



1-D Systolic Arrays Design of LMS Adaptive (FIR) Digital Filtering

Riyadh A.H. AL-Helali* Bakir A. R. Al-Hashemy** Ali H. Mahdi***

* Department of Electrical Engineering/ Engineering College/ AL-Mustansiriyah University

** Department of Electrical Engineering/ Engineering College/ University of Baghdad

*** Department of Information Engineering/ Al-Khwarizmi Engineering College/ University of Baghdad

(Received 4 March 2008; accepted 9 September 2009)

Abstract

This paper extends the 1-D systolic array approach with a method of systematic linear design of systolic algorithms. Past methods for mapping the Least-Mean-Square (LMS) Adaptive Finite-Impulse-Response (FIR) filter onto parallel and pipelined architectures either introduce delays in the coefficients updates or have excessive hardware requirements. In this article, we describe an efficient 1-D systolic array for the LMS adaptive FIR filter that produces the same output and error signals as produced by the standard LMS adaptive filter architecture with single assignment form of processor functions.

The proposed systolic architectures that are designed operate on a block-by-block basis and makes use of the flexibility in the design, which takes the inner product step (convolution sum) of the tap weight vector and the tap input vector in the design consideration. It enables us to extract more than one algorithm for the same problem. The input and output data flow sequentially and continuously into and out of the systolic arrays at the system clock rates, during each clock period, processing element of the same type operates in parallel. The most computationally demanding among them performs only two consecutive multiplications and two additions/subtractions per clock period, thereby allowing a very high throughput and very fast block signal processing to be achieved at the expense of a delay of L samples between the input and output and 100% utilization, L being the block size.

Keywords: Adaptive signal processing; LMS algorithm; VLSI signal processing.

1. Introduction

The need for higher sampling rates in the signal processing area together with the advent of VLSI technology stimulates research for new computing architectures with faster throughput than that of traditional von Neumann machine. The conventional ways of increasing the processing speed of these computers by increasing their components have apparent limits determined by the underlying physical processes.

The goal of parallel processing, and in particular the goal of parallel computing, is the acceleration of the computations: what is done by a single processing unit in time T that can possibly be done in time T/n processing units, where (n) denotes number of processors [1].

Systolic parallel processing is the class of parallel computers, which make use of local interconnection patterns and distributed memory.

Advancements in VLSI technology have spurred efforts to map complex algorithms onto regular architectures with computations that can be parallelized and/or pipelined[3].

Without exceptions, previously proposed methods to paralyze and/or pipeline the LMS adaptive FIR filter introduce either delays in the coefficients updates [4], [5]-[6] or a large hardware overhead [7] and is not represented as a systolic arrays [8]. In [5]-[6], each coefficient of the system receives an equal delay in the coefficient update, resulting in the delayed LMS adaptive filter. The delayed LMS algorithm is generalized in [4] to allow different coefficient delays for both the filter output computation and the calculation of the coefficient updates.

The pipelined LMS adaptive FIR architecture [8], computed the LMS adaptive filter output exactly and delayed by L samples, broadcasted the

input signal, and the overall complexity (high) is linear in the number of filter coefficients. The employment of LMS adaptive filter with delayed updates has difficulty in choosing the step size to obtain fast and accurate adaptation behavior. The best step size choice for these algorithms is a complicated function of the input statistics and the delays within the adaptation loop [9], [10]. Step-size normalization for the delayed LMS adaptive filter has been proposed [11]; however, the performance of the resulting system still depends on the input signal correlation statistics and the adaptation delays.

Clearly, it would be desirable to obtain a systolic array whose coefficient updates contain no adaptation delays so that the normalized LMS adaptation or other variable step-size strategies can be reliably employed.

The problem under investigation of the various systolic arrays of LMS-based adaptive (FIR) transversal digital filter involves the systolic design of the inner product output (convolution sum) of the tap-input vector with the tap-weight vector. The inner product (convolution sum) computation is a matrix-by-vector multiplication, which can be carried out systolically by inputting the samples into the processing elements (PEs) or cells of the systolic arrays [8]. Systolic algorithms exhibiting different properties can result from the systematic design of the flitters array.

One methodology of systematic design of systolic arrays, as opposed to ad hoc design, is based on the representation of algorithms by means of dependence graphs. One class of such graphs corresponds to the systolic algorithms, in that the projection of a graph from this class delivers a systolic parallel algorithm, and in particular such constituents of the algorithm as the respective systolic array and the required input/output operations. The relation between different systolic algorithms for one and the same problem is evident: applying certain transformation on a graph of this class results in other graphs of the same class which, when projected, gives rise to other systolic algorithms which exhibit different properties [1].

2. Systematic Linear Design of Systolic LMS-based Adaptive Digital Filters

Many researchers have considered the problem of efficiently mapping a given problem on parallel (systolic) architecture.

As a first step toward the implementation of an algorithm onto a processor array, it is common practice to go through a number of refinements (regularization, single assignment form, etc.) that makes the algorithm more suitable to a simple and modular VLSI design. The usual assumption is that the algorithm can be expressed by a system of recurrence equation [12].

In this work, the modeling tools and transformation techniques of the systematic design methodology introduced in [13] and [14] are used. This methodology, although not the only one available, has proved to be the most powerful in that most of the existing systolic array algorithms are covered by it and a great number of new algorithms can be synthesized from the known ones.

The output of the adaptive transversal filter is related to the input and impulse response of the filter by the convolution sum [14]:

$$y(n) = \sum_{k=1}^M w_k^* u(n-k+1) \quad \dots(1)$$

$$= \hat{w}^H(n)u(n)$$

Where the superscript H signifies Hermitian transposition, and the inner product of the tap-weight vector $\hat{w}(n)$, and the tap-input vector $u(n)$ represents the output of the filter. Let the number of the tap-input weight vector be three coefficients ($M = 3$), thus, the assignments of the filter output can be written as [15]:

$$y(n) := \hat{w}_1 \cdot u(n) + \hat{w}_2 \cdot u(n-1) + \hat{w}_3 u(n-2) \quad \dots(2)$$

Various methods of implementing discrete convolution systolically are now presented.

Method 1: A dependence graph for the convolution sum in (1) is shown in Figure (1). This is a homogeneous dependence graph in Z^n with constant dependence vectors where Z denotes the set of integer numbers. It is clear from the figure that $n = 2$. The functions performed at each vertex are specified on an arbitrary vertex shown in Figure (1-a). The following dependence vectors represent the arcs of the graph and are shown in Figure (2).

$$d^{(\hat{w})} = (1, 0)^T, \quad d^{(u)} = (0, 1)^T, \quad d^{(y)} = (1, -1)^T \quad \dots(3)$$

where u denotes the vertically-entered u -data, \hat{w} denotes the horizontally- entered w data, and y is the result of the summation in (1).

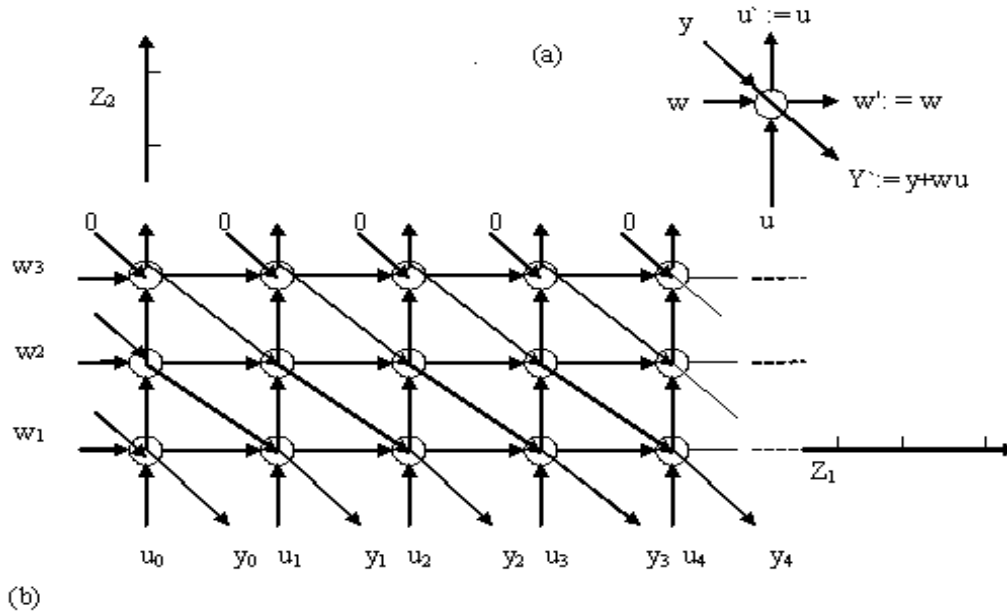


Fig.1. Systolic Dependence Graph for the Convolution Sum.

Now to give a temporal and spatial interpretation for the two coordinates of a vertex in Z^2 , the computational activities attached to a vertex $z = (z_1, z_2)$ have to be executed at time

$$t = z_1 \quad (z_1 \in Z) \quad \dots(4)$$

in a processor located in a point

$$z^* = z_2 \quad (z^* \in Z) \quad \dots(5)$$

This means that, if the original dependence graph space is Z^2 , the processor space is Z . Also, if two vertices of the dependence graph are onto one and the same processor, the respective time periods are different.

A projection vector e , $e \in Z^2$, can be used to obtain the time and location of execution of vertex z as follows

$$t = e^T \cdot z \quad \dots(6)$$

$$z^* = z - (e^T \cdot z / e^T \cdot e) e \quad \dots(7)$$

The above equations specify a projection of the dependence graph. The mapping given by equations (4 and 5) is evidently a special case of

equations (6 and 7) for the following projection vector

$$e = (1, 0)^T \quad \dots(8)$$

In this framework, the z_1 -axis and the z_2 -axis in Figure (1-a) should represent the time instances and the processor locations respectively. It may be deduced from this figure that the left bottommost cell will yield the convolution data points consecutively, one at a time, in the forward order in the fifth time instant, the five convolution data will appear at the outputs of the five processors with $y(0)$ at the left bottommost cell output and $y(3)$ at the right bottommost cell output.

Before proceeding with the mapping, we have to make sure that the causality condition is satisfied. That is,

$$e^T \cdot d \geq 0 \quad \dots(9)$$

Using (8), equation (9) can be reduced to; $d_1 \geq 0$.

The above condition is satisfied for the dependence vectors of the graph shown in Figure (2):

$$d_1^{(u)} = 0, \quad d_1^{(w)} = 1, \quad d_1^{(y)} = 1 \quad \dots(10)$$

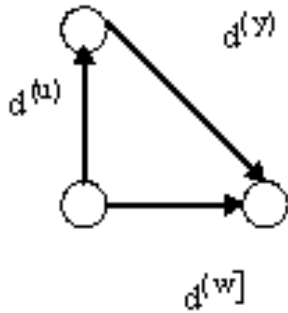


Fig.2. Dependence Vectors for the Graph of Fig. 1.

Satisfying the causality condition ensures that the computational activities of a vertex z' via an arc (z, z') described by a dependence vector d ($d = z' - z$) cannot be executed before the computations are assigned to z . The number of delay elements δ to be arranged in a connection between processors located at points z^* and $z^{*'}$ is determined as follows

$$\delta(z^*, z^{*'}) = e^T \cdot (z' - z) \quad \dots(11)$$

Applying now (7) and (11) on the dependence graph of Figure (1-a) results in a regular array structure. This structure is the systolic array shown in Figure 3. The small shaded boxes indicate delay elements. The last condition provides that at least one delay element is arranged in each connection and the resulting structure is, therefore, completely pipelined. The arcs of the dependence graph which are described by the dependence vectors $d^{(\hat{w})} = (1,0)^T$, $d^{(u)} = (0,1)^T$ and $d^{(y)} = (1,-1)^T$ are mapped onto interconnections between the respective processors via

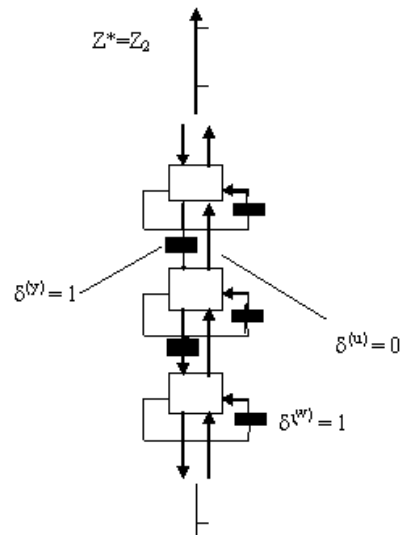
$$\delta^{(\hat{w})} = d_1^{(\hat{w})} = 1, \delta^{(u)} = d_1^{(u)} = 0, \delta^{(y)} = d_1^{(y)} = 1 \quad \dots(12)$$

The arcs described by the dependence vector $d^{(\hat{w})}$ are mapped onto loop connections: an output of a cell is connected to an input of the same cell. Since no delay elements are arranged in the u -interconnections ($\delta^{(u)} = 0$), this array is classified more precisely as semi-systolic.

The procedure executed by each cell/processor of the array is simply a repetition of the computational activities attached to each node of the dependence graph, see figure (3-b). The array features rippling or broadcasting of the u -data, staying of the \hat{w} - data, and pipelining of the y -data.

From the summary of the LMS adaptive filter equations, we can realize a semi-systolic array for the LMS algorithm depending on the systolic array of the adaptive filter output $y(n)$. Figure 4 shows a systolic array realization of LMS algorithm with three coefficients [15].

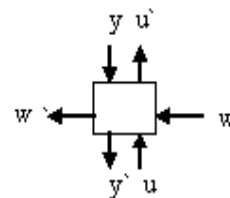
The addition of the products, which belong to one output signal sample $y(n)$, is pipelined, the input signal sample $u(n)$ which is input into the bottommost cell of the array is propagated to all other cells in the same clock period. The error signal sample $e(n)$ is propagated to all cells without delay elements, to compute the term $\mu u(n)e^*(n)$, which represents the correction that is applied to the current estimate of the tap-weight vector, $\hat{w}(n)$. The step-size parameter μ is preloaded in each cell, which represents a suitable constant value. The procedure executed by each cell/processor of the array is shown in Figure (5) [15].



(a)

```

Procedure IPS
begin
  \hat{w}' := \hat{w}; u' := u;
  y' := y + \hat{w}u
end
    
```



(b)

Fig.3. Systolic array of the inner product $\hat{w}^H(n)u(n)$.

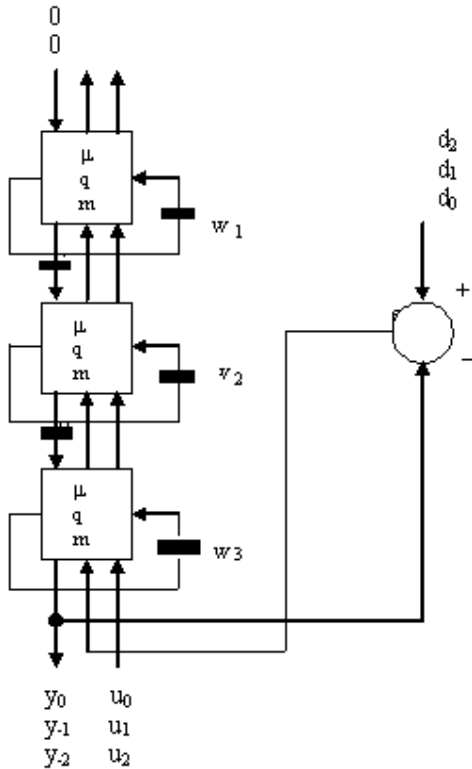
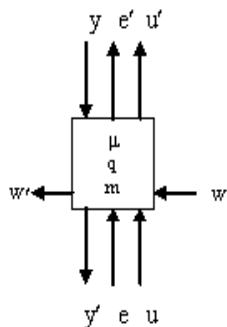


Fig.4. A Semi-Systolic Array Realization of LMS Algorithm (for M=3).



```

Procedure IPS & updates of w
begin
w' := w ; u' := u ; e' := e ;
q := μu;
m := qe;
w := w' + m;
y' := y + w u
end
    
```

Fig.5. A Procedure for the Inner Product Step and Updates the Current Estimate of the Tap-Weight Vector of the LMS Algorithm.

3. Systolic LMS Design Based on Convolution Sum

Method 2: The output of the adaptive transversal filter ($\hat{w}^H(n)u(n)$), using the LMS algorithm, enables us to take different features of the first design and then we can obtain many systolic arrays; each one has its own systolic properties that may be preferred to the other [15].

Design with Rippling of the Intermediate Results

The computations which correspond to the following assignments:

$$y(n) = \sum_{k=1}^M \hat{w}_k^* u(n-k+1)$$

can be represented graphically as shown in Figure (6). The small black boxes stand for delay elements and the triangles represent multipliers.

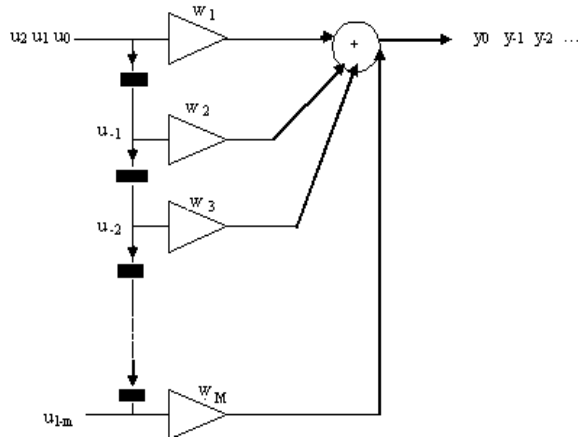


Fig.6. Canonical Realization of a FIR Filter

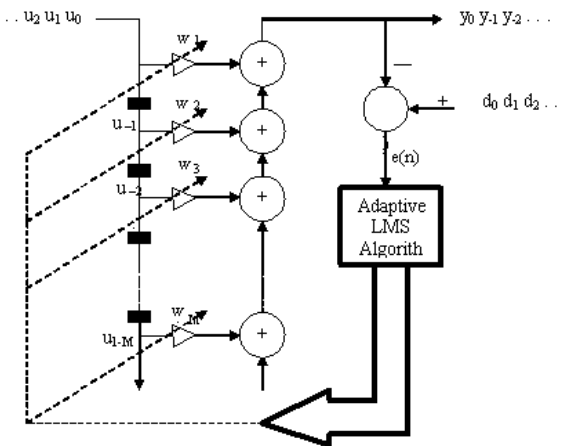


Fig.7. A Staged-Adder Canonical Form Realization of Adaptive Transversal FIR Filter Using LMS Algorithm.

In practice, the multiport adder has to be realized by a sequence of additions carried out by dual-input adders [13]. A staged-adder canonical

form the realization of the LMS adaptive FIR digital filter is shown in Figure (7). Note that the structure shown can be considered as a chain of similar stages.

The structure shown in Figure(7) can be considered as a chain of similar stages. Figure (8) gives another representation of the same structure. The samples are transferred from cell to cell via

delay elements, i.e. we have pipelining in the u-data flow. On the other hand, no delay elements are arranged in the interconnections via which the y-data is transferred. The addition of the products, which belong to one sample of the output signal, is carried out by the rippling of the signals from right to left in a single period [15].

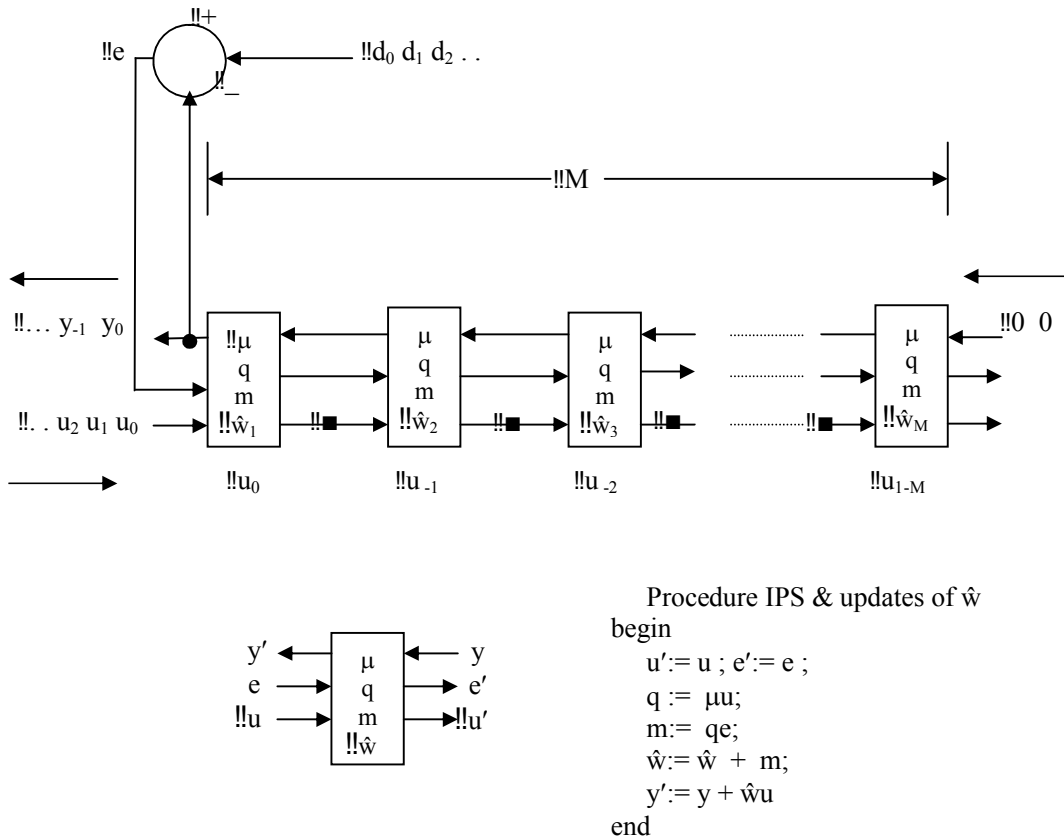


Fig.8. A Semi-Systolic Array Representation of LMS Algorithm-Based Adaptive Transversal FIR Filter.

Designs with Broadcasting of the Input Samples

Figure (9) shows an alternative semi-systolic array in which the addition of the products, which belong to one output sample, is pipelined [15]:

Out Signal Moving Faster

Figure (10) shows another systolic array for the LMS. U-data and y-data move in the same direction (such an array is classified as unidirectional) but at different velocities which are due to the different number of delay elements

in the respective interconnections. Zeros are input into the upper input of the leftmost cell. When moving to the right, these data items collect products of coefficients and input signal samples. The data items shown in parentheses above the array are intermediate results corresponding to the samples of the output signal [15]!

Input Signal Moving Faster

Another unidirectional systolic algorithm for the LMS is specified in Figure (11). In this case, the u-data moves faster than the y-data!

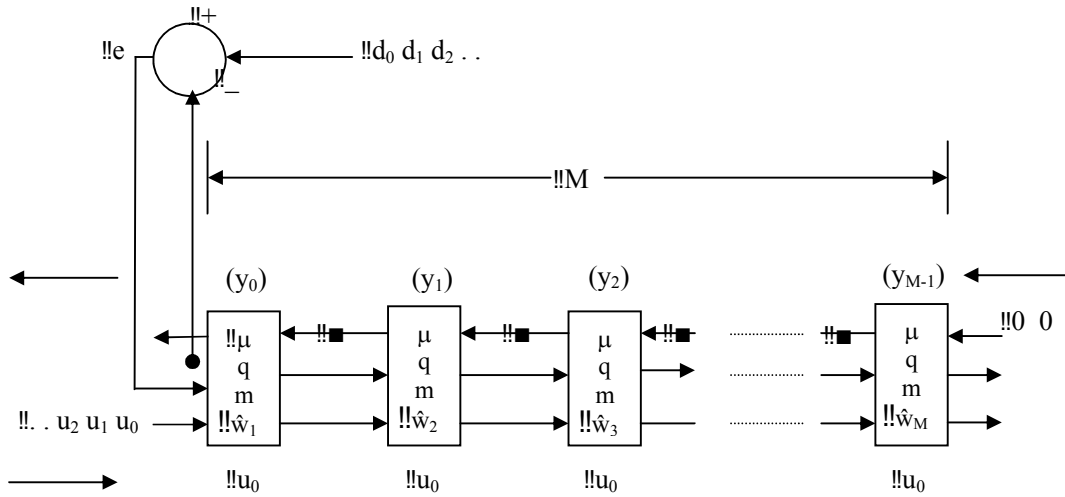


Fig.9. A Semi-Systolic Array Representation of LMS Algorithm-Based Adaptive Transversal FIR Filter with Broadcasting of the Input Signal Samples.

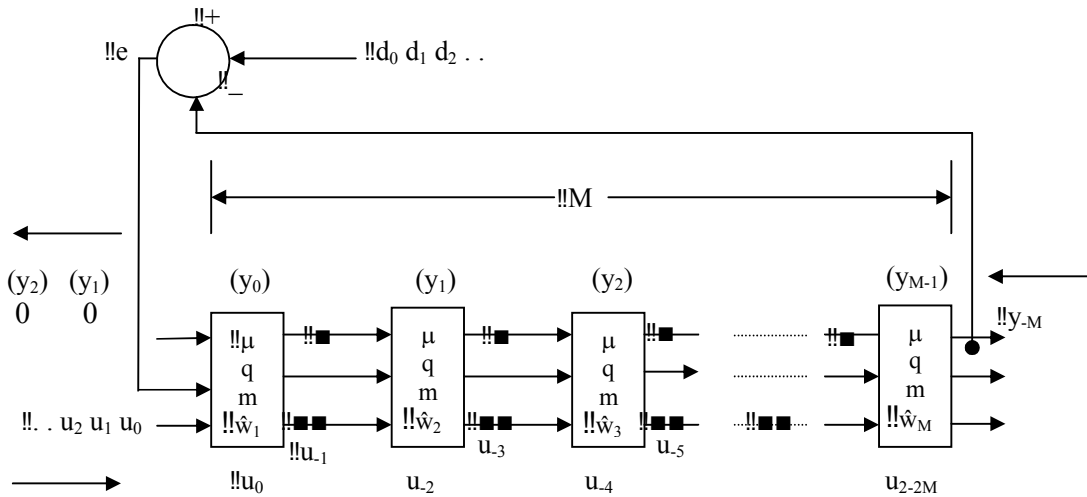


Fig.10. A Semi-Systolic Array Representation of LMS Algorithm-Based Adaptive Transversal FIR Filter in Which the y-Data Moves Twice as the u-Data.

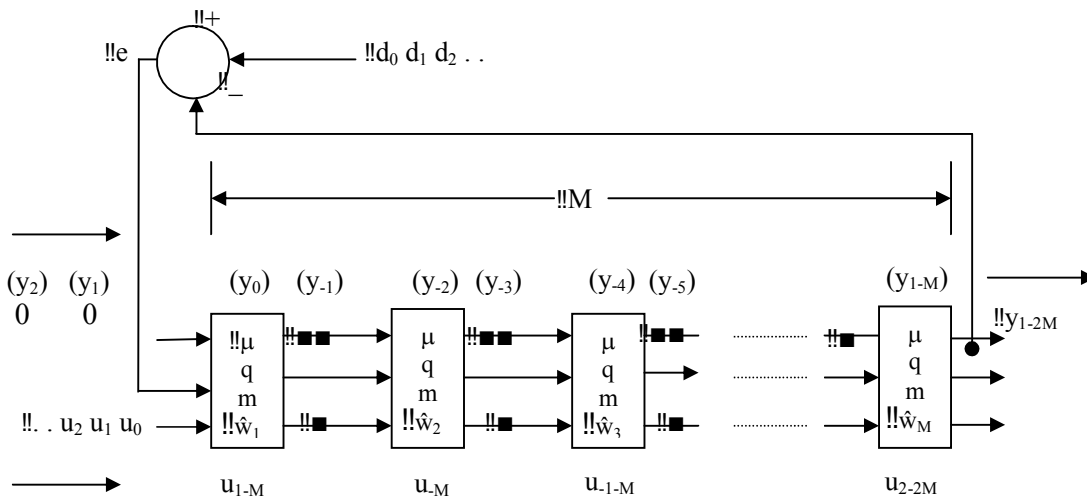


Fig.11. A Semi-Systolic Array Representation of LMS Algorithm-Based Adaptive Transversal FIR Filter in Which the u-Data Moves Twice as the y-Data.

(2)-Slow LMS Algorithms

The systolic algorithm of Figure (12) makes use of bidirectional data flow. In this algorithm, u-data and y-data move at the same velocity but in opposite direction.

If there is a second signal u^* which has to be processed with the same coefficient set to produce a second output signal y^* , the utilization can be improved. In this case, the sampling of the additional signals replaces the nil-data in the original algorithm, as shown in Figure (13) [15].

A Bidirectional LMS Algorithm with 100% Utilization

A bidirectional systolic LMS algorithm, which makes use of a completely pipelined systolic convolution algorithm and exhibits of 100% utilization, will be presented [1].

The systolic array to be described below makes use of a cell which stores two filter coefficients denoted by v and t , Figure (14). The cell is capable of executing in each clock period one of the two alternative function options depending on the type of the input data! For the LMS algorithm with a set of M coefficients, the algorithm requires an array of $M/2$ cells (M -even) [15].

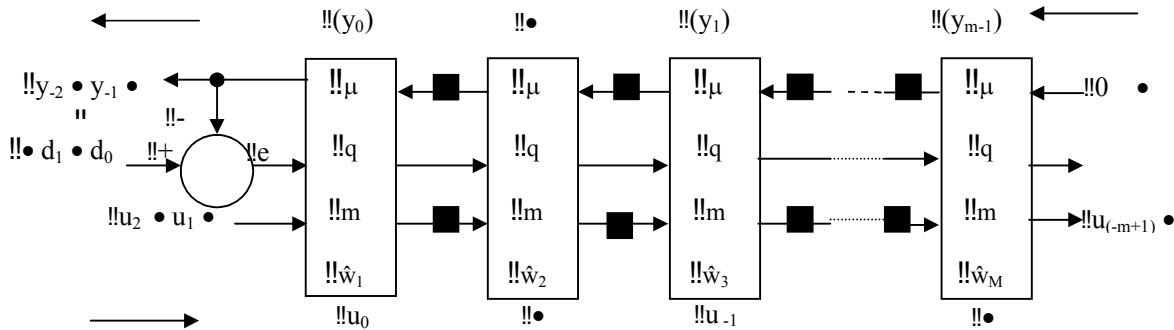


Fig.12. A bidirectional LMS Systolic Array!

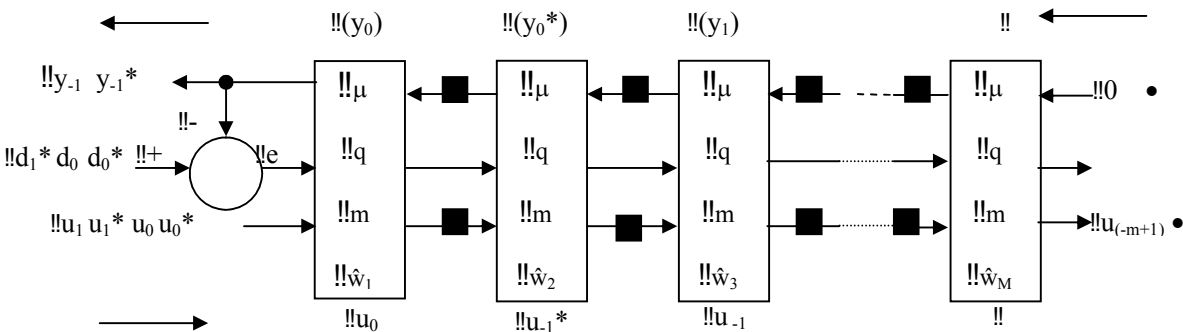
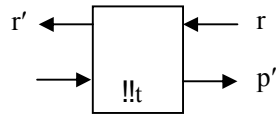


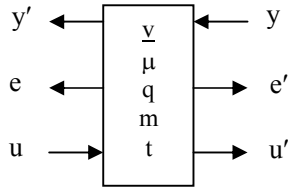
Fig.13. Concurrent Execution of Two LMS Algorithms Using the Same Coefficient Set!



```

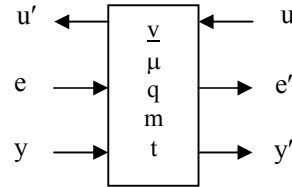
Procedure inner product step
begin
  if (p is an input signal sample) then
    begin p':= p ; r':= r + vp end ;
  else
    begin r':= r ; p':= p + tr end ;
end
    
```

(a)



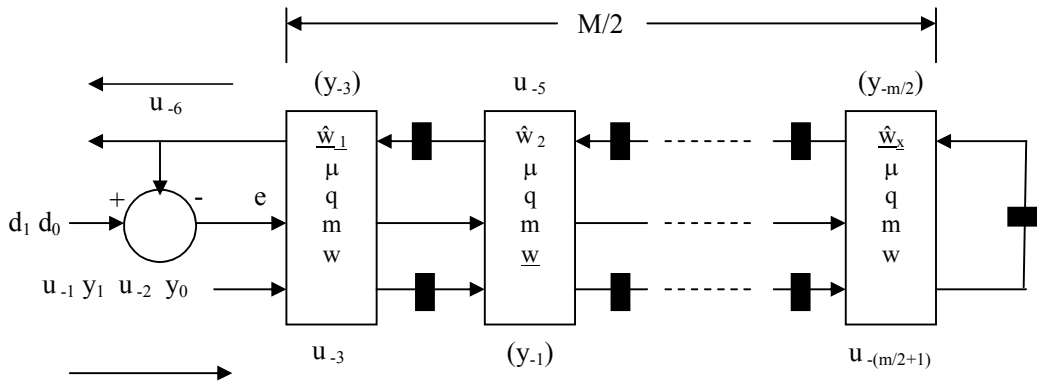
```

u':= u ; e':= e ;
q:= muu;
m:= qe;
v:= v + m;
y':= y + vu;
    
```



```

u':= u ; e':= e ;
q:= muu;
m:= qe;
t:= t + m!
y':= y + tu
    
```



(b)

Fig.14. A Bidirectional Systolic Array Realization of LMS Algorithm to Increase the Utilization. Knowing that $x = (M/2) + 1$, and Each Bottom w Equals $(x+1)$.

4. Throughput and Processing Rate

Now by using the systolic arrays and applying the simulation of data flow, it is found that:

- All systolic arrays have the same number of cells, which is equal to M , except the systolic array of Figure (14), designed with 100% utilization, and has $M/2$ cells.

- Running time is different from one array to another as follows:
- The array that is designed with broadcasting of the input samples takes M -clock periods.
- The unidirectional systolic arrays with output, input signal moving faster respectively have a number of clock periods twice that of the tap-weight vector, i.e., $2M$ clock periods.

- The arrays with bidirectional data flow (2-slow algorithm and concurrent execution of two LMS algorithm using the same coefficient) and the last array that is designed with 100% utilization, both require 2M clock periods.
- Commutative product-type measures for the array with broadcasting of the input signal is M^2 , for the unidirectional systolic arrays and for the arrays with bidirectional data flow is $(2M)M$, and the last array, designed with 100% utilization is $(M/2)2M$.
- Efficiency and utilization for all the cells is not exactly 100% due to the broadcasting of the error signal to each cell of these arrays, but for the array of 2-slow algorithm also is not exactly 50% for the same reason.
- Broadcasting happened in all the systolic arrays due to the broadcasting of the error signal samples. For this reason these arrays are classified as a semi-systolic arrays.
- Throughput and processing rate is the same magnitude of all systolic arrays designed, since these have same assignment form for each cell, therefore, the throughput is:

$$f_{LMS} = \frac{1}{1T_x + 1T_+ + T_{con.}} = \frac{1}{2T_x + 2T_+} \dots(13)$$

Where T_+ is an addition consumption time, T_x is a multiplication consumption time, and T_{con} is the convolution time. The latency is M-sample delay in the output.

A comparison of processing rate and throughput between our implementation and other parallel and serial algorithms can be made. In [16], a serial implementation of the transform-domain FIR adaptive filter operating by FFT is described; it requires a number of computations per block equal to $(3L/2) \log_2 L$ multiplications and $3L \log_2 L$ additions for two FFT's and one inverse FFT in addition to $3L$ multiplications and $2L$ additions for the adaptation rule. These computations must be performed during a time interval of LT_s seconds where T_s is the time period of the system clock. To increase the processing rate of the serial system, LT_s can be made at least equal to the time required by all these computations. Hence, T_s can at least be made equal to:

$$T_s = [(3/2) \log_2 2L + 3] T_x + [3 \log_2 L + 2] T_+ \dots(14)$$

Using the number of multiplications to demonstrate the difference in processing rate, it may be seen upon comparing Equations (13) and (14) that the processing rate of our implementation is always greater than that of the serial FFT system. To compare the systolic array implementation of the inner product (convolution sum) $w^H(n)u(n)$ that decreased latency and increased the throughput rate, with the serial operations. Knowing that convolution, which takes time proportional to L^2 , the FFT-based convolution requires time proportional to $L(1+2\log_2 L)$, but the implementation latency is proportional to L , ($M = L$). That difference can make FIR filter evaluation by FFT-a process known as fast convolution- more efficient for FIR filter past a certain length [13]. A particularly efficient implementation of the wavelet transform (Mallat's pyramid or tree algorithm) requires computation proportional (approximately) to L [16].

Comparison of the throughput of the preferable systolic array realization of LMS algorithm with more conventional architectures for delayless LMS adaptive filtering is made. The throughput of the conventional architecture is approximately

$$f_{conv.} = \frac{1}{2T_x + (L+1)T_+}$$

This throughput can be increased if a binary-tree adder architecture is used [8]; this structure is not regular, however, and is difficult to implement in VLSI, in which case, the throughput becomes

$$f_{bin} = \frac{1}{T_x + [\log_2 L + 1]T_+}$$

where $\lceil \log_2 L + 1 \rceil$ denotes the next largest integer value of $\log_2 L + 1$. For the pipelined architecture [8], they assume an adaptive noise canceling configuration for the filter, in which case, only the error signal is of interest. It is seen that the elemental computation within the structure is a single multiply and two adds; then the throughput is given by

$$f_{pipe} = \frac{1}{T_x + 2T_+}$$

The throughput of the implementation is greater than the previous two architectures, but smaller than that of the pipelined architecture which has drawbacks, over our implementation, of broadcasting the input signal. The overall

complexity (high) is linear in the number of filter coefficients if μ is chosen to be a power of two, it is not represented in a systolic form and its output latency is equal to the FIR filter length (i.e., L).

To compare our designed systolic arrays with the Frequency-domain Block LMS of [17], in [17] the throughput rate is given by:

$$f_B = \frac{1}{3 T_x + 2 T_+}$$

Which is less than the efficient implementation, and the latency is proportional to $(2L + 1)$. All cells have not single assignment form, it needs a large number of cells to implement a suitable block LMS array, and finally the utilization is not 100% because the neighboring rows of the matrix are input skewed by one clock period. But the algorithm has single assignment form, L - cells, and throughput greater than the above, 100% utilization, L latency, L -clock periods for one sample block and finally the flexibility in the design, which takes the inner product (convolution sum) $w^H(n)u(n)$ in the design consideration, which enables to extract more than one algorithm for the same problem.

And last but not least, comparing the efficient implementation of the algorithm (in systolic array) with the other implementations when the latter are implemented digitally.

The adaptive work [18], introduced a new source of error, namely, quantization precludes the tap weights reaching their optimum Wiener setting and causes a large increase in the total output mean squared error, compared to the pure analog (infinite precision) form of the algorithm.

A practical solution [14], for combating the arithmetic error is to use more bits for the tap weights than for the data but that leads to excessive complexity in the design.

Sixty-four bits are considered to be the minimum for representing a real number in scientific and technical computations, therefore, systolic arrays that operate on (input, output, or internal) variables whose values have to be represented by multibit numbers are called word level systolic arrays [1]; they can be placed on one VLSI chip.

5. Conclusions

- ❖ Various systolic arrays of LMS-based adaptive (FIR) transversal digital filters that were

designed can be effectively implemented using only local communications on a parallel system comprised of combinatorial circuits, clocked delay elements and distributed memory.

- ❖ The technique described exploits the recurrence inherent in the application and is based on the convolution sum (inner product $\hat{w}^H(n)u(n)$) in the LMS algorithm, which gives flexibility of deriving more than one systolic algorithm from the basic systolic structure.
- ❖ Proposed systolic arrays for 1-D convolution sum allow an increase in the number of processors to increase throughput rate and to reduce the latency that is not achievable using the conventional systolic array synthesis method.
- ❖ Unlike the other structures, the throughput is independent of the filter length, implying that LMS adaptive FIR filter systolic array with several hundreds of filter coefficients can be represented by a word-level systolic arrays (multibit numbers), when Considering the inner product step as a typical operation involved in many word-level systolic arrays for various signal processing operations.

There is also good future work on a 2-D LMS adaptive nonrecursive (or FIR) digital filters with excitation $u(n_1, n_2)$, response $y(n_1, n_2)$ and order of the pair (M_1, M_2) to extract a systolic array for the algorithm, and also by taking 2-D inner product step of the weight vector and the input vector as a basic systolic representation of the algorithm which can design it by the dependence graph method.

6. References

- [1] N. Petkov, "Systolic Parallel Processing". Elsevier Science Publishers, North Holland, 1993.
- [2] S. d. Peters and A. A. Antoniou, "A Parallel Adaptation Algorithm for Recursive Least-Squares Adaptive Filters in Nonstationary Environments," IEEE Trans. Signal Processing, vol. 43, no. 11, pp. 2484-2494, Nov. 1995.
- [3] S. C. Douglas, Q. Zhu, and K. F. Smith, "A pipelined LMS Adaptive FIR Filter Architecture Without Adaptation Delay," IEEE Trans. Signal Processing, vol. 46, pp. 775-778, Mar. 1998.
- [4] N. R. Shanbhag and K. K. Parhi, "Pipelined Adaptive Digital Filters", Boston, MA: Kluwer, 1994.

- [5] P. Kabal, "The stability of adaptive minimum mean square error equalizers using delayed adjustment," IEEE Trans. Commun., vol.COMM-31, pp.430-432, Mar.1983.
- [6] D. L. Jones, "Learning characteristics of transpose-form LMS adaptive filters," IEEE Trans. Circuits Syst. II, vol. 40, pp. 745-749, Oct.1992.
- [7] T. H. Y. Meng and D. G. Messerschmitt, "Arbitrarily high sampling rate adaptive filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, pp. 455-470, Apr. 1987.
- [8] Q. Zhu, S. C. Douglas, and K. F. Smith, "A pipelined LMS Adaptive FIR Filter Architecture Without Adaptation Delay," IEEE Trans. Signal Processing, vol. 46, No. 3, Mar. 1998.
- [9] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," IEEE Trans. Acoustic., Speech, Signal Processing, vol. 37, pp. 1397-1405, Sept. 1989; vol. 40, pp. 230-232, Jan. 1992.
- [10] H. Hersberg, R. Haimi-Cohen, and Y. Be'ery, "A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation," IEEE Trans. Circuits Syst. II, vol. 40, pp. 2799-2803, Nov. 1992.
- [11] M. Rupp and R. Frenzel, "The behavior of LMS and NLMS algorithms with delayed coefficient update in the presence of spherically invariant processes," IEEE Trans. Signal Processing, vol. 42, pp. 668-672, Mar. 1994.
- [12] F. Lorenzelli and K. Yao, "A Linear Systolic Array for Recursive Least Squares," IEEE Trans. Signal Processing, vol. 43, no. 12, pp.3014-3020, Dec. 1995.
- [13] B. Widrow and S.D. Stearns, "Adaptive Signal Processing", Prentice Hall, Englewood Cliffs, NJ, 1985.
- [14] S. Haykin, "Adaptive Filter Theory", Englewood Cliffs, NJ: Prentice-Hall, 1986.
- [15] R. A. H. Al-Helali, "Systolic Algorithms of LMS, RLS Adaptive (FIR) Digital Filters for Adaptive Channel Equalization," M.Sc. Thesis, Univ. of Baghdad, Baghdad, Oct. 2005.
- [16] D. Grover, J. R. Deller: Digital Signal Processing and the Microcontroller. Prentice Hall, 1999.
- [17] N. A. S. Alwan and B. A. R. Al-Hashemy, "Systolic design of frequency-domain block LMS adaptive digital filters," Comput. Electr. Eng. 24, 263-375 (1998).
- [18] J. Carlos, M. Bermudez and N. J. Bershad, "A Nonlinear Analytical Model for the Quantized LMS Algorithm-The Arbitrary Step Size Case," IEEE Trans. Signal Processing, vol. 44, no. 5, pp. 1175-1183, MAY 1996.

مصفوفة من المعالجات المتوازية ذات البعد الواحد (1-D) لترشيح الإشارات الرقمية بطريقة أَل (FIR) باستخدام خوارزمية أَل (LMS).

رياض علي عبد الحسين الهلالي* باقر عبدالرسول الهاشمي** علي حيدر مهدي***

* قسم الهندسة الكهربائية/كلية الهندسة/الجامعة المستنصرية

** قسم الهندسة الكهربائية/كلية الهندسة/جامعة بغداد

*** قسم هندسة المعلومات/كلية الهندسة الخوارزمي/جامعة بغداد

الخلاصة

هذه المقالة تتفحص هيكلية المصفوفات المتوازية ذات البعد الواحد (1-D) لترشيح الإشارات الرقمية المتوازية. إن الطور المصد تخدم لتحويل المرشح ذو النوع (FIR) خدام تقنية مربع أدنى معدل إلى هيكلية متوازية وتركيب يشبه بعمله طريقة الملاءمة والتفريغ الأندوبيترافها. أما نتائج أخيرات زمنية في تجديد قيم المعاملات للمرشح أو تحتاج إلى متطلبات مادية (دوائر خارجية) إضافية.

في هذا التصديقنا مصفوفة من المعالجات المتوازية ذات البعد الواحد (1-D) للمرشح الملائم (LMS) الذي يملك خواص إشارات الإخراج والخطأ مشابه لتلك المتولدة في حالة التصميم الاعتيادي للمرشح القياسي الملائم (LMS) وذلك باستخدام صيغة منفردة من معادلات المرشح. معيارية المصفوفة المفروضة تعمل على أساس دخول البيانات مجموعة بعد مجموعة بالاسبقادة من خواص مرونة التصديق الذي بدوره يعتمد على خطوة الالتفاف الرياضي (CONVOLUTION SUM) بين متجه الإدخالات ومتجه المعاملات للمرشح التي مكنتنا من استخراج أكثر من خوارزمية واحدة لنفس المسألة.

أن الانسياب المتسلسل والمستمر للإدخال والإخراج من وإلى خارج المنظومة تعتمد على معدل نبضات المنظومة لإزالة كل فترة زمنية في أن العناصر المعالجة لنفس النوع تعمل بشكل متوازي.

أن الحسابات الرياضية تطلب عمليتي ضرب متتابعين وعمليات جمع/طرح لكل نبضة زمنية وهذا ينتج معدل إخراج عالي جداً ومنظومة معالجات سريعة جداً" مقابل تأخير زمني مقداره L من النماذج (Samples) بين الإدخال والإخراج وكذلك كفاءة 100% .