ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Implementation of Adaptive Filter using Common Sub-expression Elimination Algorithm

[1]Shakthi Priyal N. and [2]Hemalatha K.L.

[1]*PG Student,Department of Electronics & Communication Engineering , Easwari Engineering College, Chennai-89. India.*
[2]*Assistant Professor, Department of Electronics & Communication, Engineering,Easwari Engineering College, Chennai-89. India*

ARTICLE INFO

Article history:
Received 10 March 2015
Received in revised form 20 March 2015
Accepted 25 March 2015
Available online 10 April 2015

Keywords:
Canonic signed digit (CSD),Common sub-expression elimination (CSE),Mixed integer linear programming(MILP) scheduling algorithm,Adaptive FIR filter

ABSTRACT

In this paper, an efficient architecture for the least mean square adaptive filter based on shift and add method is implemented . For achieving lower adaptation-delay and area-delay-power efficient implementation,a novel partial product generator is used and proposes a strategy for optimized balanced pipelining across the time-consuming combinational blocks of the structure. Since ,the complexity of the digital filters usually depends on the number of adders which are used to implement a multiplier,a novel common-subexpression-elimination (CSE) method for the synthesis of adaptive finite-impulse response (FIR) filters is proposed. The proposed CSE algorithm considers both the redundancy among the canonic-signed-digit (CSD) filter coefficients and the length of the critical path in the multiplier block of a transposed-form FIR filter. Therefore, CSE method can perform tradeoff designs between complexity and the throughput rate. Hence, the number of adders is reduced with that by using Mixed integer linear programming scheduling algorithm that optimizes the CSE method so that area and power can be reduced when compared to existing design .

To Cite This Article: Shakthi Priyal N. and Hemalatha K.L., Implementation of Adaptive Filter using Common Sub-expression Elimination Algorithm. ***Aust. J. Basic & Appl. Sci.,*** *9(15): 31-37, 2015*

INTRODUCTION

Adaptive filters are widely used in several digital signal processing applications. The tapped-delay line finite impulse response (FIR) filter whose weights are updated by the famous Widrow Hoff least mean square (LMS) algorithm is the most popularly used adaptive filter not only due to its simplicity but also due to its satisfactory convergence performance. Therefore, when the input signal has a high sampling rate, the critical path of the is to be reduced so that the critical path could not exceed the sampling period.The Least Mean Square (LMS) adaptive filter is the most popular and widely used adaptive filter, because of its simplicity and also for its satisfactory convergence performance.

The advent of consumer applications demanding very high data throughputs like digital television requires high-speed components such as digital filters. Because of the speed, programmable solutions such as digital signal processing.The optimization of these multiplications can lead to important improvements in various design parameters like area or power consumption. A CSE algorithm using binary representation of coefficients for the implementation of higher order FIR filter with a fewer number of adders than CSD-based CSE methods is used. CSE method is more efficient in reducing the number of adders needed to realize the multipliers when the filter coefficients are represented in the binary form.On the other hand, GD algorithms an SPT coefficient is represented by a graph. (It is noticed that the graph representation of a coefficient is not unique.) Each vertex of a graph represents a partial sum. These partial sums can be shared across coefficients if possible. In general, the more partial sums are shared, the more adders are saved. A kind of GD algorithm that we have to mention is the -dimensional reduced adder graph (RAG) algorithm. The RAG- algorithm produces filters with the least number of adders but, it shows that implementation is difficult. Hence, Mixed integer linear programming algorithm is proposed for optimizing the sub-expressions to provide bit level resource minimization. Usually, the GD algorithms outperform the CSE algorithms in terms of the required number of adders after the common sub-expression terms are determined and the ADD/SUB network of non-redundant sub-expressions (or terms) is formed, the product value corresponding to each of the coefficients is computed by an adder-tree that sums up its relevant terms. For that the Common

Corresponding Author: Shakthi Priyal N., Department of Electronics & Communication Engineering, Easwari Engineering College, Chennai-89. India.
E-mail: nshakthipriyal@gmail.com

Sub-expression Elimination (CSE) technique will be implemented to reduce the number of operation for implementing the multipliers.

Hence, mixed integer linear programming scheduling algorithms (MILP) is used for further optimization that try to reduce the number of adders to implement the multipliers and CSE is technique that searches for instances of identical sub-expressions and analyses whether it is worthwhile replacing them with a single variable holding the computed value. a mixed integer linear programming (MILP) based formulation to schedule the adder-tree of an individual coefficient to minimize the hardware resource. As linearity is required in MILP, various techniques to transform modeling friendly non-linear expressions into linear equations and inequalities are indispensable This optimization in adder trees reduces the complexity of the filter structure further reduces the area and power drastically.

Review Of Lms Algorithm:

In existing design, Reduction of power consumption is achieved by using a fast bit clock for carry-save accumulation but a much slower clock for all other operations.The existing technique reduces the LUT size to one fourth. The complexity of the digital filters usually depends on the number of adders which are used to implement a multiplier. The existing technique is used only for memory optimization but in the proposed design implementation reduction in arithmetic resources is necessary to reduce overall architecture complexity.

Table 1: Review of LMS algorithm.

Operation	Computation
Filtering	$y[n] = w^T[n]x[n]$
Compute Error	$e[n] = d[n]-y[n]$
Update Calculation	$\hat{w}[n] = \mu e[n]x[n]$ Coefficient Update $w[n] = w[n-1] + \hat{w}[n]$

The least mean-square (LMS) algorithm is the most widely used FIR adaptive filtering algorithm. The algorithm is simple to implement, has low computational complexity and the adaptation characteristics are adequate for most applications.The LMS algorithm is summarized in Table 1. The term μ is a scaling factor, or step size, that controls the speed of adaptation,where, w[n] and x[n] represent the vectors of size L of the coefficients and input samples respectively.The main drawback of the LMS algorithm is that the adaptation is sensitive to the magnitude of the input.

For every cycle, the LMS algorithm computes a filter output and an error value that is computed by the difference between current filter output and the desired response. Then, the estimated error is used to update the filter weights in every training cycle is shown in Figure 1.

During each cycle, the LMS algorithm computes a filter output and an error value that is computed by, difference between the current filter output and the desired response. Then filter weights are updated in every training cycle using this estimated error value. Thus the weights of LMS adaptive filter during the nth iteration are updated according to the following equations:

$$w(n+1) = w(n) + \mu \cdot e(n) \cdot x(n) \quad (1)$$

where, $e(n) = d(n) - y(n)$ (2)

$$y(n) = wqT(n) \cdot x(n) \quad (3)$$

The input vector x(n) and the weight vector w(n) at the nth training iteration are respectively given by,

$$x(n) = [x(n),x(n-1),...,x(n-N+1)]T \quad (4)$$

$$w(n) = [w0(n),w1(n),...,wN-1(n)]T \quad (5)$$

d(n) is represented as desired response, and y(n) is represented as the filter output of the nth iteration. e(n) is represented as the error computed during the nth iteration, which is used to update the weights, μ denotes the convergence factor, and N represents the filter length. In case of pipelined designs, the feedback error e(n) becomes available after certain number of cycles, which can be called as "adaptation delay." Thus, the delayed error e(n−m) is used for updating the current weight instead of updating the most recent error for a pipelined architecture.(where, m denotes the adaptation delay). The weight-update equation of such LMS adaptive filter is given by,

$$w(n+1) = w(n) + \mu \cdot e(n-m) \cdot x(n-m) \quad (6)$$

Fir Filter Architecture:

In this section, the proposed FIR filter architecture is presented. Fig.2 shows proposed FIR filter architecture based on the transposed direct form. The dotted portion in Fig. 2 represents the Multiplier Block (MB) [coefficient multiplier share the same input]. The MB reduces the complexity of the filter implementations, by exploiting MCM. The redundancy occurs in MCM, that redundant computations are eliminated using greedy CSE. In Fig.2, PE performs the coefficient multiplication operation with the help of a shift and add unit.

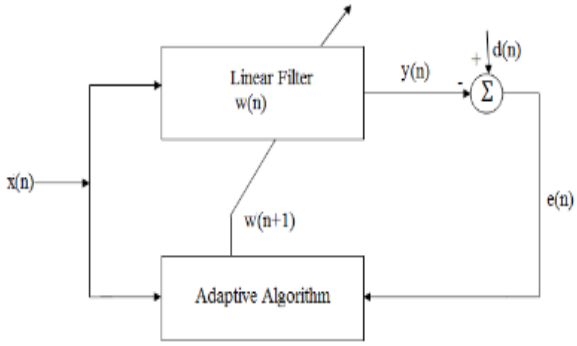

Fig. 1: Block diagram of adaptive filter using LMS algorithm.

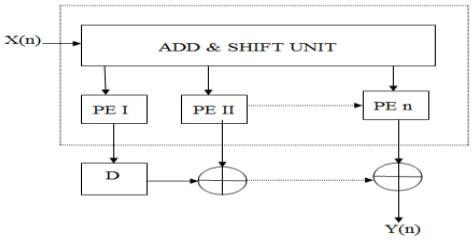

Fig. 2: FIR filter architecture using Add & Shift unit.

The architecture of PE is different for proposed CSM. In the CSM, the filter coefficients are partitioned into fixed groups and hence the PE architecture involves constant shifters.The FIR filter architecture can be realized both in a serial way or parallel way.The same PE is used for generation of all partial products by convolving the coefficients with the input signal (x[n]*h) in a serial way and in a parallel way, parallel PE architectures are employed.

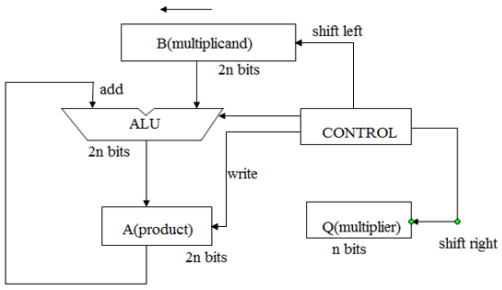

Fig. 3: Architecture of shift & add method.

The 2n-bit product register(A) is initialized to 0. Since the basic algorithm shifts the multiplicand register(B) left one position each step to align the multiplicand with the sum being accumulated in the product register, we use a 2n-bit multiplicand register with the multiplicand placed in the right half of the register and with 0 in the left half is shown in figure 3.

The algorithm starts by loading the multiplicand into the B register, loading the multiplier into the register Q, thereby, initializing the A register to 0. Then, the counter N is initialized to n. The least significant bit(LSB) of the multiplier register (Q0) determines whether the multiplicand is added to the product register. By left shifting the multiplicand, it has the effect of shifting the intermediate products to the left, just as when multiplying normally by paper and pencil.When the multiplier is right shifted it prepares the next bit of the multiplier to examine in the following iteration.

Common Subexpression Elimination Algorithm:

A CSE algorithm using binary representation of coefficients used for the implementation of higher order FIR filter with a fewer number of adders than CSD-based CSE method. CSE method is more efficient in reducing the number of adders needed to realize the multipliers when the filter coefficients are represented in the binary form. The observation is that the number of unpaired bits (bits that do not form Common Subexpressions (CSs)) is considerably fewer for CS coefficients compared to

CSD coefficients, particularly for FIR filters which has higher order. The CSE algorithm deals with elimination of redundant binary common subexpression that occurs within the coefficients. Thus,CSE technique, mainly focuses on eliminating the redundant computations in coefficient multipliers by reusing the most common binary bit patterns (CSs) present in coefficients [5]. The number of CSs that can be formed in an n-bit binary number is 2n − (n + 1).For example, a 3-bit binary representation can form four CSs, which are [0 1 1], [1 0 1], [1 1 0] and [1 1 1]. These CSs can be expressed as

[0 1 1]=x2=2−1x + 2−2x (7)

[1 0 1] = x3 = x + 2−2x (8)

[1 1 0] = x4 = x + 2−1x (9)

[1 1 1] = x5 = x + 2−1x + 2−2x (10)

where x is the input signal and note that other CSs such as [0 0 1], [0 1 0] and [1 0 0] do not require any adder for implementation as they have only one nonzero bit. A straightforward realization of above CSs would require five adders. However x2 can be obtained from x4 by a right shift operation (without using any extra adders).

x2 = 2−1x + 2−2x = 2−1(x + 2−1x) = 2−1.x4 (11)

Also, x5 can be obtained from x4 using an adder:

x5 = x + 2−1x + 2−2x = x4 + 2−2x. (12)

Therefore, it is realized that only three adders are needed to realize the CSs x2 to x5.

Minium Resource Adder-Tree Scheduling Using Milp:

In this section, a mixed integer programming (MIP) based formulation to schedule the adder-tree of an individual coefficient to minimize the hardware resource is described. As linearity is required in MIP, various techniques to transform modeling friendly non-linear expressions into linear equations and inequalities are indispensable and discussed in detail. This MIP procedure is then used in the next section as the building block for resource minimization for the entire FIR filter. Given a set of input terms and their earliest arrival time or delay , we try to form an adder-tree of maximum depth to sum up the input terms and at the same time minimize the hardware resource used.

In order to model the above problem, we construct a complete binary tree of depth . Each node V_i V on the tree is a position to hold a binary operator (ADD/SUB), and . Each edge is a E_i denotes the potential operand position to accommodate an input term, and . Fig. 4 shows an example of the decision tree when . An input term of delay is schedulable on all edge positions of layer and downwards. For each input term , we create a set of binary variables $TSet_i=\{t_{i,j} \mid E_j$ of depths equal/greater than $D_i\}$.T_i is said to be scheduled on if is assigned 1. The adder- tree scheduling problem is equivalent to finding an assignment of input terms to the edge positions on the binary tree such that the resource of the adder-tree is minimized.

Objective Function:

The objective function that minimizes there source cost of all operators on the adder-tree and its structural operator(s) is shown in eqn (13).

minimize : $\sum_{\forall Ej} c_j + \sum_{\forall structural\ ops} c_s$ (13)

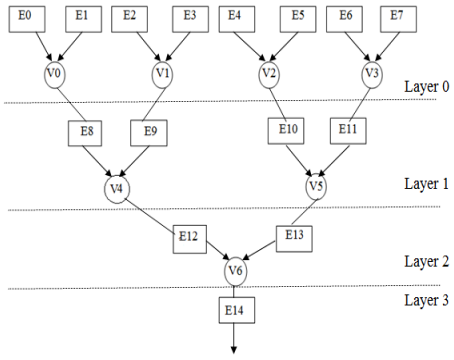

Fig. 4: Binary adder tree of depth L=3 using MIP modeling.

Resource Minimization For The Entire Fir Filter:

In this section, the top level algorithm to optimize the resource consumption of the entire FIR filteris described based on the minimum resource of each individual coefficient returned by solving the MIP formulations discussed previously. The top level algorithm is based on the observation that logic depth relaxation on the coefficient may result in adder-tree of less re- source. Without affecting the filter performance, logic depth relaxation is applicable to all the coefficients whose logic depths are smaller than the logic depth of the filter. Here, first compute the minimum logic depth required by a coefficient adder tree,and then apply logic depth relaxation to improve the minimum resource when applicable.

The algorithm for resource minimization of the entire FIR filter is elaborated in Algorithm . For a particular coefficient, the MIP based resource

minimization procedure is first applied to yield an adder-tree schedule at its minimum logic depth. Further logic depth relaxation is attempted provided that the current depth does not exceed that of the filter's ,and the current attempt did improve the minimum resource. At the end with its adder tree schedule.The total resource of the entire filter is minimized, without increasing its overall logic depth.

Resource Minimization Algorithm of FIR Filter:

1 compute the logic depth of the filter L_{filter} ;
2 **for** each non-zero coefficient C_i **do**
3 compute C_i 's minimum logic depth L_{Ci} ;
4 **repeat**
5 rsrc : = MipSched(C_i , L_{Ci}) ;
6 **if** rsrc is not improved over theprevious iteration **then**
7 use the previous adder-tree schedule and break ;
8 L_{Ci} := L_{Ci} + 1 ;
until L_{Ci} > L_{filter} ;

Algorithm 1. Adder tree scheduling for resource minimization

Minimum Logic Depth of a Coefficient Adder-Tree:

Firstly, the total number of ADD/SUB operators needed to sum up input terms is . Based on the binary tree model of the MIP formulation, a input term with latency/logic depth of nullifies atleast predecessor operator positions on the binary tree. As a result, the minimum binary tree required to accommodating the input terms should consist of atleast N-1 operator positions. On the other hand, a binary tree of depths L contains 2^{L} - 1 operators. .The logic depth of the entire FIR filter takes the largest one among all its coefficients' logic depths.

Proposed Adaptive Fir Filter Architecture:

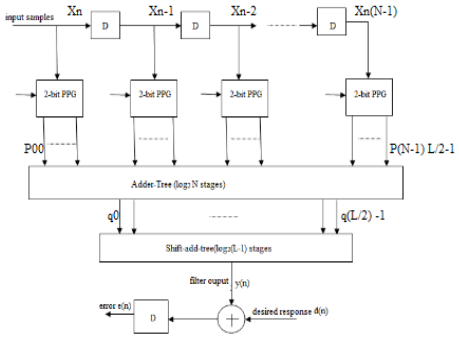

Fig. 5: Proposed structure of error computation block.

As shown in Fig. 5 & Fig. 6, there are two main computing blocks in the adaptive filter architecture: 1) the error-computation block, and 2) weight-update block. In this Section, the design strategy of the proposed structure to minimize the adaptation delay in the error-computation block, followed by the weight-update block is discussed. The proposed structure for error-computation unit of an N-tap LMS adaptive filter is shown in Fig. 5. It consists of N number of 2-b partial product generators (PPG) corresponding to N multipliers and a cluster of L/2 binary adder trees, followed by a single shift–add tree. Conventionally, we should have performed the shift-add operation on the partial products of each PPG separately to obtain the product value and then added all the N product values to compute the desired inner product.

However, the shift-add operation is performed to obtain the product value that increases the word length, and also simultaneously, increases the adder size of N −1 additions of the product values. Increase in word size of the adders is avoided by adding all the N partial products of the same place value from all the N PPGs by one adder tree. On the other hand, some of those pipeline latches are redundant in the sense that they are not required to maintain a critical path of one addition time. The final adder in the shift–add tree contributes to the maximum delay to the critical path. Based on that observation, we have identified the pipeline latches that do not contribute significantly to the critical path and could exclude those without any noticeable increase of the critical path.

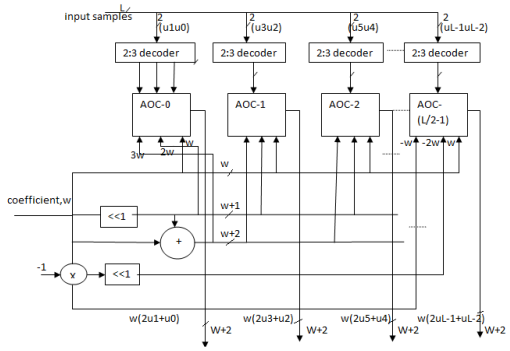

Fig. 6: Structure of weight update block.

The structure for the weight-update block is shown in Fig.6. It performs N multiply-accumulate operations of the form (μ × e) × xi + wi to update N filter weights. The step size μ is taken as a negative power of 2 to realize the multiplication with recently available error only by a shift operation.Since the scaled error (μ × e) is multiplied with all the N delayed input values in the weight-update block, this subexpression can be shared across all the multipliers as well. Hence,this leads to the substantial reduction of the complexity in adders

RESULTS AND DISCUSSION

The coefficients are quantized as 16-bit signed and then converted to their Canonical Signed Digit (CSD) representations. Thus resource minimization problem is identified in the scheduling of adder-tree operations for the shift and add block and a common sub-expression identification algorithm is used which iteratively identifies and eliminates the most frequently occurring coefficients,which is developed to produce the input term network for the adder-trees is known as integers and MIP scheduling algorithm is proposed to optimize the CSE elimination for exact resource minimization are separately coded and noises in the signals are removed that produce the output signal without noise and these are coded in verilog and simulated with Modelsim-Altera 6.4a tool that is shown in figure 7. Table 2. shows that area used by the logic cells are 409/5,136 and 79.13mW power is achieved for the optimized adder/subtractor network of the shift and add block that also reduces the complexity of implementing FIR filters than existing design.

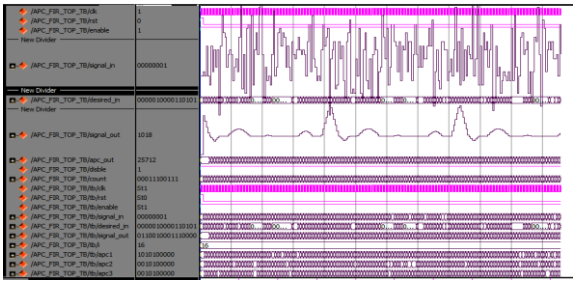

Fig. 7: Simulation output.

Table 2: Comparison table for area and power.

Parameters	AREA	POWER
APC & OMS technique(existing design)	736	107.95mW
CSE algorithm with MILP(Proposed design)	409	79.13mW

Conclusion:

In this paper, the resource minimization problem is identified using common sub-expression elimination algorithm in the scheduling of adder-tree operations for the adaptive filter architecture based on shift and add method, and presented an MIP-based algorithm for optimizing the sub-expression algorithm for exact bit-level resource optimization. Experimental result shows that up to 15% reduction of area and 11.6% reduction of power can be achieved when compared to already optimized ADD/SUB networks of MCM blocks. Further exploration of efficient heuristic algorithms for resource minimization of adder-trees of FIR filters could be done in the future.

ACKNOWLEDGMENT

We would like to extend sincere appreciation to Easwari Engineering College and to all members of Research Group at ECE Department, Anna University, for all the supports and encouragement.

REFERENCES

Allred.D.J, Yoo.H, Krishnan.V, Huang.W, and Anderson.D.V.,2005. LMS adaptive filters using distributed arithmetic for high throughput. IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 7, pp. 1327–1337.

Dong-hwan Lee and Wonyong Sung.,2000.Parallel computation of adaptive lattice filters School of Electrical Engineering, Seoul National University Gwanak-gu, Seoul, 151-742 Republic of Korea .

Haykin.S and Widrow.B.,2003. Least-Mean-Square Adaptive Filters. Hoboken, NJ, USA: Wiley.

Meyer.M.D and Agrawal.D.P.,1993.A high sampling rate delayed LMS filter architecture.IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 40, no. 11, pp. 727–729.

Mohanty.B.K and Meher.P.K.,2008.Delayed block LMS algorithm and concurrent architecture for high-speed implementation of adaptive FIR filters.presented at the IEEE Region10 TENCON 2008 Conf., Hyderabad, India.

Ramanathan.S and Visvanathan.V,1996. A systolic architecture for LMS adaptive filtering with minimal adaptation delay, in Proc. Int. Conf. Very Large Scale Integr. (VLSI) Design, pp. 286–289.

Sang Yoon Park and Pramod Kumar Meher.,2013. Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic IEEE transactions on circuits and systems—ii: express briefs, vol. 60, no. 6.

Widrow.B and Stearns.S.D,1985. Adaptive Signal Processing. Englewood Cliffs, NJ, USA: Prentice-Hall.

Woods.R, Ting.L.K and Cowan C.F.N,2005. 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.

Yi.Y, Woods.R, TingL.K, and CowanC.F.N.,1985. High speed FPGA-based implementations of delayed-LMS filters. Very Large Scale Integr.(VLSI) Signal Process., vol. 39, nos. 1–2, pp. 113–131.