error and we extended our work into the application of EEG.

Application:
- Digital communication & Signal processing applications.
- Digital radio receivers.
- Down converts.
- Software Radio.

Advantages:
- No fixed point issues.
- Less area requirement, Low delay.

Enhancement Work: We will apply this Delayed LMS adaptive filtering in the applications like EEG (electroencephalogram).

VI. REFERENCES


Author’s Profile:

Mr.Ramayanapu Vamsi Gopala Krishna has completed his B.Tech in ECE Department from Sri Aditya Engineering College, Jntu Kakinada. Present he is pursuing his Masters in VLSI DESIGN from Aditya College of Engineering And Technology, Surampalem, Andhra Pradesh, India.

Mr.N.Venkata Satish received the M.TECH degree from NIT Warangal. He is the Sr.Assistant professor in the Department of Electronics and Communication Engineering. He has 8 years of teaching experience.