Abstract—In this paper, an efficient distributed arithmetic (DA) formulation for the implementation of least mean squared (LMS) adaptive filter for hardware realization. Conventional-distributive DA will be replaced by coefficient DA for more appropriation to use with adaptive filter that the filter coefficients are varied by time. In existing architecture look-up table (LUT) was replaced by 4-bit adder network and 16:1 multiplexer structure, but the structure of adder network and multiplexer will be increased when filter length is so high. In the proposed architecture adder network and multiplexer structures are reduced and which leads to reduction of hardware complexity. Moreover, reduction of latency and power consumption will be achieved. The target architecture is designed, verified and simulated with Verilog HDL. For power consumption, latency and area of target architecture is synthesized in cadence digital lab.

Keywords: Adaptive filters, distributed arithmetic (DA), LMS algorithm.

I. INTRODUCTION

Adaptive filters are widely used in various signal processing applications, such as echo cancellation, noise cancellation, channel equalization and system identification etc. Amongst the existing ADFs, least mean square (LMS) based finite impulse response (FIR) adaptive filter is the most popular one due to its inherent simplicity and satisfactory convergence performance.

Distributed arithmetic (DA) was proposed about two decades ago and has used widely in VLSI Implementations of DSP architectures. Most of these applications are computation intensive with multiplication and/or addition being the predominant operation. The main advantage of distributed arithmetic approach is that it speeds up the multiply process by pre-computing all the possible medium values and storing these values in a ROM. The input data can then be used to directly address the memory and the result [1].

DA is used for implementation of both fixed-coefficient digital filter and adaptive filter (coefficient vary in time) in many styles [2-4]. Partial products can be pre-computed and stored in look-up table (LUT). The input samples will be distributed into bit level and used as LUT addressing, and then contents in LUT will be accumulated by scaling accumulator to produce the output. Consequently, multiplier-less computation can be achieved. DA is used for implementation of both fixed-coefficient digital filter and adaptive filter (coefficient vary in time) in many styles [2-4].

Distributed arithmetic (DA) technique is two types based on characteristics of coefficients.

- Conventional DA.
- Coefficient distributive DA.

However, since the filter coefficients of adaptive filter vary by time, if the conventional DA is used then the contents in LUT require updating which leads to more complexity of computation and circuits. The proposed in [5] uses auxiliary LUT to relax the complexity of LUT updating but it requires 2 LUTs and also need the time for updating. Coefficient-distributive DA is considered in [6], contents in LUT are different from conventional DA. The contents in LUT are sum of input samples that correspond to the filter coefficients in bit-level, and coefficients in bit-level will be used for LUT addressing. The extra circuit and extra time are need for LUT updating.

In this paper, the LUT-free DA implementation for LMS adaptive filter will be proposed. Behind concept is still based on coefficient-distributive DA, but the LUT will be replaced by 2 bit adder network and two 2:1 multiplexers structure. Therefore, multiplier less LMS adaptive filter can be obtained by DA implementation and also without LUT, and as a result do not require time for LUT updating. In section II, conventional DA is reviewed. Coefficient-distributive DA concept and the proposed hardware architecture using adder network and multiplexer structure for LMS adaptive filter will be explained in section III and IV respectively. Simulation results will be shown in section V. Finally, section VI is the conclusion.

II. CONVENTIONAL DA

Conventional distributive arithmetic is suitable for implementation of fixed coefficient digital filters. In digital filter implementation using DA, the input samples will be distributed into bit-level and used for LUT addressing. The contents in LUT are partial products of the impulse response and input samples in bit-level. Consider N-tap FIR filter,
\[
y(n) = \sum_{k=0}^{N-1} h_k x(n-k) \quad (1)
\]

The input samples will be represented by \(B\)-bit in 2's complement format, which sign bit is MSB (Most Significant bit).

\[
x(n-k) = -x_k + \sum_{j=1}^{B-1} x_j 2^{-j} \quad (2)
\]

Replace (2) in (1), the output \(y(n)\) is

\[
y(n) = -\sum_{k=0}^{N-1} h_k x_k + \sum_{j=1}^{B-1} \left[ \sum_{k=0}^{N-1} h_k x_k \right] 2^{-j} \quad (3)
\]

Where \(c_j = \sum_{k=0}^{N-1} h_k x_k, J=1 \ldots B-1, \quad c_0 = -\sum_{k=0}^{N-1} h_k x_k\)

Therefore, from (3) can be rewritten as

\[
y(n) = \sum_{j=0}^{B-1} c_j 2^{-j} \quad (4)
\]

The values of can pre-compute and store in LUT, the input samples in bit-level will be used for LUT addressing. Based on the DA structure, the multiplier less digital filter can be realized.

**III. THE PROPOSED COEFFICIENT-DISTRIBUTIVE DA**

Coefficient distributive DA is very suitable for the digital filter which filter coefficients are varied (change with time). In adaptive filter, the filter coefficients are time-varying, that means the contents in LUT will be updated in each iteration. This leads to high complexity and high latency for adaptive filter. In this paper, the idea of conventional DA (coefficients are lumped, input sampled are distributed into bit-level) is swapped. The filter coefficients are distributed into bit-level in \(B\)-bit 2's complement format and the input samples are lumped, i.e.

\[
h_k = -h_{k0} + \sum_{j=0}^{B-1} h_k 2^{-j} \quad (5)
\]

Replace (5) in (1), then the output \(y(n)\) is

\[
y(n) = -\sum_{k=0}^{B-1} (x(n-k))h_{k0} + \sum_{j=1}^{B-1} \left[ \sum_{k=0}^{N-1} x(n-k) h_k \right] 2^{-j} \quad (6)
\]

When compared with (3), the contents in LUT are sum of input samples that correspond to the filter coefficients in bit level, and coefficients in bit-level will be used for LUT addressing. This concept is also used in [6], but it also needs updating in LUT (read/write process is required). However, this style of DA realization is more suitable than conventional DA for adaptive filter implementation because updating of LUT contents is independent from coefficients updating algorithm. From the term in bracket in (6), the contents in LUT are shown in Table I for \(N=4\).

Since the process is still require the LUT, and then modes of operation consist of filtering mode (read process) and updating contents in LUT mode (write process). Although, updating of filter coefficients can be operated simultaneously with the LUT updating, but still need extra time for an update anyway (for read old data and write the new data into LUT). The 4-bit adder network and 16:1 multiplexer structure for coefficient-distributive DA is used in existed architecture. The filter coefficients in bit-level will be used as the select signal for multiplexer. The adder network is used to generate the input of multiplexer. The output of multiplexer is the input of scaling accumulator in order to compute the filtering output \(y(n)\). The structure of adder network that used for 4-tap \((N=4)\) FIR filter is shown in Fig. 1. But the structure of adder network and multiplexer will be increased when filter length is so high.

The 2-bit adder network and two 2:1 multiplexer’s for coefficient-distributive DA is used in proposed architecture. The coefficient-distributive DA by the 2-bit adder network and two 2:1 multiplexer’s structure does not require any LUT. Therefore, for adaptive filter implementation, we do not need the extra time or extra circuit for LUT updating.

**IV. LMS ADAPTIVE FILTER**

The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff in 1959. It is an adaptive algorithm, which uses a gradient-based method of steepest descent. LMS algorithm uses the estimates of the gradient vector from the available data. LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which leads to the minimum mean square error. Compared to other algorithms LMS algorithm is relatively simple. The updating equation for filter coefficients using LMS algorithm is

\[
h_k(n+1) = h_k(n) + \mu e(n)x(n-k) \quad (7)
\]

\(\mu\) is convergence factor, \(e(n)\) is the difference between desired signal \(d(n)\) and actual output \(y(n)\). The error signal \(e(n)\) is defined by

\[
e(n) = d(n) - h(n)x(n) \quad (8)
\]

Where \(h(n) = [h_0(n)h_1(n) \ldots h_{N-1}(n)]\) and

\[
x(n) = [x(n)x(n-1) \ldots x(n-(N-1))]^T
\]
Contents in LUT will be generated by adder network in Fig. 1. The filter coefficients in each row of LUT are used as address. From (5)-(8) and Fig. 1, the proposed 4-tap LMS adaptive filter by coefficient-distributive DA without LUT can be shown in Fig. 2.

The steps of operation can be explained as follows:

1. Input sample \( x(n) \) will be loaded to parallel-in parallel-out (PiPO) shift registers to make the delayed input samples for adder network. The adder network as shown in Fig. 1 will generate the input for multiplexer.

2. The filter coefficients that stored in buffers will be loaded to parallel-in serial-out (PISO) shift registers then signal \( clk \) will shift the filter coefficients into bit-level and used as control signal of multiplexer.

3. Each output of multiplexer is the input of scaling Accumulator, which is controlled by signals \( s/a \), \( lac \), and \( clac \). For the last bits (sign bit) of filter coefficients in bit level, the signal \( s/a \) will change mode of operation from adder to subtractor, then the filtering output can be produced and loaded to the buffer by signal \( lr \).

4. The error signal \( e(n) \) is generated from subtractor according to (8), and then loaded to buffer in the same time with clearing accumulator which is controlled by signal \( clac \).

5. Error signal will be right shifted \( L \)-bit; this operation represents multiply by convergence factor \( \mu \) to produce the term \( \mu e(n) \). That means the value of \( \mu \) will be \( 2^{-p} \) only for our design. In our experiment, \( \mu = 2^{-4} \) is used for testing.

6. The value of \( \mu e(n) \) will be quantized into the form of \( 2^{-p} \), where \( p \) is integer which can give \( 2^{-p} \approx \mu e(n) \). This quantization has a little effect for the convergence of adaptation [5].

7. The approximation of \( e(n) \) by \( 2^{-p} \) will be used for control the barrel shifter in order to shift the input samples to produce the term \( \mu e(n)x(n-k) \).

8. The values of \( \mu e(n)x(n-k) \) will be sent to add with the old filter coefficients and then store into the buffers which are controlled by signal \( upd \). Repeat step 1-8 continuously.

Table 1: LUT for Coefficient Distributive DA

<table>
<thead>
<tr>
<th>Address</th>
<th>Coefficients</th>
<th>Contents in LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1</td>
<td>( x(n) )</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0</td>
<td>( x(n-1) )</td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1</td>
<td>( x(n)+x(n-1) )</td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0</td>
<td>( x(n-2) )</td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1</td>
<td>( x(n-2)+x(n) )</td>
</tr>
<tr>
<td>6</td>
<td>0 1 1 0</td>
<td>( x(n-2)+x(n-1) )</td>
</tr>
<tr>
<td>7</td>
<td>0 1 1 1</td>
<td>( x(n-2)+x(n-1)+x(n) )</td>
</tr>
<tr>
<td>8</td>
<td>1 0 0 0</td>
<td>( x(n-3) )</td>
</tr>
<tr>
<td>9</td>
<td>1 0 0 1</td>
<td>( x(n-3)+x(n) )</td>
</tr>
<tr>
<td>10</td>
<td>1 0 1 0</td>
<td>( x(n-3)+x(n-1) )</td>
</tr>
<tr>
<td>11</td>
<td>1 0 1 1</td>
<td>( x(n-3)+x(n-1)+x(n) )</td>
</tr>
<tr>
<td>12</td>
<td>1 1 0 0</td>
<td>( x(n-3)+x(n-2) )</td>
</tr>
<tr>
<td>13</td>
<td>1 1 0 1</td>
<td>( x(n-3)+x(n-2)+x(n-1) )</td>
</tr>
<tr>
<td>14</td>
<td>1 1 1 0</td>
<td>( x(n-3)+x(n-2)+x(n-1)+x(n) )</td>
</tr>
<tr>
<td>15</td>
<td>1 1 1 1</td>
<td>( x(n-3)+x(n-1)+x(n-2)+x(n) )</td>
</tr>
</tbody>
</table>
Figure 2: The existing DA LMS adaptive filter architecture.

Figure 3: The proposed DA LMS adaptive filter architecture.
In proposed architecture 2-bit adder network and two 2:1 multiplexers have been used instead of 4-bit adder network and 16:1 multiplexer for 4-tap filter, which leads to reduce the complexity of adder network when filter length is high. The least two significant bits of PISO register are used as address for the 2-bit adder network and most two significant bits of PISO register are used as selected signals for two 2:1 multiplexers respectively. The output of multiplexer is input for the accumulator.

V. Simulation Results

Verilog is used for the implementation of proposed adaptive filter and behavior of circuit. The Verilog simulation result of proposed LMS adaptive filter is shown in fig.4.

![Verilog simulation result](image)

Figure 4: Verilog simulation result.

<table>
<thead>
<tr>
<th>Architecture name</th>
<th>Latency(ps)</th>
<th>Area(sq.um)</th>
<th>Power consumption(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing LMS adaptive filter</td>
<td>3582</td>
<td>7797</td>
<td>1045060</td>
</tr>
<tr>
<td>Proposed LMS adaptive filter</td>
<td>1985</td>
<td>7198</td>
<td>1006219</td>
</tr>
</tbody>
</table>

VI. Conclusion

The proposed LMS adaptive filter architecture for reduction of hardware complexity, latency and power consumption. By using proposed architecture 44% of latency and 7% of area will be decreased. Coefficient DA method is used for the implementation of multiplier-less LMS adaptive filter. 2-bit adder network is used instead of 4-bit adder network in order to reduce the complexity of adder network when filter length is so high. Moreover LUT is replaced by 2-bit adder network and two 2:1 multiplexer’s which lead to reduction of latency. This proposed architecture will be used in applications such as system identification, channel equalization and noise cancellation.

REFERENCES