Delay-optimized realization of 2-parallel delayed LMS adaptive FIR filter

Ming Liu\textsuperscript{a),} Mingjiang Wang\textsuperscript{b),} De Liu, and Boya Zhao

Harbin Institute of Technology,
Shenzhen, Xili University Town, Shenzhen 518055, Guangdong, China
\textsuperscript{a)} lm hit 1986@126.com
\textsuperscript{b)} mjwang@hit.edu.cn, Corresponding Author

Abstract: Motivated by improvement of convergence characteristics and throughput, this work develops a delay-optimized VLSI realization of the adaptive filter based on the 2-parallel delayed LMS (PDLMS) algorithm. The proposed design uses a novel parallel FIR filter structure based on the fast FIR algorithm. The throughput of the proposed architecture is not only two times that of the traditional structure at the same frequency, but also the convergence characteristic is close to that of the LMS algorithm. The fine-grained dot-product unit, fine-grained fused multiply-add unit and multiple-input-addition unit are adopted to reduce the latency of critical path. From the ASIC synthesis results we find that the proposed architecture of an 8-tap filter has nearly 25\% less power and nearly 24\% less area-delay-product (ADP) than the best existing structure.

Keywords: 2-parallel DLMS algorithm, fined-grained, dot-product unit, fused multiply-add unit, multiple-input-addition unit

Classification: Integrated circuits

References


1 Introduction

In the digital signal processing field, adaptive filter has been used widely in system identification, adaptive noise cancelation (ANC), channel equalization, measure system, etc. [1, 4, 5, 7]. The most widely used algorithm for adaptive filters is the least-mean-square (LMS) algorithm due to its merits of superior performance and simple calculation. However, in practical applications, the throughput of the adaptive filter is the important factor for hardware implementation [10]. We presented the architecture with a high throughput and a good convergence characteristics based on the parallel delayed LMS algorithm.

Some researchers have made great efforts on the systolic architectures of the DLMS algorithm in [3, 9, 10]. B. K. Mohanty et al. [12] have proposed delayed block LMS algorithm and its efficient implementation. P. K. Meher [1] have proposed a direct form LMS adaptive filter with 2 adaptation-delay. Van et al. [13] have proposed a systolic structure using a tree rule for reducing the adaptation delay. Y. Yi et al. [3] have proposed a retimed 8-tap predictor system with the TDF-RDLMS architecture that has the delay of a multiplier on critical path.

In order to scientifically compare the speed of a variety of different computing units employed, we use logic level [2, 11] to estimate the latency of different circuits without limitation of manufacturing technology. Generally speaking, we assumed the XOR gate involves 2 logic levels of delay which is two times that of the AND/OR gate. Fig. 1 shows an 8-bit Kogge-Stone adder structure which has a delay of 10 logic levels consisting of 1 level of □ (2LL), 3 \((\log_2 8)\) levels of ■ (3 * 2LL = 6LL) and 1 level of ◊ (2LL). The N-bit Kogge-Stone adder involves roughly \((2 + \log_2 N + 2)\) logic levels of delay.
The rest of chapters are arranged as follows. In Sec. 2, we give a simple description of 2-parallel DLMS algorithm. The derivation process of the proposed architecture is given in Sec. 3. Sec. 4 presents the experimental simulation results and performance analyses. The conclusions are presented in Sec. 5.

2 The proposed 2-parallel delayed LMS algorithm

Before clarifying the content of the proposed algorithm, First of all, we give a brief introduction of the delayed LMS (DLMS) algorithm as the background knowledge of the pipelined DLMS algorithm.

2.1 DLMS algorithm

The iterative formula of DLMS adaptive filtering algorithm is given by

\[ y(n) = W^T(n)X(n) \]  
\[ e(n-m) = d(n-m) - W^T(n-m)X(n-m) \]  
\[ W(n+1) = W(n) + \mu e(n-m)X(n-m). \]

where \( N \) is equal to the filter length, \( m \) is the adaptation-delay, \( e(n-m) \) is the delayed error signal of the DLMS algorithm, \( \mu \) is the step size for adaptation of the weight vector, \( d(n) \) is the desired signal, the filter weight vector and the input sequence indicated by \( W(n) \) and \( X(n) \) respectively are expressed by

\[ X(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^T \]  
\[ W(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^T. \]

2.2 2-Parallel DLMS algorithm

Consider a length-N FIR filter that can be expressed with the general formulation as shown below.

\[ y(n) = h(n) \ast x(n) = \sum_{k=0}^{N-1} h(k)x(n-k), \quad n = 0, 1, 2, \ldots, \infty \]

where \( \{x(n)\} \) is an infinite length input sequence and \( \{h(n)\} \) are the N-tap FIR filter coefficients. It is generally known that parallel processing is an effective way to reduce power consumption. Using the polyphase decomposition technique [6], equation (6) will be changed into 2-parallel form expressed as

Fig. 1. The Kogge-Stone adder (8-bit).
\[ Y = Y_0 + z^{-1}Y_1 \]
\[ Y_0 = H_0X_0 + z^{-2}H_1X_1 \]
\[ Y_1 = (H_0 + H_1)(X_0 + X_1) - H_0X_0 - H_1X_1 \]
where \( H_0 \) and \( H_1 \) are the even term and odd term of \( H \), respectively.

We can get an efficient 2-parallel adaptive filter architecture based on fast FIR algorithm as shown in Fig. 2. We use only one path to realize adaptive processing in order to reduce computational complexity. The single path 2-parallel PDLMS can be described as follows:

\[
\begin{bmatrix}
  h_{0,j} \\
  h_{1,j}
\end{bmatrix} = \begin{bmatrix}
  h_{0,j-1} \\
  h_{1,j-1}
\end{bmatrix} + \mu e_{0,j-D} \begin{bmatrix}
  X_{1,j-D} \\
  X_{0,j-D}
\end{bmatrix}
\]

where subscript 0 and 1 represent the even and odd branch signals, respectively. The input sequence \( X_{0,j} \) and \( X_{1,j} \) are given by

\[ X_{0,j} = [x_{2j} \ x_{2j-2} \ \ldots \ x_{2j-N}]_{x^2} \]
\[ X_{1,j} = [x_{2j+1} \ x_{2j-1} \ \ldots \ x_{2j+1-N}]_{x^2} \]

From Fig. 2, we can see that the proposed algorithm consists of 2-by-2 FIR block, weight update block and error calculation block. The complexity of this algorithm is nearly the same as that of DLMS algorithm, especially the weight coefficient calculation process.

### 3 The proposed architecture

Before give the concrete structure, we first introduce several necessary technique such as multiple-operand-addition, fused-multiply-add unit and the fine-grained Dot-product-unit. These three techniques are important means to reduce the delay of critical path. Note that, in the following, \( t_{add} \) and \( t_{mul} \) are the delay consumed on the addition and multiplication operation. In order to ensure the accuracy of the algorithm, the word-length of input sequence is selected to be 16.
3.1 Multiple-operand-addition

The easiest way to achieve the sum of \( l \) N-bit words is that we use \( l-1 \) carry-propagate adders (CPA). Owing to the use of Kogge-Stone adder as the CPA, the delay of 4-input addition is \( 2 + (\log_2 4) \times 2 + 2 \times 3 = 24 \) logic levels. The most popular method to reduce the delay is adopting \([4:2]\) compressor as shown in Fig. 3(a). Because the \([4:2]\) compressor has 6 logic levels, the delay of 4-input addition become 14 logic levels shown in Fig. 3(b).

How to choose optimal structure becomes a very practical problem with the increase of number of operand. The \( N \)-operand adder adopting wallace-tree structure is achieved by \( \lceil \log_3 2^N \rceil \) levels CSA and one level of CPA, where \( \lceil x \rceil \) represents the smallest integer greater than \( x \). Based on the above arithmetic, the 6-input addition needs 3 levels of CSA which have \( 4 \times 3 = 12 \) logic levels. For \([4:2]\) compressor-tree structure, it needs only \( \lceil \log_2 N/2 \rceil \) levels of \([4:2]\) compressor. However, the improved architecture consisting of CSA and \([4:2]\) compressor has only 10 logic levels before enter CPA as illustrated in Fig. 3(c). From the above example, we can see that the \([4:2]\) compressor combined with CSA is sometimes an optimal structure.

In VLSI design, the issue of regular structure is particularly important. Although \([4:2]\) compressor-tree structure using the \([4:2]\) compressor has a more regular layout, it cannot meet the delay requirements when the number of operand is not equal to power of 4. In addition, from the perspective of power consumption, we should use \([4:2]\) compressor as far as possible in the multi-operand-addition.

3.2 Fused multiply-add

Many DSP algorithms, especially in the field of communication, require computing \( Y = A \times B + C \). Although this arithmetic can be implemented with a multiplier and an adder, it is much faster to use a fused multiply-add unit. Fig. 4 shows an example of a 16 \( \times \) 16 multiplier which has 9 partial products. The nine rows partial product can be generated through Booth encoding as shown in Fig. 5. The fused multiply-add unit use the design idea of a booth multiplier to accept another input \( C \) that is summed just like the other partial products. Owing to the addition of only one partial product, the delay increases by just one extra CSA compare to a multiplier.

The only difference being that the compressor of the fused multiply-add unit has 2 logic levels more than that of an ordinary multiplier as shown in Fig. 6. The overall architecture of fused multiply-add consisting of booth-encoding module,
[4:2] compressor tree module and CPA module is shown in Fig. 5(a). Because the architecture of fused multiply-add is nearly the same as that of booth multiplier except the compressor tree, the area of the fused multiply-add is approximately equal to that of booth multiplier when word-length is relatively large.

The delay analysis for the fused multiply-add unit:

- Partial Product Generation. The new Booth coding structure is used for this block. Since this block contains 3 levels of basic gate level circuits, the delay of this module is about four logic levels—$4L$.
- [4:2] Compressor Tree. There are 10 partial products for the signed multiply-add operation. This module contains one level of full adder and two levels of [4:2] compressor. Because the critical path of full adder is two levels of XOR gate, it has a delay of four logic levels. As we mentioned above, the delay of one level of [4:2] compressor is six logic levels. We can get the delay for this module is $(4 + 6 \times 2)$ logic levels—$16L$.
- The CPA Module. This module is a 32-bit carry-propagate adder implemented by Kogge-Stone adder mentioned above. The delay of this module is nearly $(2 + 2 \times \log_2 32 + 2)$ logic levels—$14L$.

According to the delay analysis of the previous 3 modules, we can draw the conclusion that the delay of proposed fused multiply-add unit has a delay of 34 logic levels—$34L$.
3.3 The fine-grained structure of dot-product-unit

In addition to inserting the register to shorten the critical path, we have designed a fine-grained dot-product unit to reduce the delay of the critical path further. Fig. 7 shows a complete structure of the dot-product unit composed of 4 parts. The function of this dot-product unit is to perform the operation of \( A1 \times B1 + A2 \times B2 + A3 \times B3 + A4 \times B4 \).

Using traditional method to implement the dot-product operation, the delay of the critical path is \( t_{mul} + 2 \times t_{add} \). As mentioned above, the delay of the 16 \( \times \) 16 multiplier is roughly \( (4 + 14 + 14) = 32 \) logic levels. The critical path of dot-product unit proposed by P. K. Meher [1] has a delay of \( (32 + 2 \times 14) = 60 \) logic levels based on the previous assumption.

From Fig. 7, we can see that the only difference between the dot-product unit and the booth multiplier is the structure of the compressor tree. Although there are four identical partial product generation units, these units are implemented in parallel. This module has the same delay as that of PPG unit.

![Diagram of dot-product unit](image_url)
Because each PPG module can produce nine partial products, there are \(4 \times 9 = 36\) partial products for the dot-product unit. The [4:2] compressor tree of the dot-product unit consists of three levels of [4:2] compressor and two levels of CSA. Since the [4:2] compressor circuit has a delay of 6 logic levels, the delay for this module is \(6 \times 3 + 4 \times 2\) = 26 logic levels.

Through the above analysis of the delay in terms of logic level, we can get the conclusion that the delay of the proposed dot-product unit has a delay of \(4 + 26 + 14\) = 44 logic levels—44L.

### 3.4 Implementation of 2-parallel DLMS algorithm

In order to reduce the computational complexity, the step size \(\mu\) is set to be a power of two. Through doing this, the multiplication operation can be replaced with shift operation. We give an example of an 8-tap adaptive filter structure for delay analysis as shown in Fig. 8. The derivation process of proposed design is divided into two steps.

In the first step, we firstly given the coarse-grained architecture for analyzing delay intuitively. It is generally known that the convergence characteristics deteriorate with the increase of the number of adaptation delay [8]. The proposed architecture has only one adaptation delay, which can ensure good convergence rate just like that of LMS algorithm. As we mentioned that the bit-width used is set to be 16, such as the input sequence \(x(n)\), desired sequence \(d(n)\), the weight coefficient \(w(n)\) and so on. In Fig. 8, the critical path in the first step is given by

\[
T = t_{\text{mul}} + \left(\frac{N + 4}{2}\right) \times t_{\text{add}}
\]

(12)

where \(N\) is the filter length of DLMS algorithm for the comparison with uniform criteria. The delay will become very large with the increase of filter order. To shorten the critical path further, we give a fine-grained architecture as shown in Fig. 9 in the second step.

The most commonly used methods of reducing critical path is inserting registers. In this Figure, we have added only an adaptation delay compare to Fig. 8. We have also deigned a fine-grained dot-product unit and a fine-grained fused multiply-add unit to reduce delay further. There are two grid registers at the

![Fig. 8. The structure of 2-parallel DLMS adaptive filter (8-tap) with one adaptation-delay.](image-url)
output of the $MOA(6)$ block at first. We use the retiming technique to move the two register to the six input signals of the $MOA(6)$ block, which can reduce the critical path but increase by four registers.

The critical path in the second step is given by

$$T = t_{\text{ppg}} + t_{[4:2]-\text{tree}} + \left\lfloor \log_2 \frac{N}{8} \right\rfloor \ast t_{[4:2]} + t_{\text{moa4}}$$

(13)

where $t_{\text{ppg}}$ stands for delay of PPG unit, $t_{[4:2]-\text{tree}}$ represents the delay of [4:2] compressor-tree block and $t_{[4:2]}$ represents the delay of [4:2] compressor. The proposed architecture involves additional 10 registers to the architecture with zero adaptation delay.

**Critical path analysis for 2-Parallel DLMS with 1 adaptation delay in Fig. 9**

- **The CPA Module.** This module is used to calculate the sum of the two weight coefficients from two parallel branches. The 16-bit CPA has a delay of 12 logic levels—12L.

- **The PPG Module.** The 16-bit PPG unit implemented by the booth-encoding technique as shown above has a delay of 4 logic levels—4L.

- **The [4:2] Compressor-tree adder.** There are three levels of [4:2] compressor and two level of [3:2] adder. So this module has a delay of $(6 \times 3 + 4 \times 2) = 26$ logic levels—26L.
From the detailed analysis above, we can conclude that the critical path has a delay of 42 logic levels—$42L$.

4 Results and comparisons

To prove its effectiveness, we present the most Area-Delay-Power efficient structure in the recent works proposed by P. K. Meher in [1]. Fig. 10 shows the overall 8-tap architecture with the same assumption above. The critical-path in this figure is marked with the red line.

Fig. 10. The structure for 2-adaptation-delay DF-LMS adaptive filter.

Critical path analysis for [1] with 2 adaptation delay in Fig. 10

- **The Binary-tree adder.** This module has $\lceil \log_2 N/2 \rceil$ levels of CPA. Because the word-length for CPA is 32, the delay of CPA as mentioned above is 14 logic levels. This module has a delay of $14 \times 2 = 28$ logic levels—$28L$.

- **The Error Calculation Module.** This module is implemented by a 16-bit subtractor for computing the error: $e(n-1)$. Only one level of XOR logic more than adder, so this module has a delay of $(2 + \log_2(16) \times 2 + 2) + 2 = 14$ logic levels—$14L$.

- **The Right Shifter.** This module is implemented by a barrel shifter as shown in Fig. 11. The barrel shifter has 4 levels of multiplexer with a delay of 2 logic levels, so the delay of this module is $2 \times 4 = 8$ logic levels—$8L$.

Based on the delay analysis above, the critical path has a delay of 50 logic levels—$50L$. 
Table I. Comparison of hardware and time complexities

<table>
<thead>
<tr>
<th>Design</th>
<th>Critical-Path</th>
<th>AD (m)</th>
<th>Number of calculator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ADD</td>
<td>MUL</td>
</tr>
<tr>
<td>K. K. Parhi [7]</td>
<td>$(N + 1)\tau_{\text{add}} + 3\tau_{\text{mu}}$</td>
<td>0</td>
<td>$2N + 1$</td>
</tr>
<tr>
<td>Long et al. [4]</td>
<td>$\tau_{\text{add}} + \tau_{\text{mul}}$</td>
<td>$1 + \log_2N$</td>
<td>$2N$</td>
</tr>
<tr>
<td>Y. Yi et al. [3]</td>
<td>$\tau_{\text{mul}}$</td>
<td>$2 + \log_2N$</td>
<td>$2N$</td>
</tr>
<tr>
<td>Meher et al. [1]</td>
<td>$\tau_{\text{add, prec}} + \tau_{\text{add}} + \tau_{\text{shift}}$</td>
<td>2</td>
<td>$2N$</td>
</tr>
<tr>
<td>Van&amp;Feng [13]</td>
<td>$\tau_{\text{add}} + \tau_{\text{mul}}$</td>
<td>$N/4 + 3$</td>
<td>$2N$</td>
</tr>
<tr>
<td>This work</td>
<td>$\tau_{\text{dot}} + \log_2(N/8) \times (t_{\text{add}} + \tau_{\text{mu add}})$</td>
<td>2</td>
<td>$N/2 + 3$</td>
</tr>
</tbody>
</table>


Fig. 12. The convergence of adaptive filter for different designs.

Table I shows the comparison of hardware and the time complexities for the proposed design and other ones. From this table, we can see that the architecture proposed by Y. Yi et al. [3] has the shortest critical path, but it has the largest registers generating more power. The size of the multiplier is nearly the same as that of fused multiply-add unit, so the number of multipliers can be changed by the number of fused multiply-add unit. We can see that the proposed architecture requires less than half the amount of calculations of addition required by the conventional architectures. The $\tau_{\text{dot}}$ of proposed design is the delay consisting of the PPG unit and [4:2] compressor-tree unit except the last CPA module, so it has less latency than a multiplier.
Simulation on computer as a key step is used to prove the correctness of the proposed algorithm as shown in Fig. 12. We use the same size convergence factor $\mu$ to distinguish the performance of different algorithms containing the direct-from (DF) and the transpose-form (TF). The input signal $x(n)$ is a mixed signal consisting of the trigonometric sine function signal and the white Gaussian noise with $\text{SNR} = 15\,\text{dB}$. From the simulation results, it is easy to see that the convergence characteristics of proposed architecture is nearly the same as that of the LMS algorithm. The best existing architecture proposed by P. K. Meher [1] has the same convergence characteristics as the DLMS algorithm so that the proposed design has better convergence characteristics than that of P. K. Meher [1].

The proposed design and other ones is realized in Verilog HDL included in the Table II. To reflect the advantages of delay optimization, we have compared to the design of P. K. Meher [1] in terms of logic levels. The critical path of proposed architecture has a delay of 42 logic levels (42L) which is 16% lower than that of P. K. Meher (42 VS 50), which is nearly identical with DC synthesis comparison results for the DAT. Applying the same timing constrains to all designs in DC synthesis, DC will give an automatic optimization for them. Although the design presented by Y. Yi [3] has the shortest CP with only a multiplier, DC will give a relatively large optimization which led to the mismatch between the logic levels and the DAT. Compared to the architecture proposed by P. K. Meher, the proposed design saves nearly 9% of area (46753 VS 51649) and 25% of power (7.43 VS 9.93) and estimated at DAT. It is found that the proposed architecture involves nearly 24% less ADP than the best existing work designed by P. K. Meher for the 8-tap filter.

5 Conclusion

This paper proposed a delay-optimized architecture using fine-grained dot product unit, fused multiply-add unit and multi-operand-addition for the 2-parallel DLMS algorithm. The proposed algorithm has a nearly equal convergence behavior with the LMS algorithm. Based on the detailed analysis above, the proposed design has the shorter critical path than that of the design of P. K. Meher in terms of logic levels (42 VS 50). Owing to utilize of the three fine-grained unit mentioned above, the DAT of 2PDLMS architecture is reduced by 16.5% compared to that of P. K. Meher. Furthermore, the proposed design involves nearly 24% less ADP and nearly 25% power than that of the best existing structure. The proposed design can be achieved with 2-adaptation-delay, small hardware area and lower power consumption, simultaneously.

Acknowledgments

This work is supported by the project of Shenzhen Infrastructure layout (Grant No.: JCYJ20150827165024088) and the project of supporting platform in Guangdong Province (Grant No.: 2014B0909B001).