AN OPTIMIZED IMPLEMENTATION OF LMS ADAPTIVE FILTER WITH LOW ADAPTATION DELAY IN FPGA TECHNOLOGY

B. LAKSHMI,
PG Scholar in VLSI System Design,
C. KARTHIK,
M.E, Asst. Professor, ECE Department,
lakshmibommala@gmail.com,
karthik.kamalesh@gmail.com.

Abstract: In this paper, the fir filter is proposed an efficient multiplication stage technique. To investigate the area, speed, power tradeoffs for implementation of FIR filters using MCM and digit-serial arithmetic. Multiple constant multiplications (MCM) are an efficient way of implementing several constant multiplications with the same input data. The coefficients of multiplier are expressed using shifts, adders, and subtractions. To introduce the Wallace tree multiplier for reduce both the number of adders and subtractions as well as the number of shifts. This method can be used to reduce the amount of embedded multipliers in large MCM blocks. We use Xilinx 14.5 to provide VHDL coding for our architecture. Result shows the better performance rate of our proposed work than existing algorithms. This paper is used to reduce the delay product and reduce the area and power consumption in FIR filtering.

Keywords: Multiple constant multiplication, Wallace tree architecture, RPAG algorithm, FIR filter, Pipelining, Partial product generation, Adder depth

I. INTRODUCTION

The Least Mean Square (LMS) adaptive filter is the widely used filter because of its simplicity and performance. Least mean squares (LMS) algorithm ms 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 stochastic gradient method in that the filter is only adapted based on the error at the current time. The LMS algorithm is the most popular method for adapting a filter, which have made it widely adopted in many applications. Applications include adaptive channel equalization, adaptive predictive speech coding, Noise Suppression and on -line system identification. Recently, because of the progress of digital signal processors, a variety of selective coefficient update of gradient-based adaptive algorithms could be implemented in practice. The Least Mean Square adaptive filter is used here because it differs from a traditional digital filter in the following ways: A traditional digital filter has only one input signal x(n) and one output signal y(n). An adaptive filter requires an additional input signal d(n) and returns an additional output signal e(n). The filter coefficients of a traditional digital filter do not change over time. The coefficients of an adaptive filter change over time. Therefore, adaptive filters have a self-learning ability that traditional digital filters do not have. The filter is an important component in the communication world. It can eliminate unwanted signals from useful information. However, to obtain an optimal filtering
performance, it requires ‘a priori’ knowledge of both the signal and its embedded noise statistical information. The classical approach to this problem is to design frequency selective filters, which approximate the frequency band of the signal of interest and reject those signals outside this frequency band.

Fig. 1: Adaptive filter system

II. EXISTING SYSTEM

This algorithm is 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 [1]. The LMS algorithm was devised for the study of a pattern recognition machine known as the adaptive linear element. The LMS algorithm is a stochastic gradient algorithm in that it iterates each tap weight of the transversal filter in the direction of the instantaneous gradient of the squared error signal with respect to the tap weight [2], which permits processing of data over a wide dynamic range, at temporal and hardware complexities significantly less than that of a floating-point processor [7]. The implementation of adaptive filters with fixed-point arithmetic requires to evaluate the computation quality. The accuracy may be determined by calculating the global quantization noise power in the system output [8]. The LMS algorithm is the most popular method for adapting a filter, which is used in many applications such as adaptive channel equalization, adaptive predictive speech coding, Noise Suppression and online system identification.

In existing system, they proposed fixed point adaptive filter with low adaptive delay for optimized balanced pipelining across the time-consuming combinational blocks of the structure. The adaptation delay of m cycles amounts to the delay introduced by the whole of adaptive filter structure consisting of finite impulse response (FIR) filtering and the weight-update process. Drawback of the existing is to slow convergence (due to Eigen-value spread). This length of time might cause problems in these applications because the adaptive filter must work in real time to filter the input signals. The steady-state behavior of the LMS algorithm, the step size is small. The DLMS algorithm is reduced the convergence speed and poorer tracking performance. It increases output latency.
III. DELAYED LMS ADAPTIVE FILTER

For every input sample, the LMS algorithm calculates the filter output and finds the difference between the computed output and the desired response. Using this difference the filter weights are updated in every cycle. During the n-th iteration, LMS algorithm updates the weights as follows:

\[ W_{n+1} = w_n + \mu \cdot e(n) \cdot x(n) \]  

(1a)

Where,

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

\[ y(n) = w^T_n \cdot x(n) \]  

(1b)

Here,

\( x(n) \) is the input vector\n\( w(n) \) is the weight vector of an Nth order LMS adaptive filter at the nth iteration, respectively, given by,

\[ x(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^T \]

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

\( d(n) \) is the desired response and \( y(n) \) is the filter output of the nth iteration. \( e(n) \) denotes the error computed in the nth iteration which is used to update the weights. \( \mu \) is the convergence-factor.

The DLMS algorithm, instead of using the recent most feedback-error \( e(n) \) corresponding to the n-th iteration for updating the filter weights, it uses the delayed error \( e(n-m) \), (i.e.) the error corresponding to (n-m)-th iteration for updating the current weight. The weight-update equation of DLMS algorithm is given by,

\[ W_{n+1} = w_n + \mu \cdot e(n-m) \cdot x(n-m) \]  

(2)

Where, \( m \) is the adaptation-delay.

IV. PROPOSED SYSTEM

In the conventional DLMS algorithm (Fig.1) the adaptation delay of \( m \) cycles amounts to the delay introduced by the whole of adaptive filter structure consisting of FIR filtering and weight adaptation process. But instead, this adaptation delay could be decomposed into two parts. One part is the delay introduced due to the FIR filtering and the other part is due to the delay involved in weight adaptation.

Based on such decomposition of delay, the
is shown in Fig. 5. The proposed adaptive filter architecture consists of two main computing blocks, namely the error computation block and weight update block. The computation of filter output and the final subtraction to compute the feedback error are merged in the error computation unit to reduce the latency of error computation path. If the latency of computation of error is n1 cycles, the error computed by the structure at the nth cycle is e(n − n1), which is used with the input samples delayed by n1 cycles to generate the weight -increment term. The weight update equation of the proposed delayed LMS algorithm is, therefore, given by,

\[ w_{n+1} = w_n + \mu \cdot e(n-n1) \cdot x(n-n1) \]  

(3a)

Where,

\[ e(n-n1) = d(n-n1) - y(n-n1) \]  

(3b)

And

\[ y(n) = w^T n-n2 \cdot x(n) \]  

(3c)

We can notice that during weight adaptation, the error with n1 delays is used while the filtering unit uses the weights delayed by n2 cycles. By this approach the adaptation -delay is effectively reduced by n2 cycles. The proposed algorithm can be implemented efficiently with very low adaptation -delay which is not effected substantially by the increase in filter order.

**Error computation block**

The error computation block is implemented by a pipelined inner-product computation module (Fig.5) and the weight update block is implemented by N number of pipelined multiply-accumulate units. The multiplications of input samples with corresponding weights followed by their summation in the filter computation block have been realized by a carry-save chain and the error computation is merged with the filter output computation. In the weight update block, the multiplication of the convergence -factor with the error and input values are combined with the addition with old weights according to (3a) to obtain the final updated weights in N parallel carry-save chains.

The function of weight-update block is shown in Fig.6. The convergence-factor is taken to be a negative power of two to realize the corresponding multiplication of (3a) by a shift operation.

The weight-update block consists of N carry -save units to update N weights. Each of those carry -save units performs the multiplication of shifted error values with the delayed input samples along with the addition with the old weights. Note that the addition of old weight with weight increment term is merged with multiplication pertaining to the calculation of weight -increment term.

The final outputs of the carry -save units constitute the updated weights which serve as an input to the error computation block as well as the weight-update block for the next iteration. A pair of delays are introduced before and after the final addition of the weight-update block to keep the critical-path equal to one addition time.

The shaded region in Fig.6 indicates the carry- save unit corresponding to the first weight which takes the delayed input samples and shifted from of delayed error value (to take care of the multiplication by convergence factor) and the old weights as input. Thus the weight-update block takes a total of two delays, i.e., n2 = 2.
Fig 7: Structure of Weight update block

V.PERFORMANCE RESULTS

This section evaluates the performance of the proposed modified least mean square (LMS) algorithm and shows the simulation results. The first result declares about the output of LMS adaptive filter with delay. It is having some delay in the output of Least Mean Square adaptive filter. And the second result declares about the output of LMS adaptive filter without delay. After the clock input has given the output of the adaptive filter is achieved without delay. The ModelSIM is the tool used here to check the performance of LMS adaptive filter. It is a complete HDL simulation environment that enables to verify the source code and functional and timing models using test bench.

Output of LMS Adaptive Filter With Delay

Fig 8: Output with delay

Output of LMS Adaptive Filter without Delay

Fig 9: Output without delay

CONCLUSION

In this paper implementation of low power digit-serial FIR filters using multiple constant multiplication (MCM) techniques has been considered. Some conclusions regarding design guidelines for low power digit-serial multiplier blocks can be deduced. The actual complexity in terms of adder cost and number of shifts is not the main factor determining the power consumption. Instead the adder depth, as for parallel arithmetic, is a main contributor. Hence, an algorithm with low adder depth should be used. Furthermore, the shifts prevent glitch propagation through subsequent adders. For even coefficients the shifts can be placed either before or after the final additions. Hence, a heuristic for placing the shifts would be also useful. Better results were achieved using RPAG compared to optimally pipelined adder graphs for FPGAs and the method presented in targeting ASICs. Furthermore, it was shown that the algorithm often finds better solutions for minimal total AD compared to existing system.
REFERENCES


