**Int. J. Nonlinear Anal. Appl.**
**Volume 12, Special Issue, Winter and Spring 2021, 1827–1833**
**ISSN: 2008-6822 (electronic)**
http://dx.doi.org/10.22075/ijnaa.2021.5893

# Design and implementation of low complexity LMS adaptive filter

V. Chowthri[a], A. Uma[b,*], P. Kalpana[b]

[a] *ME VLSI, ECE, PSG College of Technology, India*
[b] *Department of ECE,PSG College of Technology, India*

*(Communicated by Madjid Eshaghi Gordji)*

## Abstract

An adaptive filter is a real-time computational device that iteratively simulates the relationship between a filter's input and output signals. It is based on an adaptive algorithm that iteratively self-adjusts the linear filter coefficients to decrease the power of e. (n). The LMS method is one of the most widely used adaptive algorithms for adjusting the coefficients of adaptive filters, among others. The error-computation block and the weight-update block, which determine the filter's efficiency, are the two key computing blocks of the direct-form LMS adaptive filter. In this paper, adaptive filter is implemented in two different architectures namely, zero adaptation delay adaptive filter and two adaptations delay adaptive filter which results in low power consumption and less area complexity. Zero adaptation delay adaptive filter provides nearly 52% savings in the area and the delay decreases by 26% in two adaptations delay adaptive filter over the conventional adaptive filter. Hence based on the required speed and area for the application, any one of the proposed structures can be used.

*Keywords:* Adaptive filter, Least mean square algorithms, LMS adaptive filter, Adaptation delay, Area, Delay

## 1. Introduction

Signal prediction, signal augmentation, channel equalisation, system identification, and noise cancellation are all areas where adaptive filters are applied. In filtering applications, however, real-time actions are necessary. The transfer function of an adaptive filter is self-adjusted according to an optimum algorithm, and precision can be achieved by modifying its properties [1]. An adaptive filter

*Email addresses:* chowthri.v_psg@gmail.com (V. Chowthri), au.ece@psgtech.ac.in (A. Uma), pkalpana.ece@psgtech.ac.in (P. Kalpana)

*Received:* August 2021 *Accepted:* November 2021

is made up of two parts: a filter that processes the input signal $x(n)$ and an adaptive algorithm that updates the filter's coefficient weights. Fig. 1 shows the adaptive filter's block diagram, with $d(n)$ representing the desired response, $y(n)$ representing the filter's output response, and $e(n)$ representing the error difference between $d(n)$ and $y(n)$. The filter's coefficient weights will be updated by the adaptive algorithm until the filter's output response is equal to or substantially equivalent to the desired response [3].

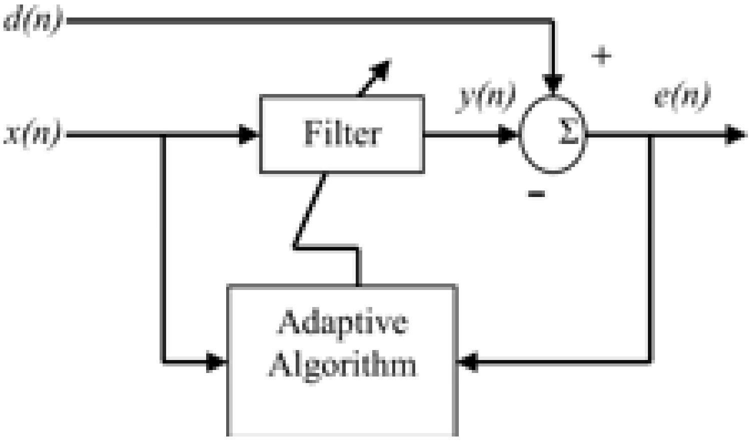


Figure 1: Block diagram of adaptive filter

To decrease signal error, adaptive algorithms are employed to alter the weights of adaptive equalisers towards an optimum configuration. It is chosen based on metrics such as pace of convergence, misadjustment, level of complexity, and numerical qualities, among others. Adaptive equalisation algorithms include zero forcing (ZF), least mean squares (LMS), Recursive Least Square (RLS), Kalman, and Least Square Lattice (LSL) [4]. LMS is the most extensively used adaptive filter algorithm because to its ease of use, rapid convergence, minimal complexity, and great performance. A quick convergence signifies that the adaptive filter calculates the filter coefficients in a shorter amount of time.

During each sampling period, the LMS algorithm is executed by adjusting the filter weights using the predicted error [7, 6]. The difference between the desired response and the current filter output is used to calculate estimated error. To update the coefficients of an adaptive FIR filter, the LMS algorithm performs the following steps:

1. Calculates the output signal y(n) from the FIR filter

$$y(n) = \sum_{j=1}^{p} \widehat{w_j}(n) x_j(n) \tag{1}$$

2. Updates the filter coefficients by using the following equation:

$$W(n+1) = \widehat{w}(n) + \eta[d(n) - x^T(n)\widehat{w}(n)]x(n) \tag{2}$$

Here $\eta$ represents the step size, $\widehat{w}(n)$ is the filter coefficient, $x(n)$ is the input signal to the filter and $n$ is the number of samples. Because area, time, and performance complexity are all escalating, an efficient implementation of the LMS adaptive filter is required. This paper covers the Verilog implementation of the LMS finite impulse response (FIR) adaptive filter as well as adaptive filter performance indicators.

The suggested architecture uses an efficient addition approach for the error computation and weight update blocks to reduce the adaption latency and the critical route to support high sampling rates.

In the next section, the structure of conventional adaptive filter is given and we have described the proposed architectures in section 3. Section 4 shows the simulation results of conventional filter and proposed architectures with the comparison in area and delay followed by conclusion in section 5.

## 2. Conventionaladaptive filter

A direct-form LMS adaptive filter's traditional adaptive filter structure consists of N multipliers. The multiplier has two inputs: one from the common tap delay line, which comprises of x(n-D) delayed input signals, and the other from the 2:1multiplexer. The error computation block, which receives the estimated error value multiplied by the step size, and the weight update block, which receives updated coefficient values, will feed the multiplexer [8]. The proposed structure also consists of N adders for calculating N weights. The adder tree gives the output of the filter by adding the outputs from N multipliers. In addition, a subtractor is required to calculate the error difference between the obtained and desired responses, as well as 2:1 de-multiplexer to transfer the product values to the adder tree block or the weight update block. The multiplexers and de-multiplexers are controlled by the clock signal. [5].

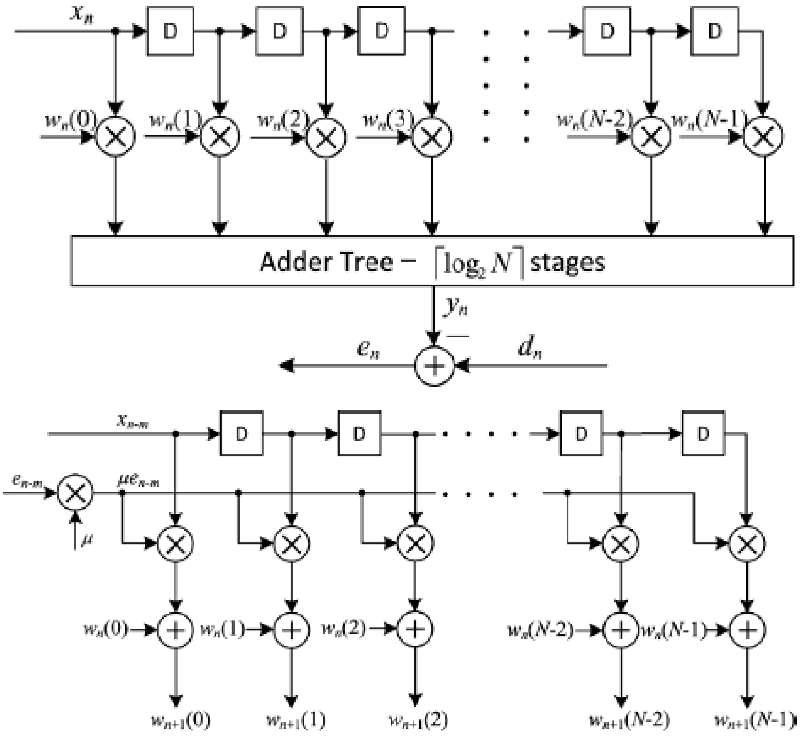


Figure 2: (i). Error computation block, (ii). Weight update block

The error computation and weight update blocks are presented in Fig. 2(i) and 2(ii), respectively, in a standard adaptive filter (ii). A pipelined latch is inserted after the error value is multiplied by the step size in the error computation block. The multiplication is done using the hardwired shift approach because the step size is a fraction of two. As a result, there is no register overhead with pipelining. The registers in each blocks will be the same.

## 3. Implementation of proposed structures

### *A. Zero adaptation delay adaptive filter*

Zero adaptation delay denotes that the computation of error value and weight updating takes place at the same clock cycle. This cannot occur concurrently because the structure is non-pipelined. Zero adaptation delay can be achieved by making the blocks to complete its operation within half clock cycle and multiplexing the same set of multipliers. Same set of registers are used for storing the computed values in both the blocks. The Structure of zero adaptation delay adaptive filter is shown in Fig 3.

The structure is built in such a way that every clock cycle, it should take a new input sample. As a result, the registers in the delay path process the input signal at each rising edge of the clock and remain unmodified for the duration of the clock period. During the first half of each clock cycle, the multiplexers send the weight values stored in registers to the multiplier. After multiplication, the multiplier sends the product words to the adder tree, which uses the de-multiplexers to calculate the filter output, and the subtractor computes the error value. After a right shift operation, the determined error value is broadcast to all multipliers in the weight update block.

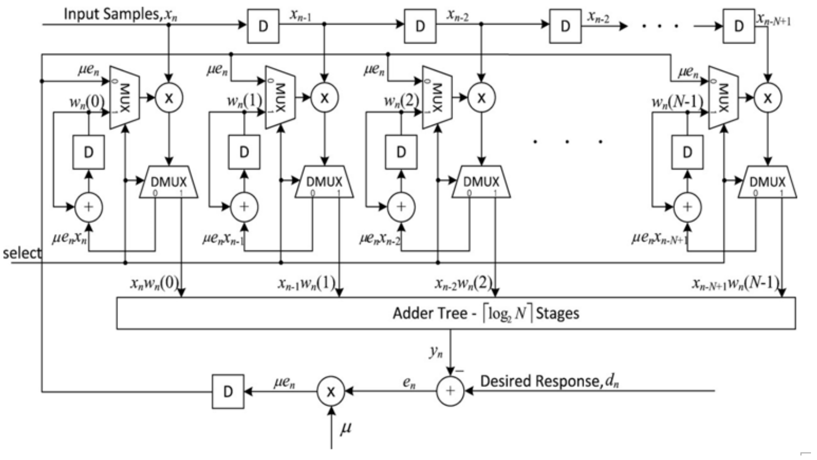


Figure 3: Structure of zero adaptation delay adaptive filter

The select line of the multiplexer and de-multiplexer decides the functionality of the filter. If it is zero, the filter will update the coefficient values else it will work as normal FIR filter. Since the error value is multiplied by the step size in error computation block and given to weight update blocks, the adaptation time of the filter is totally decided by the performance of those two blocks. The weight updation takes lesser time when compared to error computation, but we can't predict when the blocks will complete its working. Therefore, the structure is implemented in such a way that error computation is done first and weight updation is done at the second half cycle [2].

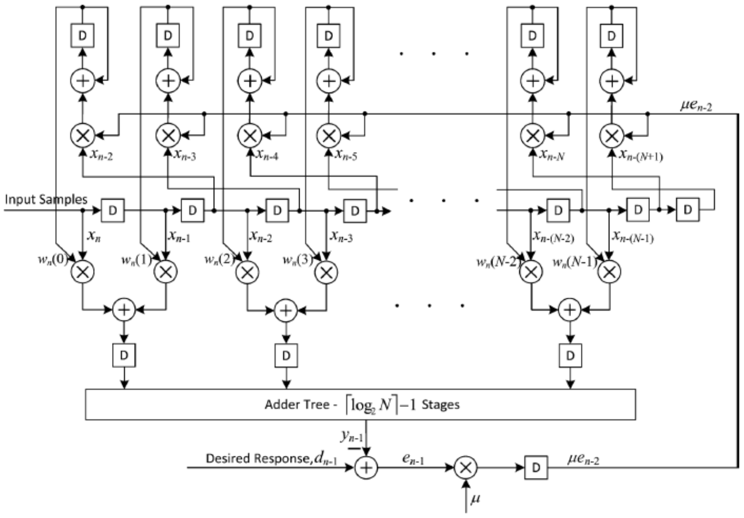


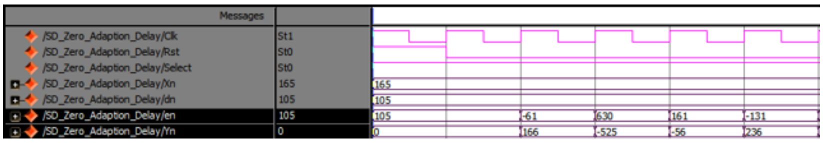
Figure 4: Structure of two adaptation delay adaptive filter

*B. Two adaptation delay adaptive filter*

Figure 4 depicts the structure of the proposed two adaptation delay LMS adaptive FIR filter. Three pipeline phases are used to implement it. In the error computation block, the first level of the adder tree appears in the first pipeline step. The remainder of the block is included in the following pipeline stage. The weight update block is where the third pipeline stage is created. Because of the pipelining, this structure requires more registers than a traditional adaptive filter. For calculating weights, this structure has N multipliers and N adders. To obtain the filter output, an adder tree is constructed by adding the outputs of multipliers. As in the zero adaption delay adaptive filter, the suggested structure also requires a subtractor to calculate the error value and 2:1 de-multiplexer to send the product values to the adder tree block or the weight update block. The multiplexers and de-multiplexers are controlled by the clock signal.

## 4. Simulation results

The conventional LMS Adaptive filter and the proposed structures are implemented using Verilog HDL. This design is simulated using Modelsim 6.4c and synthesized by Xilinx 13.2/9.1. The comparison is made on area and delay of conventional, zero adaptation delay and two adaptation delay adaptive filters. The simulation waveform of conventional adaptive filter is shown in Fig. 6. From the waveform, it is clear that if reset goes high then the error value and expected response value is same since the output value is zero. If reset goes low, the filter will calculate the error value and update the filter coefficient value with one adaptation delay that is with one clock cycle delay.



```
Messages
/SD_Zero_Adaption_Delay/Clk      St1
/SD_Zero_Adaption_Delay/Rst      St0
/SD_Zero_Adaption_Delay/Select   St0
/SD_Zero_Adaption_Delay/Xn       165   165
/SD_Zero_Adaption_Delay/dn       105   105
/SD_Zero_Adaption_Delay/en       105   105   -61   630   161   -131
/SD_Zero_Adaption_Delay/Yn       0     0     166   -525  -56   236
```

Figure 5: Simulation result of Conventional adaptive filter

The simulation result of zero adaptation delay adaptive filter is shown in Fig. 6. From the waveform, it is inferred that if reset is zero, the filter will calculate the error value and update the filter coefficient value without any delay. Hence the structure is called zero adaptation delay adaptive filter.

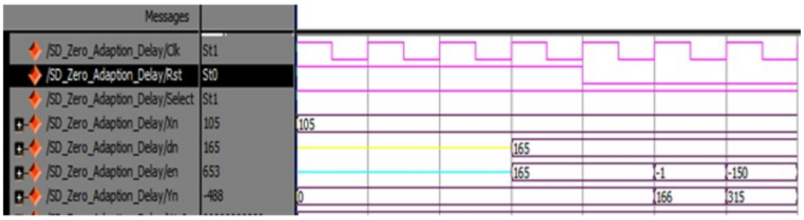

Messages
/SD_Zero_Adaption_Delay/Clk St1
/SD_Zero_Adaption_Delay/Rst St0
/SD_Zero_Adaption_Delay/Select St1
/SD_Zero_Adaption_Delay/Xn 105
/SD_Zero_Adaption_Delay/dn 165
/SD_Zero_Adaption_Delay/en 653
/SD_Zero_Adaption_Delay/Yn -488
105
165
165
-1
-150
0
166
315

Figure 6: Simulation result of zero adaptation delay adaptive filter

The simulation result of conventional adaptive filter is shown in figure 8. This structure calculates the error value and updates the filter coefficients with delay of two clock cycles. Hence the structure is called two adaptation delay adaptive filter.

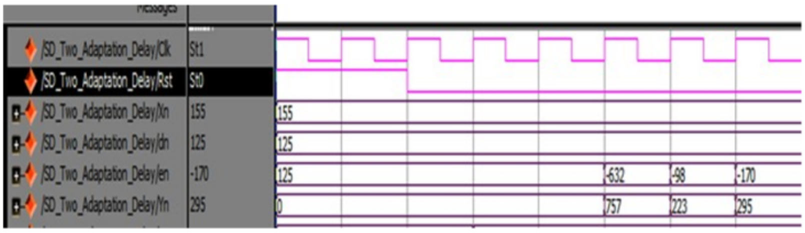

Messages
/SD_Two_Adaptation_Delay/Clk St1
/SD_Two_Adaptation_Delay/Rst St0
/SD_Two_Adaptation_Delay/Xn 155
/SD_Two_Adaptation_Delay/dn 125
/SD_Two_Adaptation_Delay/en -170
/SD_Two_Adaptation_Delay/Yn 295
155
125
125
-632
-98
-170
0
757
223
295

Figure 7: Simulation result of two adaptation delay adaptive filter

Table 1 shows the comparison between conventional, zero adaptation delay and two adaptation delay adaptive filters. From the comparison table, it is inferred that hardware requirement for zero adaptation delay adaptive filter is lesser than two adaptations delay adaptive filter at the cost of increase in delay.

Table 1: Comparison on area and delay

| | **Area** | | | **Delay** | | |
|---|---|---|---|---|---|---|
| **Method name** | **Gates** | **Slice FF** | **LU T** | **Max Delay (ns)** | **Gate Delay (ns)** | **Path Delay (ns)** |
| **Proposed structures**<br>A. Zero Adaptation Delay [5] | 17.253 | 78 | 165 | 21.205 | 13.879 | 7.125 |
| B. Two Adaptation Delay [5] | 33.017 | 98 | 99 | 17.241 | 13.118 | 4.288 |
| Conventional Adaptation Delay [5] | 33.441 | 33 | 110 | 23.303 | 16.855 | 6.448 |

## 5. Conclusion

Based on the area and delay analysis the low-complexity architectures for the LMS adaptive filter are proposed. The direct form LMS adaptive filter needs very less registers than the transpose-form adaptive filter so it converges faster than the transpose form adaptive filter, but the drawback is delayed weight adaptation and large area requirement. To overcome this, two different architectures of direct form LMS adaptive filter with i) zero adaptation delay adaptive filter, ii) two adaptation delay adaptive filter are proposed. Zero adaptation delay adaptive filter does not have any adaptation delay. It has the minimum area requirement when compared with the two adaptation delay adaptive filter, but the drawback is extra select pin and this architecture will not update the coefficient value automatically. So, it can be used in the areas where noise is very less like noise cancelling in AC electrical measurements. In two adaptation delay adaptive filter, path delay is very less and it will automatically update the coefficient value based on the error, so no need of extra pins to update the coefficient values. The weight update block in this structure will update the weight in two clock cycle. It can be used in application which require accurate data like noise removal in ECG signals.

## References

[1] W.A. Harrison, J.S. Lim and E. Singer, *A new application of adaptive noise cancellation*, IEEE Trans. Acoust. Speech, Signal Process. 34(1) (1986) 21–27.

[2] S. Haykin and B. Widrow, *Least-Mean-Square Adaptive Filters*, Hoboken, NJ, USA: Wiley-Interscience, 2003.

[3] R.H. Hearn, J.R. Zeidler, E. Dong and R.C. Goodlin, *Adaptive noise cancelling: principles and applications*, Proc. IEEE. 63(12) (1975) 1692–1716.

[4] H. Kaur and R. Talwar, *Performance comparison of adaptive filter algorithms for noise cancellation*, IEEE Signal Process Lett. 16(1) (2009).

[5] P. Kumar Meher and S. Yoon Park, *Critical-path analysis and low-complexity implementation of the LMS adaptive algorithm*, IEEE Trans Circuits Syst I Regul Pap. 61(3) (2014) 778–788.

[6] G. Long, F. Ling and J.G. Proakis, *The LMS algorithm with delayed coefficient adaptation*, IEEE Trans. Acoust., Speech. Signal Process. 37(9) (1989) 1397–1405.

[7] P.K. Meher and M. Maheshwari, *A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm*, Proc. IEEE Int. Symp. Circuits Syst. (2011) 121–124.

[8] M.D. Meyer and D.P. Agrawal, *Amodular pipelined implementation of a delayed LMS transversal adaptive filter*, Proc. IEEE Int. Symp. Circuits Syst. (1990) 1943–1946.