International Journal of Engineering Research and General Science Volume 4, Issue 3, May-June, 2016
ISSN 2091-2730

ADAPTIVE FIR FILTER USING DLMS FOR AREA EFFICIENT DESIGN

Indhumathi.S
Assistant Professor, Apollo Engineering College, Chennai, India
indhu.137@gmail.com, 9790712827

Abstract— least mean square (LMS) adaptive filter is the most popular and most widely used adaptive filter. It is simple and also its convergence performance is satisfactory. The structure can be modified to support pipelining is called delayed least mean square (DLMS) algorithm. From the structure of the DLMS adaptive filter, it is clear that there is a scope for reducing the area and delay in the existing structure of DLMS filter. This work uses a novel partial product generator for achieving lower adaptation-delay and area. Based on this efficient architecture for the implementation of a delayed least mean square adaptive filter have been developed and compared with the existing structure. These works propose a strategy for optimized balanced pipelining across the time-consuming combinational blocks of the structure. We find that the proposed design offers nearly less area and delay than the best of the existing systolic structures on seeing synthesis results.

Keywords— LMS adaptive filter, DLMS adaptive filter, AOC AND/OR cell, PPG partial product generator, ESM electronic support measure, MAC multiply and accumulate.

I. INTRODUCTION

Least mean square (LMS) adaptive filter is the most desired and most widely used adaptive filter. It is simple and also its convergence performance is satisfactory. In direct-form adaptive filter, pipelined implementation is needed to reduce the long critical path which arises due to inner-product computation. Critical path as to be reduced by pipelined implementation when it exceeds the selected sampling period. Because of its recursive behavior conventional LMS algorithm does not support pipelined implementation .Algorithm as to be modified to a form called the delayed LMS (DLMS) algorithm, which allows pipelined implementation of the filter.

Impulse response of a FIR filter is finite because it becomes zero in finite duration. IIR (infinite impulse response) filter continue to respond indefinitely, it as internal feedback. For Nth-order FIR filter it lasts for N+ 1 sample, and then settles to zero. FIR filters can be discrete-time or continuous-time, and digital or analog.FIR filters are the digital filters whose impulse response reaches zero in finite number of steps .FIR filter is implemented by convolving its impulse response with the time data sequence it is filtering. FIR filters are simpler than IIR filters, which contain one or more feedback terms. Difference equations are used to implement FIR filter. Recursive techniques are also used to implement FIR filter.

One of the simplest implementation of FIR is "linear Phase" design. Here phase is not disturbed, input signal is delayed. They are simple to implement.FIR calculation is done by looping a single instruction on DSP processors. They are suited to multi-rate applications. By multi-rate, we can reduce the sampling rate by "decimation" and we can increase the sampling, or both. By both the process some of the values are excluded thus providing an important computational efficiency.

Whereas in IIR filters are used, each outputs are individually calculated (therefore feedback is combined into the filter). They have desirable numeric properties. DSP filters must be implemented using "finite-precision" arithmetic, that is, a limited number of bits. But certain significant problems occur when feedback is used. But using fewer bits fir filters can be implemented, because they have no feedback. But lesser problems will arise to solve related to non-ideal arithmetic. Fractional arithmetic is also used to implement. FIR filter is also implemented with coefficients less than 1.0. To make implementation simpler this is an important factor to consider.

IMPULSE RESPONSE – It is a set of FIR coefficients. (For example impulse response of a FIR filter which consists of a "1" sample following "0" samples, the output will be the set of coefficients, as a sample moves past each coefficient following to form the output.)

TAP - "Tap" is simply a coefficient or delay pair. The amount of FIR taps, (denominated as "N") is an illustration of 1) the number of memory required to implement the filter, 2) the number of computations required, and 3) the amount of "filtering" the filter can do; in effect, more taps means more narrower filters, less ripple, stop band attenuation etc.)

MULTIPLY-ACCUMULATE (MAC) – MAC operation is done by delaying the data sample, multiply it with the coefficient and accumulate the result. FIRs usually require one MAC per tap. In DSP microprocessors MAC operation is implemented in a single instruction cycle.

TRANSITION BAND – It is the band of frequencies present in between pass band edges and stop band edges. More of taps are needed for narrow band. ("small" transition band -"sharp" filter.)

DELAY LINE – Delay line is the set of memory elements that implement the "Z^(-1)" delay elements of the FIR calculation.

CIRCULAR BUFFER – It is a buffer which wrap to the origin when it is incremented and wraps to the terminus when it is decremented. DSP microprocessors provides circular buffer not by literally moving in a memory but simply replacing it.

II. LITERATURE SURVEY

"A Systolic Array Realization Of An LMS Adaptive Filter And The Effects Of Delayed Adaptation",Hanan Herzberg, Raziel Haimi_cohen.

Description -This paper presents a design of a systolic array of an adaptive filter. Filter with LMS algorithm creates problem in implementation, therefore a modified algorithm, a special case of the delayed LMS (DLMS) algorithm introduces a delay in the restoring of the filter coefficients. The steady state and convergence behavior of the systolic array are analyzed. But in practical systolic array and conventional LMS implementation is similar.

Disadvantages of Existing System

The structure involves high adaptation delay for large order filters and so the convergence performance degrades considerably for high adaptation delay.

"An Efficient Systolic Architecture For The DLMS Adaptive Filter And Its Applications",Lan-Da Van, and Wu-Shiung Feng

Description-This paper presents an efficient systolic architecture for the delay least-mean-square (DLMS) adaptive FIR digital filter. It is based on a processing element (PE) tree- systolic and an optimized tree-level rule. Applying our processing element, a higher convergence rate than that of the conventional DLMS structures can be obtained . No need to sacrifice the properties of the systolic-array architecture It operates at the highest throughput in the word-level and also considers finite driving or update of the feedback error signal.

Furthermore, based on our proposed optimized tree-level rule that takes account of minimum delay and high regularity, an efficient N -tap systolic adaptive FIR digital filter can be easily determined under the constraint of maximum driving of the feedback error signal.

Disadvantages of Existing System

The systolic architecture uses relatively large processing elements for inner product computation .This achieves lower adaptation delay but involves critical path.

"Virtex FPGA Implementation of a Pipelined Adaptive LMS Predictor for Electronic Support Measures Receivers",**Lok-Kee Ting, Roger Woods and Colin. F. N. Cowan.**

Description

In this paper FPGA of an LMS filter along with ESM digital receiver is implemented. Here "fine-grained" pipelining is used which is nothing but pipelining is done within the processor. Finally the result is there is an increase in output latency when we use in LMS system. Main challenge in this paper is there should be an increase in pipelining but output latency as to be low in order to have high speed. In this paper DLMS algorithm is implemented in direct form as well as in transposed form using fine-grained pipelined FPGA. Results are compared. Among both the implementations, direct form LMS filter consumes more FPGA resources effectively with a sampling rate of 120 MHz .A high-speed implementation features and flexibility of a COTS platform is seen in FPGA. Nowadays in FPGA's various classes of adaptive algorithms were implemented on a single FPGA device. Since adaptive filter as error feedback signal in the recursive structure one cannot design adaptive filter using direct implementation. At this point FPGA proves to be slower in performance. most of the DSP algorithm improves its system throughput rate by accomplishing concurrency in the form of parallelism and pipelining. But pipelining of adaptive recursive structures will inevitably destroy the adaptive filtering stability and performance due to the increased latency in the feedback structure. One way to stabilize the filter is to employ a smaller step size, but this hampers the adaptive filtering performance. FPGA implementations of pipelined LMS filters are presented in this paper. "Fine-grained" pipelining application within the filter processors and careful selection of the filter architecture, results in a high speed, low latency design. Comparison of area, latency and speed is been done using DLMS implementation in both direct as well transposed structure. All designs implemented using the Xilinx Virtex FPGA technology.

Disadvantages of Existing System

Critical path is limited to the maximum of one addition time and it also supports high sampling frequency, but main disadvantage higher power consumption and lot of area overhead for pipelining due to its large number of pipeline latches.

"A High-Speed FIR Adaptive Filter Architecture using a Modified Delayed LMS Algorithm",**Pramod K. Meher, Megha Maheshwari.**

Description

This paper presents a modified DLMS algorithm and a new architecture for high-speed adaptive filtering with very low adaptation-delay. The proposed architecture involves two pipelined blocks: (i) one for the computation of error, (ii) and the other for weight-updating. The final subtraction to compute error is merged with the computation of filter output, and realized by a pipelined inner-product unit. Similarly, the multiplications and additions involved in weight updating are merged and performed by N pipelined units in parallel for Nth order filter. The critical-path is restricted to one addition time using carry-save chain in both these units and introducing feed-forward cut-sets at desired locations.It is more efficient in terms of power-delay product and area delay product

compared with the existing structures, since it involves very less number of pipelining latches. Besides, it is easily scalable for higher order filters, since the number of pipeline stages and so also the adaptation-delay do not change significantly with the filter order.

Disadvantages of Existing System

This structure involves several stages of carry save unit to produce final product result and this structure does not support pipelining hence involves delay.

III. ADAPTIVE FIR FILTER

3.1 MODULES

- AOC Block.
- PPG unit.
- Error-computation block.
- Weight-update block.
- LMS adaptive filter.

3.2 MODULE DESCRIPTION

There are two main computing blocks in the adaptive filter architecture:

- The error-computation block
- weight-update block.

In this Section, in the error computation block the adaption delay is minimized which is followed by the weight-update block.

3.2.1 AOC Block

The structure and function of an AOC is depicted in Figure 3.1.Each AOC consists of three AND cells and two OR cells. Each AND cell takes an input D(n-bit) and input b (a single bit), and consists of n AND gates. It distributes all the n bits of input D to its n AND gates as one of the inputs.

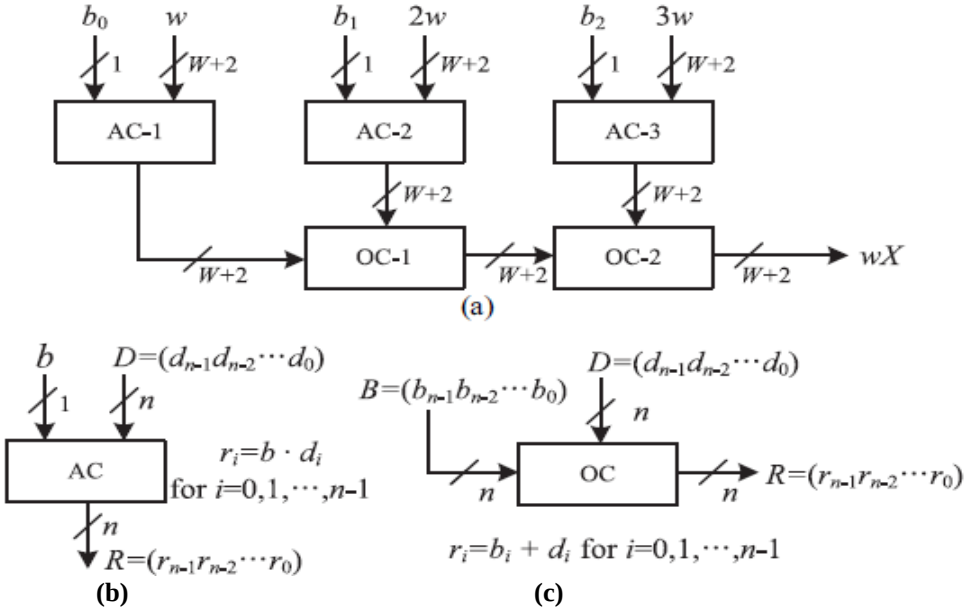


Figure 3.1 Structure and function of AND/OR cell

Other inputs of all the n AND gates are fed with the input b (single-bit), each OR cell similarly takes a pair of input (n-bit) words and has OR (n) gates. A pair of bits in the same bit position in B and D is fed to the same OR gate. w, 2w, and 3w are the output of an AOC corresponding to the decimal values 1, 2, and 3 of the 2-b input (u_1u_0). A multiplication of w (input operand) with a 2-b digit (u_1u_0) is performed by decoder along with the AOC, such that the PPG of Figure.3.2. performs L/2 parallel multiplications of input word w with a 2-b digit to produce L/2 partial products of the product word wu.

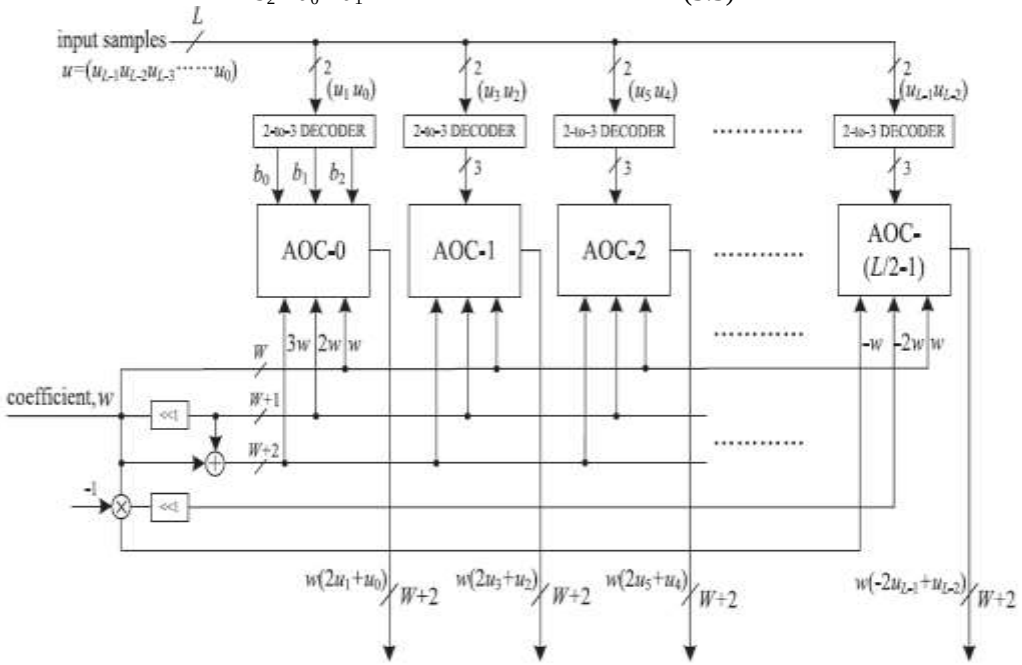
3.2.2 PPG UNIT

The structure of each PPG is shown in Figure3.2.It as L/2 number of 2-to-3 decoders and the same number of AND or(AOC) OR cells. All 2-to-3 decoders takes a 2-b digit (u_1u_0) as input and produces three outputs as given in equations (3.1),(3.2)and (3.3) such that $b_0 = 1$ for $(u_1u_0) = 1$, $b_1 = 1$ for $(u_1u_0) = 2$, and $b_2 = 1$ for $(u_1u_0) = 3$.

$$b_0 = u_0 \cdot \bar{u}_1 \quad (3.1)$$

$$b_1 = \bar{u}_0 \cdot u_1 \quad (3.2)$$

$$b_2 = u_0 \cdot u_1 \quad (3.3)$$



- **Figure 3.2** Proposed structure of PPG

b_0, b_1 and b_2 which is the decoded output fed with w, 2w, and 3w to an AOC. w, 2w, and 3w are sign extended so that each may have (W + 2) bits & they are in two's complement representation . w, −2w, and –w are fed as input to AOC (L/2 − 1) . so that (u_{L-1} u_{L-2}) can have four possible values 0, 1, −2, and −1.This is done to consider the sign of the input samples during the computation of partial product corresponding to (MSD) most significant digit (u_{L-1} u_{L-2}) of the input sample.

3.2.3 PPG and A Normal Multiplier

Normal multiplier performs bit by bit multiplication whereas in case of partial product generator, multiplication can be done by 2 bits parallelly.Hence the computation made simpler and also reduces the delay in generating the partial products. This makes PPG to be more efficient than the multiplier.

Manual Example

Multiplier

```
    U      00000010
    W      00000011
           00000010
         00000010
        00000000
       00000000
      00000000
     00000000
    00000000
   00000000
  000000000000110
```

PPG

Let U=00000010 W=00000011

Decoder output

$u_0 = 0 \quad u_1 = 1$

Substitute the above values in equations (3.1), (3.2) and (3.3) we get,

$b_0 = 0.0 = 0$

$b_1 = 1.1 = 1$

$b_2 = 1.0 = 0$

$W_1 = 00000011$

$W_2 = 000000110$ (<<1)

$W_3 = 0000001001$ ($W_2 + W_1$)

$b_0.W_1 = 0$,

$b_1.W_2 = 0000000110$,

$b_2.W_3 = 0$

AOC output=0000000110. Other three AOC outputs are 0.

In the above example multiplier generates 8 partial products but PPG generates only 4 partial products and hence reduces the delay and computation complexity.

3.2.4 Structure of Adder Tree

In practice we perform shift-add operations on partial products of PCG in order to get product value. And inner product is obtained by adding all the N product values. But word length increases due to shift-add operation to get product value. Due to that adder size of N − 1 additions of the product values also increases. N partial products with the same place value from all the PPGs(N) is been added by one adder tree to overcome such an increase in word size of adders. (L/2) binary adder trees are added with all the L/2 partial products generated by each of the N PPGs.

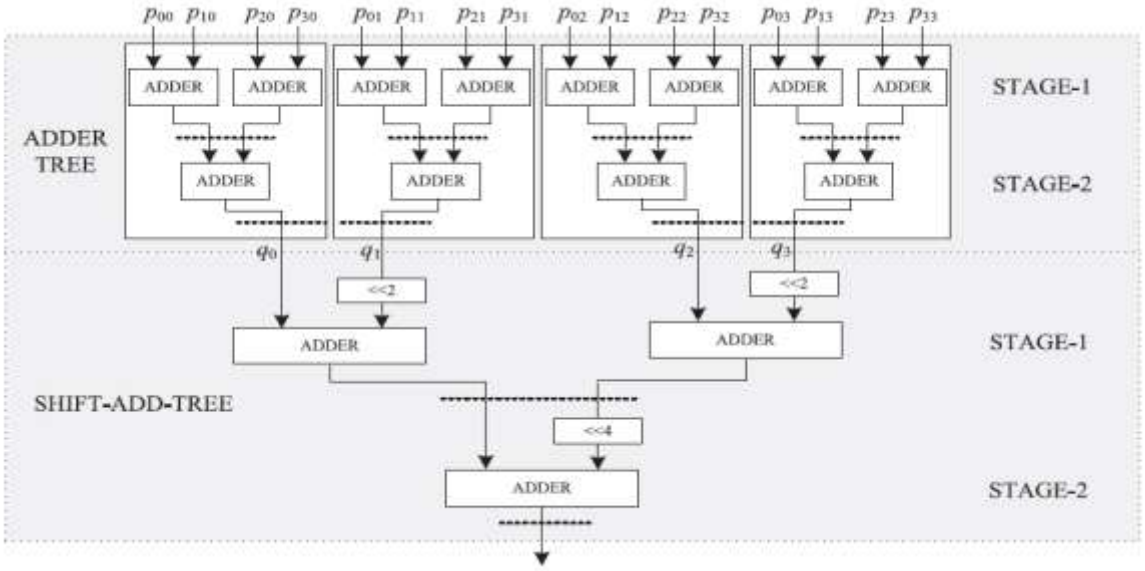


Figure 3.3 Adder structure of the filtering unit for N=4 and L=8

According to their place values the outputs of the L/2 adder trees are then added by a shift-add tree. To add N partial product each of the binary adder trees require log2 N stages of adders, and to add L/2 output of L/2 binary adder trees the shift–add tree requires log2 L − 1 stages of adders. The addition scheme for the error-computation block for a four-tap filter and input word size L = 8. The adder network requires 4 binary adder trees of 2 stages each and a 2-stage shift–add tree for N = 4 and L = 8. In the Figure3.3, we have shown all possible locations of pipeline latches by dashed lines.

3.2.5 Error-Computation Block

The proposed structure for error-computation unit of an N-tap DLMS adaptive filter is shown in Figure3.4

Outputs are generated using equations (3.4) and (3.5) for error and the filter.

$$e_{n-n1} = d_{n-n1} - y_{n-n1} \quad (3.4)$$

$$y_n = w^T_{n-n2} \cdot x_n \quad (3.5)$$

This block consists of

1)2-bit PPG equal to N multipliers

2) A cluster of L/2 binary adder trees

3)A single shift–add tree

Each sub block is described in detail in previous sections.

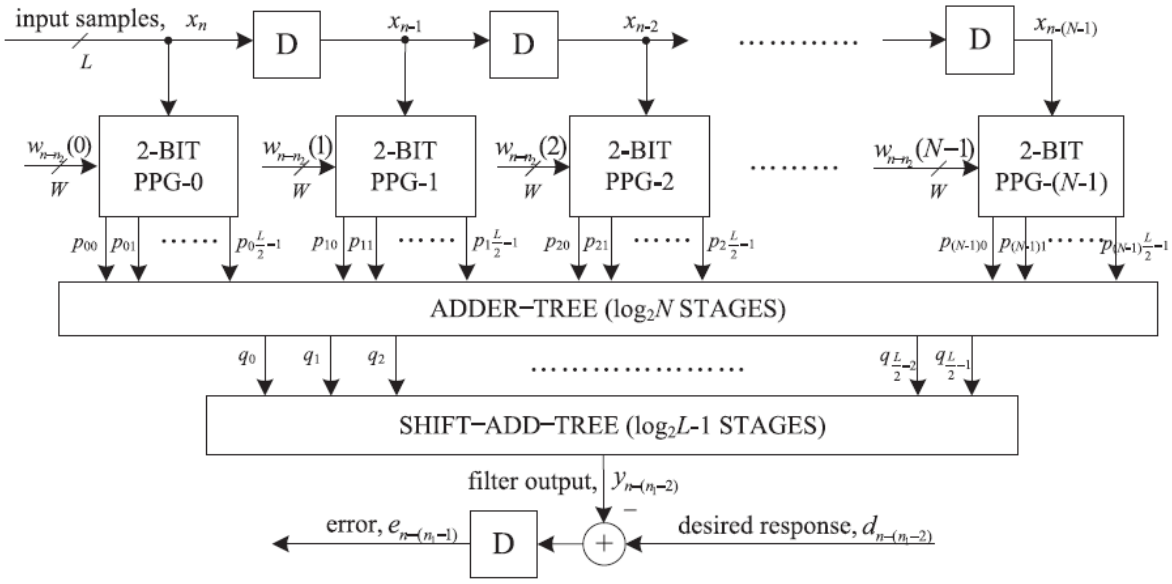


Figure 3.4 Proposed structure of error computation block

3.2.6 Weight-Update Block

Figure 3.5 represents the proposed structure for the weight-update block is shown in. It performs (μ × e) × xi + wi operation which is N multiply-accumulate operations to update N filter weights. μ, the step size is taken as power of negative of 2 to realize the multiplication with recently available error. Shift operation is done to perform this process. So every MAC units performs the multiplication of xi (delayed input samples) with the the shifted value of error. After that addition is performed with the corresponding old weight values wi.

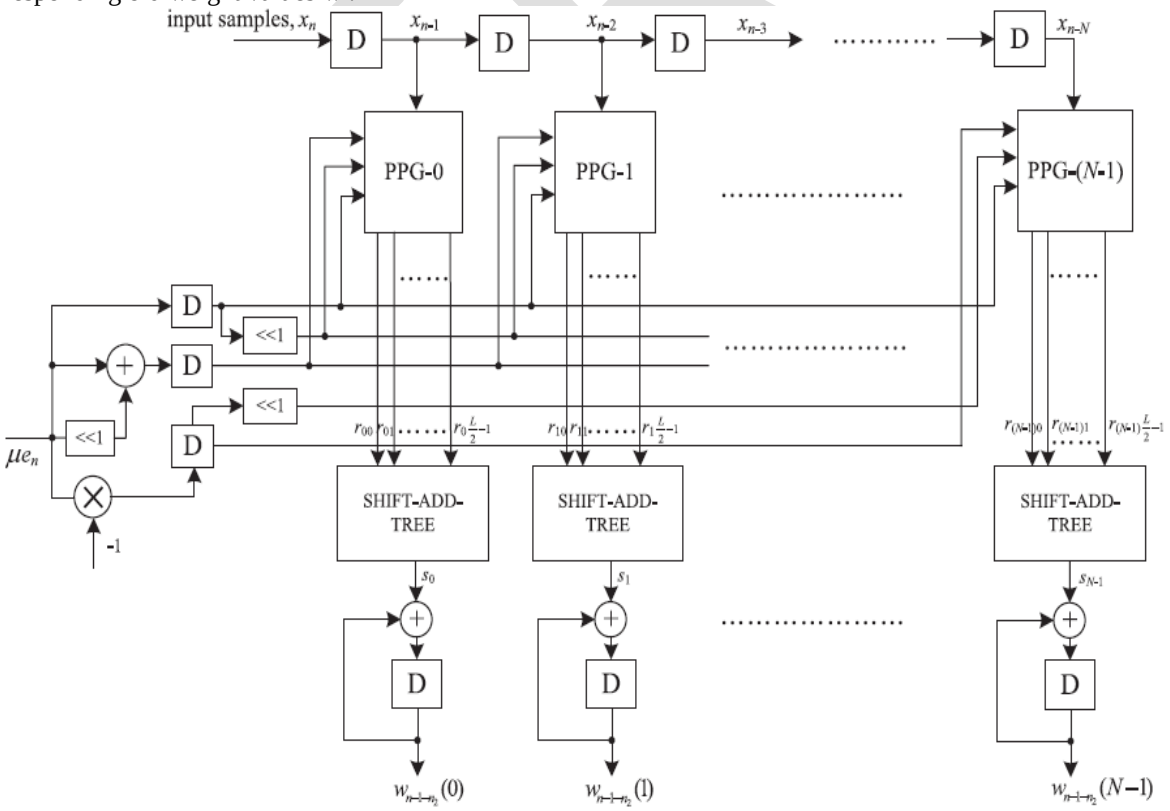


Figure 3.5 Proposed structure of the weight update block

N PPGs performs all the N multiplications for the MAC operations which are followed by N shift– add trees. Every PPGs generates L/2 partial products. These products are equal to the product of the recently shifted error value μ × e with L/2, the number of input word xi (2-bit digits), where the sub expression 3μ×e is shared within the multiplier. The μ×e(scaled error) is multiplied with all N delayed input values in the weight-update block. Therefore this sub expression can be allotted across all the multipliers. Due to this there is a substantial reduction of the adder complexity. For the next iteration error- computation block and weight-update block is loaded with the inputs coming from MAC units. MAC units generate the desired updated weights.

3.2.7 LMS Adaptive Filter

Adaptation delay of conventional LMS is dissolved into two parts 1) delay in pipeline stages during FIR filtering 2) due to the delay involved in pipelining the weight update process. Based on such a decomposition of delay, the LMS adaptive filter can be implemented by a structure shown in Figure3.6**.**

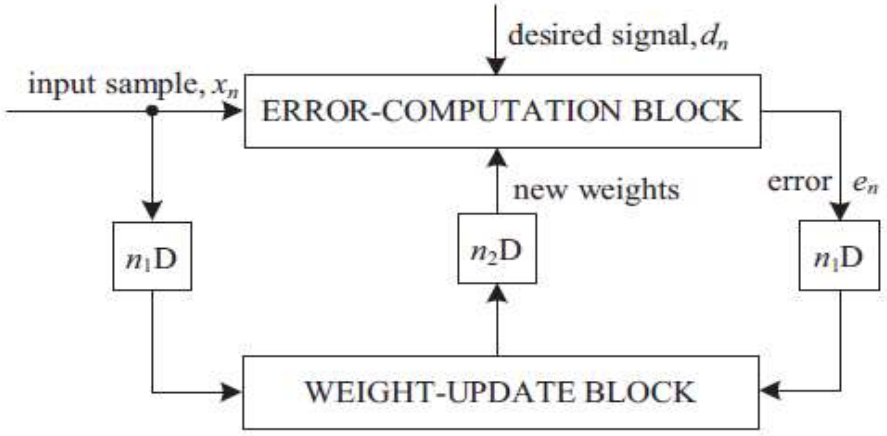


Figure 3.6 Structure of modified delayed LMS adaptive filter

3.2.8 Adder-Tree Optimization

For the computation of yn the adder tree and shift–add tree can be pruned for further optimization of power complexity, delay and area. For the computation of filter output with that the proposed pruning optimization of adder tree and shift–add tree is illustrated , a simple example is taken with the filter length N = 4 ,L=8 &W=8(word length). The dot diagram of the adder tree is shown in figure3.7. Each row of the dot diagram denotes partial products generated by the PPG unit, for W = 8. Each row contains 10 dots. We have taken L value as 8.therefore we will get four sets of partial products corresponding to four partial products of each multiplier. Each set of partial products of the same weight values contains four terms, since N = 4. The final sum without truncation should be 18 b. However, we use only 8 b in the final sum, and the rest 10 b are finally discarded. Computational complexities are reduced by truncating some of the LSBs of inputs of the adder tree. Truncation impact on the error performance of the adaptive filter can be minimized by the use of guard bits. In figure 3.7, four bits are taken as the guard bits and the rest six LSBs are truncated. PPGs do not generate the truncated bits in order to have more hardware saving, in addition to that the complexity of PPGs also gets reduced.

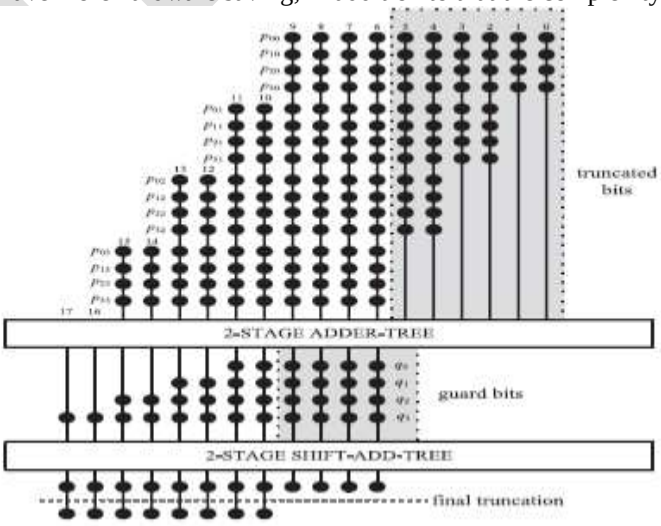


Figure 3.7 Dot diagram for optimization of the adder tree

IV. SOFTWARE REQUIREMENTS

4.2.1 Verification Tool

- ModelSim 6.4c

4.2.2 Synthesis Tool

- Xilinx ISE 13.2

V. RESULTS AND DISCUSSION

5.1 PROPOSED SYSTEM SIMULATION RESULT:

The figure 5.1 shows the simulation result of 2x3 decoder for inputs u0=0 and u1=1

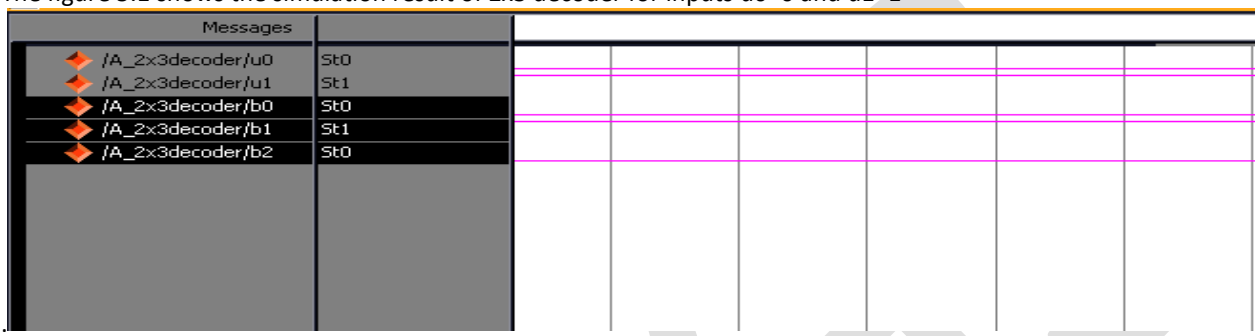


Figure 5.1 2x3 decoder

The input to the decoder is 2 bit and produces 3 bit output. Decoder is used to reduce computation complexity since even both the input bits are set to be 1,only one bit in the output is 1.

The figure 5.2 shows the simulation result of AOC block with inputs b0,b1,b2 , w0,w1and w2.Corres ponding b and w bit get multiplied and at the end all products are added to get wout.

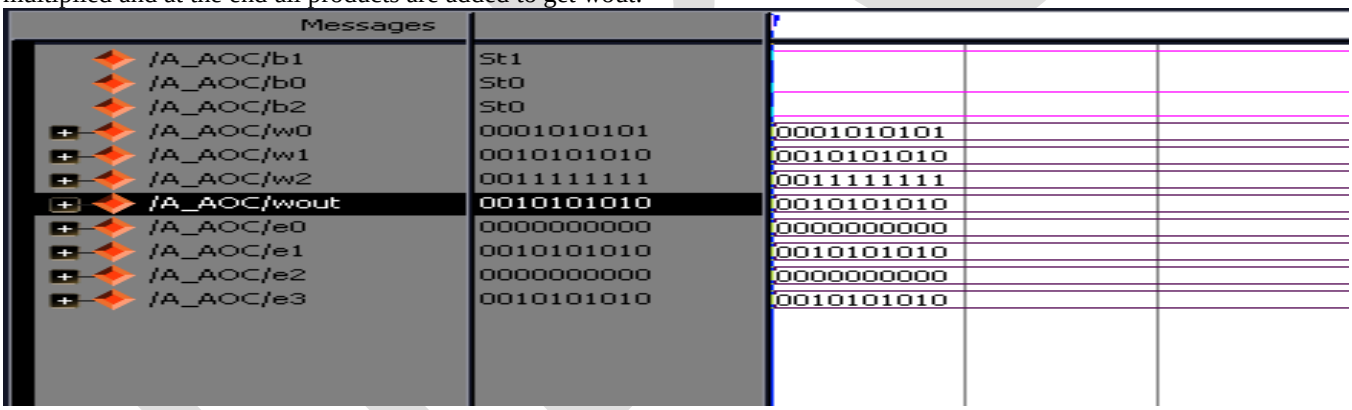


Figure 5.2 AOC block

The figure 5.3 shows the simulation result of PPG unit with inputs u and w and generates the partial products w0,w1,w2 and w3.

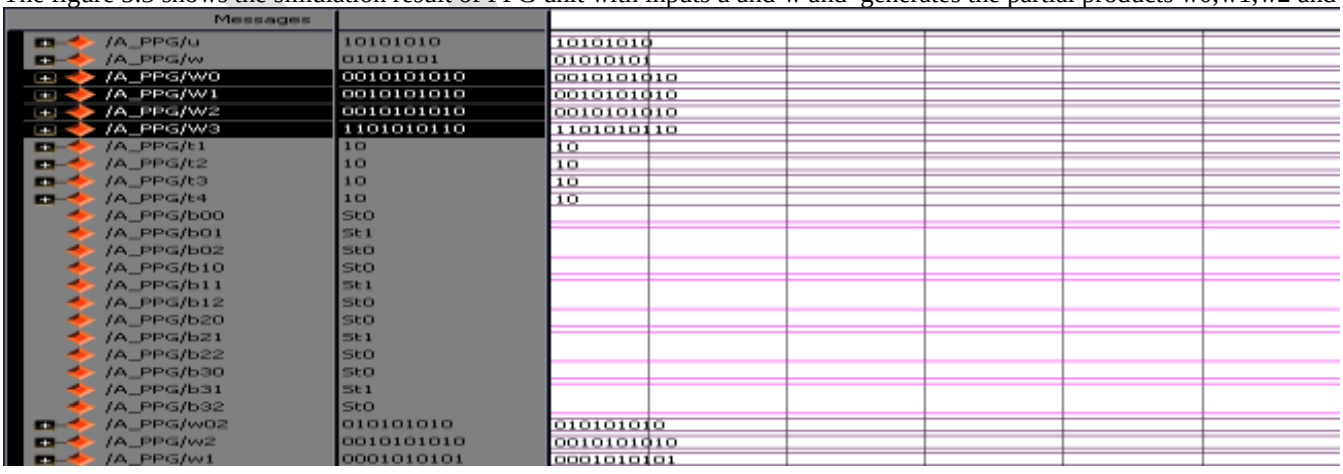


Figure 5.3 PPG unit

The figure 5.4 shows the simulation result of error computation block with inputs u, w, clock and reset and generates the filter output y and error en.

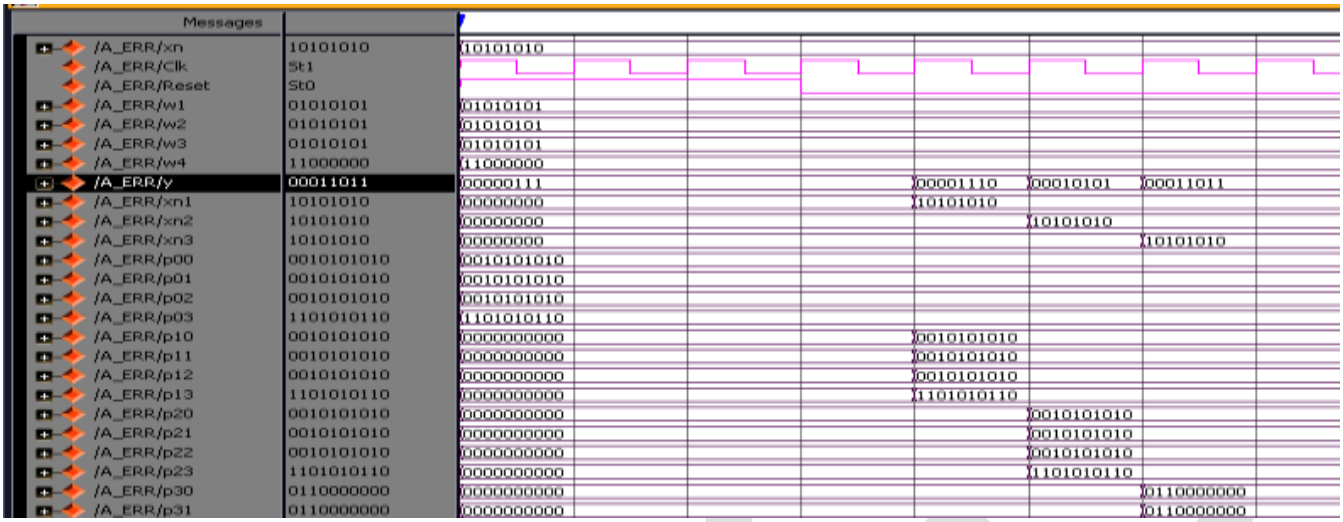


Figure 5.4 Error computation block

The figure 5.5 shows the simulation result of weight update block with inputs u ,en ,clock and reset.

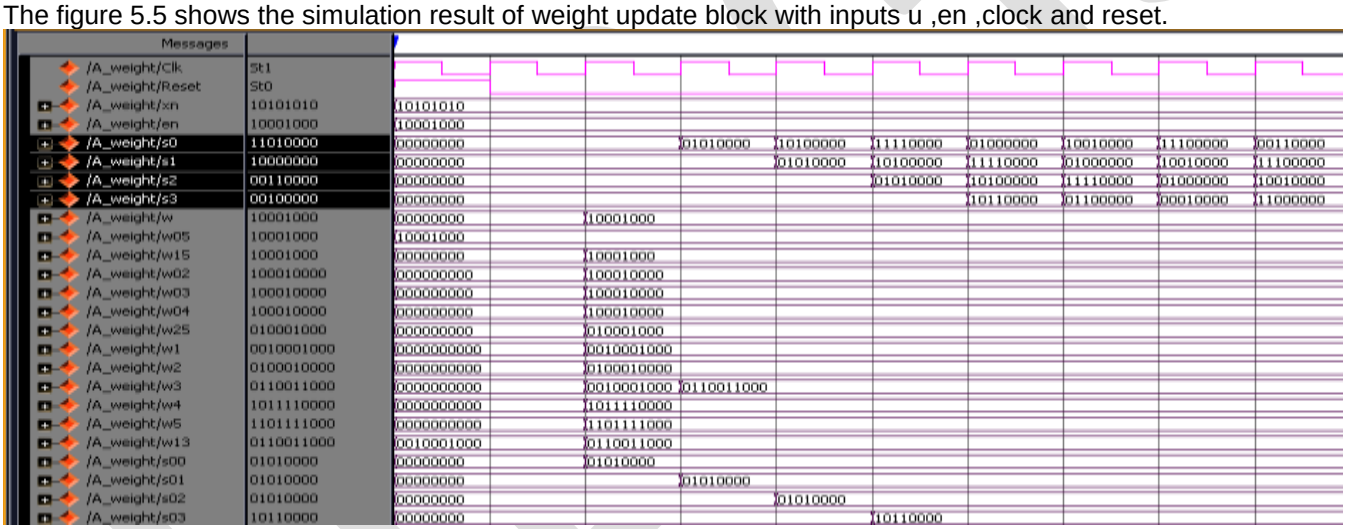


Figure 5.5 Weight update block

The figure 5.6 shows the simulation result of filter with inputs xn ,dn,clock and reset.

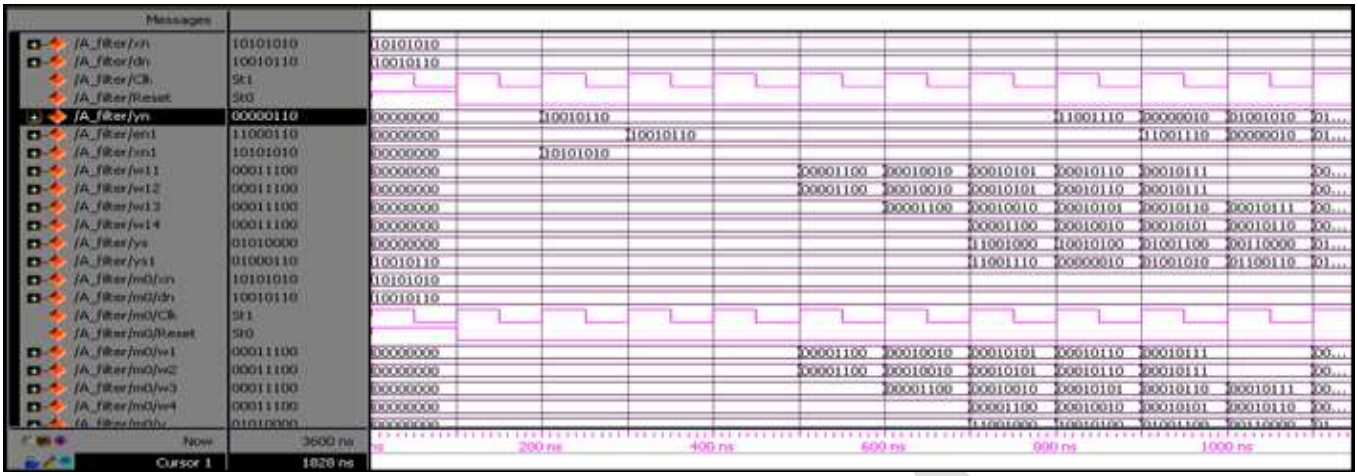


Figure 5.6 Filter

5.2 SYNTHESIS REPORT

5.2.1 AREA CALCULATION

Table 5.1 Comparison of area

FOR L=8 ,N=4	**Total Equivalent Gate Count**
Existing	5295
Proposed	4990
Modified	4970

The table 5.1 compares the number of gates used in existing, proposed and modified structure. The existing system does not include truncation and so the gate required is high. The proposed system uses truncation and reduces the number of gates required to 4990.Further the structure is modified by using adders instead of subtractor and hence the gate count is reduced to 4970.

5.2.2 DELAY CALCULATION

Table 5.2 Comparison of delay

FOR L=8 N=4	**Delay**
Existing	19.949
Proposed	7.241
Modified	7.241

The table 5.2 compares the delay involved in existing, proposed and modified structures. The existing system involves more delay that can be reduced by implementing truncation in proposed system.

5.3 RTL SCHEMATIC

The figure 5.7 shows the rtl schematic structure of adaptive filter. It has four inputs dn,xn with clock and reset and generates the output yn.

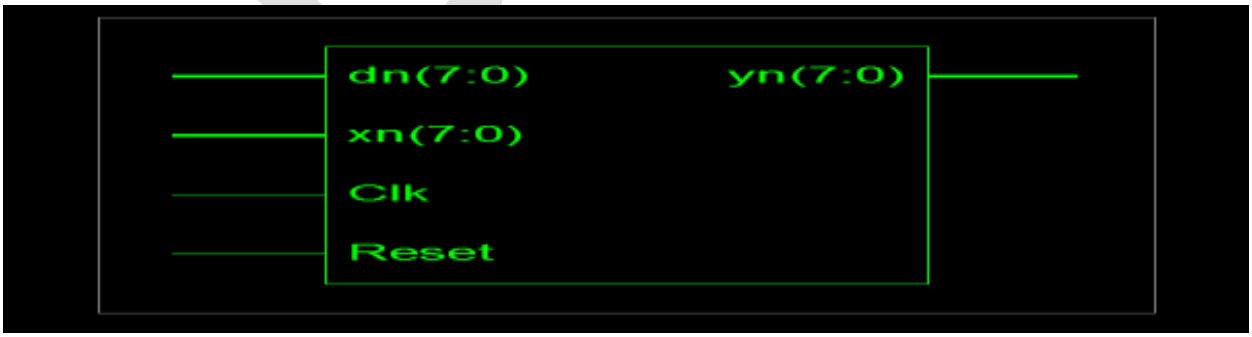


Figure 5.7 Schematic diagram of adaptive filter

The figure 5.8 shows the rtl schematic structure of partial product generator in which the inputs are xn,wn with clock and reset .

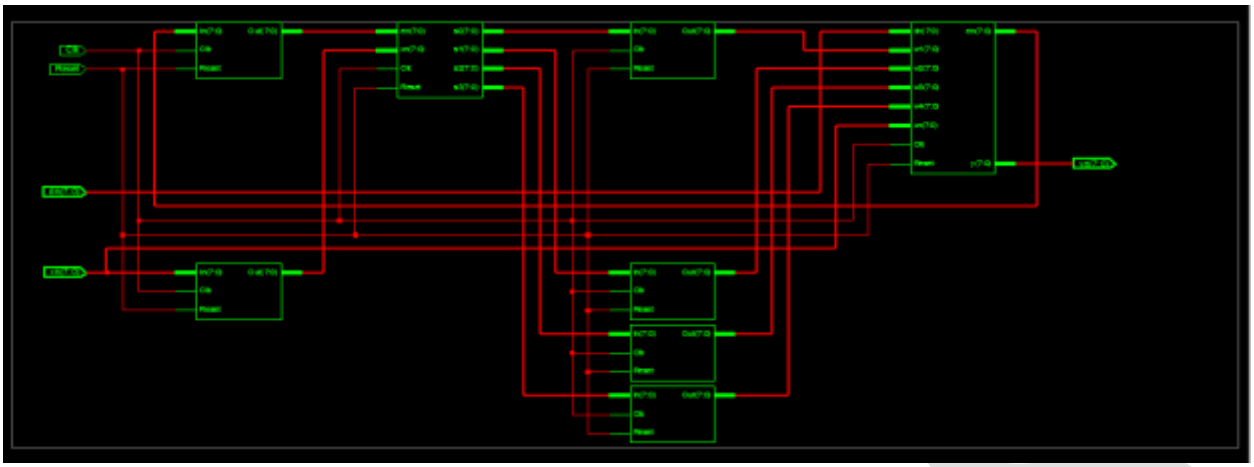


Figure 5.8 Schematic diagram of partial product generator

The figure 5.9 shows the rtl schematic structure of error computation block in which the inputs are xn,wn with clock and reset and generate the outputs yn and en.

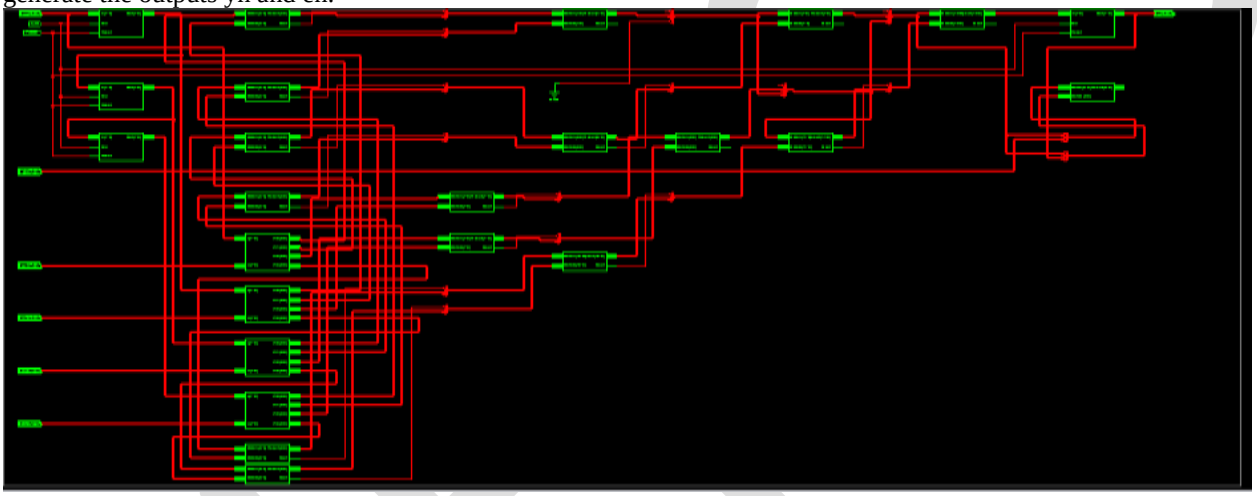


Figure 5.9 Schematic diagram of error computation block

The figure 5.10 shows the rtl schematic structure of weight update block with inputs xn and en and generates the output wn.

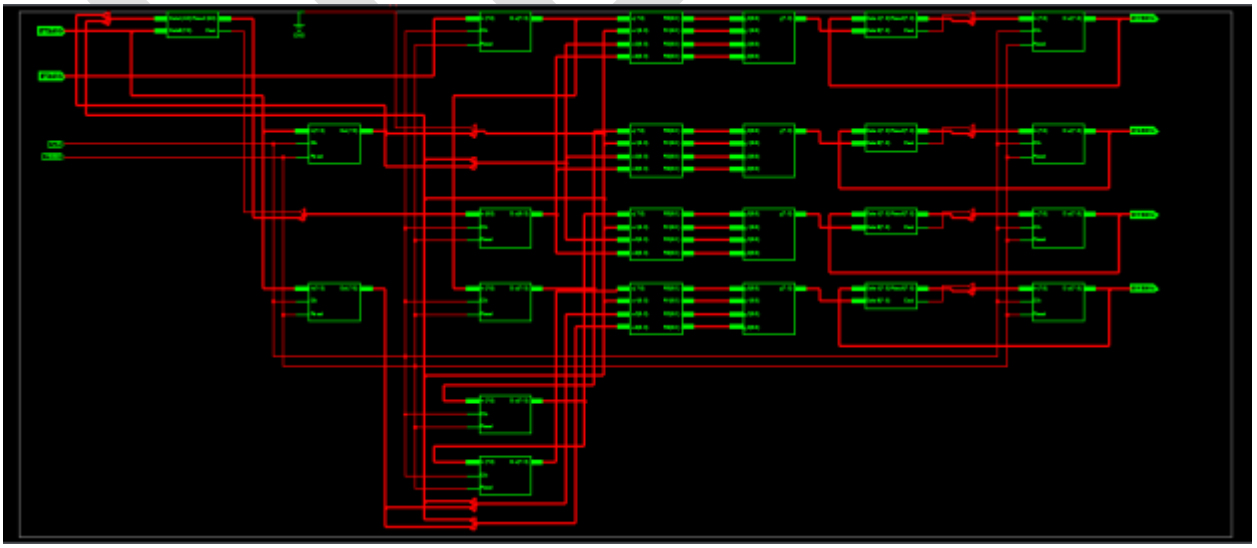


Figure 5.10 Schematic diagram of weight update block

VI.CONCLUSION

An architecture for fixed-point implementation of LMS adaptive filter is been implemented which offers an area–delay-power efficient low adaptation delay. A new PPG which computes general multiplications and inner-product computation by common sub expression sharing is been implemented. An efficient addition scheme is been proposed for inner-product computation in order to reduce the adaptation delay. So that critical path is been reduced which supports high sampling rates & convergence performance is faster. Power consumption & adaption delay as been reduced by optimizing balanced pipeline across the time-consuming blocks of the structure. On comparing with the existing structures area, adaption delay is been reduced in the proposed structure. The delay and area can be reduced to the level of 63.7% and 5.76% when compared to existing structures.

REFERENCES:

[1] H.Herzberg, and R.Haimi-Cohen, "A systolic array realization of an LMS adaptive filter and the effects of delayed adaptative filter and the effects of delayed adaption," IEEE Trans. Signal Process., vol. 40, no. 11, pp. 2799–2803, Nov. 1992.
[2] P.K.Meher, and M.Maheshwari, "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in Proc. IEEE Int. Symp. Circuits Syst , May 2011, pp. 121–124.
[3] P.K.Meher, and S.Y.Park, "Low adaptation-delay LMS adaptive filter part-I: Introducing a novel multiplication cell," in Proc. IEEE Int. Midwest Symp. Circuits Syst , Aug. 2011, pp.1–4.
[4] S.Ramanathan, and V.Visvanathan, "A systolic architecture for LMS adaptive filtering with minimal adaptation delay," in Proc. Int. Conf. Very Large Scale Integr. (VLSI) Design, Jan. 1996, pp. 286–289.
[5] L.K.Ting, R.Woods, and C. F. N.Cowan, "Virtex FPGA implementation of a pipelined adaptive LMS predictor for electronic support measures receivers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 1, pp. 86–99, Jan. 2005.
[6] L. D.Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 48, no. 4, pp. 359–366, Apr. 2001.
[7] Y. Yi, R. Woods, L.-K. Ting, and C. F. N. Cowan, "High speed FPGA-based implementations of delayed-LMS filter," J. Very Large-scale Integr. (VLSI) Signal Process. vol. 39, nos. 1–2, pp. 113–131, Jan. 2005.
[8] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Accuracy evaluation of fixed-point LMS algorithm," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process, May 2004, pp. 237–240.
[9] K. K. Parhi, VLSI Digital Signal Procesing Systems: Design and Implementation. New York, USA: Wiley, 1999.
[10] G. Long, F. Ling, and J. G. Proakis, "Corrections to 'The LMS algorithm with delayed coefficient adaptation'," IEEE Trans. Signal Process.,vol. 40, no. 1, pp. 230–232, Jan. 1992.
[11] M. D. Meyer and D. P. Agrawal, "A high sampling rate delayed LMS filter architecture," IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 40, no. 11, pp. 727–729, Nov. 1993.
[12] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter part-II: An optimized architecture," in Proc. IEEE Int. Midwest Symp.Circuits Syst , Aug. 2011, pp. 1–4.