THE DESIGN OF RESIDUE NUMBER SYSTEM ARITHMETIC UNITS FOR A VLSI ADAPTIVE EQUALIZER

Inseop Lee and W. Kenneth Jenkins

Department of Electrical and Computer Engineering and
The Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 West Main Street, Urbana, IL 61801
email: is-lee@uiuc.edu and jenkins@uicsl.csl.uiuc.edu

ABSTRACT

This paper presents the design details of an experimental ASIC for an all-digital adaptive equalizer. In this design, the LMS algorithm is chosen because of its simplicity. The adaptive equalizer design, which is based on an RNS architecture, consists of an RNS multiplier, an RNS adder, an RNS filter, a binary-to-residue converter, a residue-to-binary converter, and an update algorithm. The design is verified by a high level hardware simulation tool. The designs of all these units are discussed in this paper.

1. INTRODUCTION

As the transmission rate increases, high speed adaptive equalization is necessary. The most popular and widely used adaptive algorithm in practical applications is the least-mean-square (LMS) algorithm because it is a simple but powerful algorithm. Although the LMS algorithm is a simple adaptation algorithm, implementing a high speed and area efficient adaptive equalizer is not a simple problem because a considerable number of multiplications is required to implement the update equation and filter [1]. To overcome this complexity in the LMS algorithm, modifications were made on the LMS algorithm. Examples of these modifications are the use of the sign LMS, the block LMS and the delayed LMS algorithms. These algorithms are not the exact optimal solutions, but for some applications, the performance is acceptable [2].

Another approach to solving this problem is to use RNS arithmetic to achieve high speed multiplication. The residue number system (RNS) has been a successful alternative arithmetic for high-speed finite impulse response (FIR) filters [3]. Because of its carry-free property, it accommodates parallel implementation for high speed applications. However, a scaling operation, which is a simple shift operation in the binary number system, is not simple in the RNS [4]. Because of this cumbersome scaling operation, the RNS-based implementation of the LMS algorithm has not been attractive as a general approach.

The design strategy used here moves the scaling operation into the binary number system. This approach requires increasing the internal dynamic range. To limit the growth of this internal dynamic range, which is increased to avoid scaling within the RNS, and to improve the finite precision performance, a modified LMS algorithm was proposed and demonstrated in [5]. Also, to avoid the large computational complexity of residue-to-binary conversion, an approximated Chinese remainder theorem (ACRT) is implemented [5]. Figure 1 shows block diagram of the RNS implementation of the modified LMS algorithm.

2. THE RESIDUE NUMBER SYSTEM

The binary number system is called a weighted number system because each digits have different weights. A number X is represented in the binary number system as

$$X = b_{n-1}b_{n-2}\cdots b_0 = \sum_{j=0}^{n-1} b_j 2^j \qquad (1)$$

$$b_j = 0, 1$$

where

X : Nonnegative integer, $0 \leq X < 2^n$.

This X also can be represented by a non weighted number system using the RNS. X in the RNS is

$$X = (r_1, r_2, \cdots, r_n), \qquad (2)$$

0-8186-8409-7/98 $10.00

$$r_j = X \bmod m_j, \quad (3)$$

$$X = q_j m_j + r_j, \quad (4)$$

where

$$j = 1,2, \cdots, N.$$

r_j is called residue of X modulo m_j. $M = \prod_{j=1}^{N} m_j$ is the dynamic range of this number system. If $X \leq M$, then X can be uniquely represented by the RNS and can be perfectly converted to X from its RNS representation. This conversion process from the RNS to the binary number system is based on the Chinese Remainder Theorem(CRT). The CRT states that if m_j and r_j are given, $X \leq M$, and $(m_j, m_k)(j \neq k)$ pairs are relatively prime, then X can be recovered by

$$|X|_M = |\sum_{j=1}^{N} \bar{m}_j |\frac{r_j}{\bar{m}_j}|_{m_j}|_M \quad (5)$$

where,

$$m_j = \frac{M}{m_j} \quad (6)$$

$$M = \prod_{j=1}^{N} m_j \quad (7)$$

$$|\cdot|_M = \cdot \bmod M \quad (8)$$

$|\frac{1}{\bar{m}_j}|_{m_j}$: Multiplicative inverse which satisfies

$$|\bar{m}_j \cdot \frac{1}{\bar{m}_j}|_{m_j} = 1. \quad (9)$$

By this old theorem and some restrictions on numbers (pairwise relatively moduli, dynamic range of X), we can represent numbers in different ways without losing any information. Arithmetic operations in the RNS can be categorized by two groups, simple and difficult operations. Addition, subtraction and multiplication belong to the simple operation group. Division, scaling, sign detection and magnitude comparison belong to the difficult group. Addition and subtraction in the RNS is

$$\begin{aligned} X &= (x_1, x_2, \cdots, x_n) \\ Y &= (y_1, y_2, \cdots, y_n) \\ |X \pm Y|_M &= (|x_1 \pm y1|_{m_1}, |x_2 \pm y2|_{m_2}, \cdots \\ &\quad , |x_N \pm yN|_{m_N}). \end{aligned} \quad (10)$$

Multiplication in the RNS is given by

$$\begin{aligned} X &= (x_1, x_2, \cdots, x_n) \\ Y &= (y_1, y_2, \cdots, y_n) \\ |X \cdot Y|_M &= (|x_1 \cdot y1|_{m_1}, |x_2 \cdot y2|_{m_2}, \cdots, \\ &\quad |x_N \cdot yN|_{m_N}). \end{aligned} \quad (11)$$

It is easily observed that there is no carry propagation in these operations and this is the most attractive property of the RNS. However scaling by a fixed number in the RNS is not a simple operation because all digits are not weighted, although in a weighted number system, like the binary number system, it is a simple shift operation. This scaling operation in the RNS includes sign detection and division. Both belong to the difficult group of operations. Details of these operations can be found in [4].

3. THE RNS IMPLEMENTATION OF THE LMS ALGORITHM

As mentioned in Section 1 and 2, scaling in the RNS is a very high-cost operation. Therefore it is desirable to eliminate this operation or to implement it in the binary number system. It is impossible to eliminate the scaling in the fixed point implementation of the LMS algorithm because of its feedback structure. However, it can be done in the binary number system rather than in the RNS. This approach can eliminate scaling operations in the RNS but requires increasing the number of bits for internal processing.

Assume that $N = 2^{B-1}$, where B is the number of bits. The coefficients $\hat{\mathbf{w}}(n)$, data $\mathbf{x}(n)$, and step size μ are quantized using B bits, and shifting operations are required at every multiplication. Therefore, the LMS update equation in fixed point arithmetic is,

$$\begin{aligned} \hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n) + \mu \mathbf{x}(n)(\frac{1}{N^2} d(n) \\ &\quad - \frac{1}{N^3} \mathbf{w}^H(n)\mathbf{x}(n)). \end{aligned} \quad (12)$$

According to eq.(12), if the scaled desired signal, $\frac{1}{N^2} d(n)$ is provided and $\frac{1}{N^3}$ is formed by shifting in the binary number system, scaling in the RNS is no longer a problem. However, to implement this, a large number of bits are required to represent $\hat{\mathbf{w}}^H(n)\mathbf{x}(n)$. For B=6, 24 bits are required to represent $y(n) = \hat{\mathbf{w}}^H(n)\mathbf{x}(n)$ properly. This architecture has a serious drawback when high resolution is required, because a large number of internal bits are needed. All coefficients and data are quantized using B bits and $Q_{B-1}[\cdot]$ is a quantizer with B bits. Then,

$$\begin{aligned} \hat{\mathbf{w}}(n+1) &= \mathbf{w}(n) + \mu \mathbf{x}(n)(Q_{B-1}[d(n)] - Q_{B-1}[y(n)]) \\ &= \mathbf{w}(n) + \mu \mathbf{x}(n)(Q_{B-5}[d(n)] + Q_{4-1}[d(n)] \cdot 2^{-4} \\ &\quad - Q_{B-5}[y(n)] - Q_{4-1}[y(n)] \cdot 2^{-4}) \end{aligned} \quad (13)$$

If $Q_{4-1}[d(n)] = 0$,

$$\begin{aligned} \hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n) + \mu \mathbf{x}(n)(Q_{B-5}[d(n)] - Q_{B-5}[y(n)] \\ &\quad - Q_{4-1}[y(n)] \cdot 2^{-4}) \end{aligned} \quad (14)$$

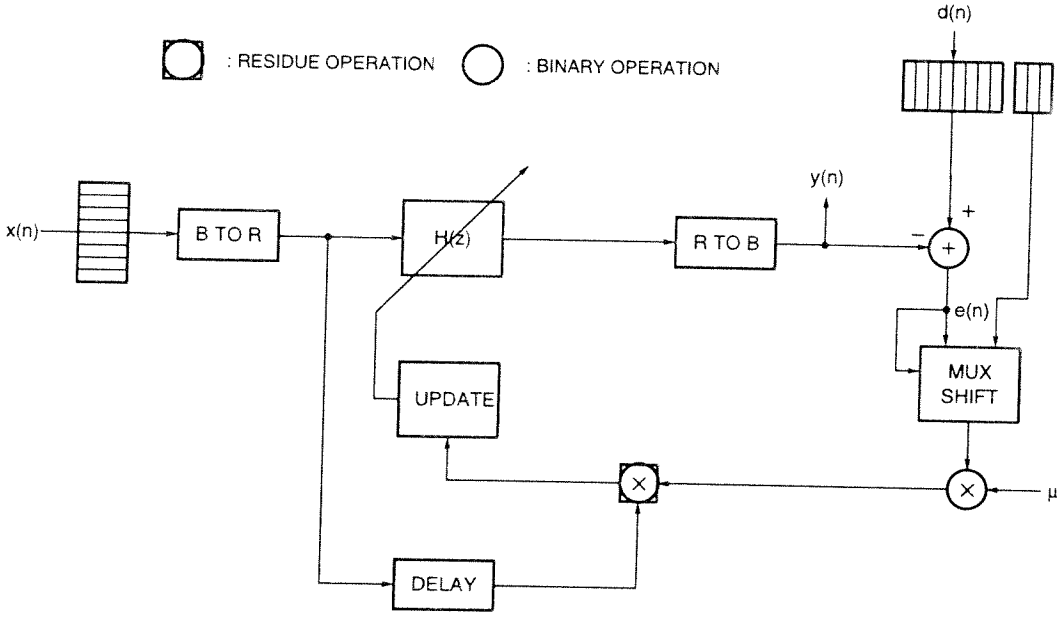

Figure 1: Block diagram of the RNS implementation of the modified LMS algorithm.

After stalling occurs, $Q_{B-5}[d(n)]-Q_{B-5}[y(n)]=0$, then

$$\hat{\mathbf{w}}(n+1)=\hat{\mathbf{w}}(n)+\mu Q_{4-1}[\mathbf{x}(n)(-y(n)\cdot 2^{-4})]. \quad (15)$$

Eq.(15) has the same effect as a quantizer with more bits.

4. DESIGN OF THE RNS ARITHMETIC UNITS

A relatively large complexity for the residue-to-binary converter limits the RNS from being used in many digital signal processing applications. Eq.(5) can be rewritten using modular algebraic operations and scaling [7].

$$X\cdot\frac{2^d}{M}=|\sum_{j=1}^{N}|\frac{r_j}{m_j}|_{m_j}\cdot\frac{2^d}{m_j}|_{2^d}. \quad (16)$$

In eq.(16) we can apply the following approximation to simplify the hardware architecture:

$$\frac{1}{m_j}\cdot|\frac{r_j}{m_j}|_{m_j}\approx\frac{k}{m_j}, \quad (17)$$

$$R(k)=\frac{2^d\cdot k}{m_j}\approx\lfloor\frac{2^d}{m_j}\rfloor\cdot k, \quad (18)$$

where k is an integer and $0\le k<m_j$. Finally, $X\cdot\frac{2^d}{M}$ can be computed by summing the precomputed ROM outputs, $R(k)$, according to eq.(16), resulting in

$$X\cdot\frac{2^d}{M}\approx|\sum_{j=1}^{N}R(|\frac{r_j}{m_j}|_{m_j})|_{2^d}. \quad (19)$$

Figure 2 shows a block diagram of an ACRT implementation [5]. Inputs of this ACRT are RNS encoded numbers and ROMs contain precomputed values of eq.(18). And the implementation of power of 2 modulo adder is the same as a binary adder with floating overflow bit. To reduce errors cause by using finite precision ROMs, LSBs are truncated. Bias control is realized by two comparators, a multiplexor and a binary adder. The truncated output is compared to two preset values to determine the bias and then subtracted to correct bias.

The binary-to-residue converter is designed based on the algorithm which is described in [3, 6]. A K bit number X can be expressed,

$$X=\sum_{j=0}^{K1}b_j2^j+\sum_{j=K1+1}^{K}b_j2^j, \quad (20)$$

where b_j is the sign bit. The residue of X mod m_i, where $m_i, i=1,\cdots,N$, is the i-th modulus used to define the RNS code, can be written using eq.(20),

$$\begin{aligned}|X|_{m_i}&=|\sum_{j=0}^{K1}b_j2^j+\sum_{j=K1+1}^{K}b_j2^j|_{m_i}\\&=|\sum_{j=0}^{K1}b_j2^j|_{m_i}\ominus|\sum_{j=K1+1}^{K}b_j2^j|_{m_i}. \quad (21)\end{aligned}$$

where $\oplus$ represents modulo addition. According to eq.(21), the binary-to-residue converter can be implemented by two precomputed ROMs and a modulo adder. For the smaller modulus, it is simpler to use programmable logic arrays (PLAs) instead of ROMs. In this design we use both approaches to reduce hardware complexity. A block diagram is given in figure 3.

One of the primary building blocks is the modulo adder. The basic architecture of an adder used in this system is the vector merging adder (VMA) with overflow control, which is suitable for high speed application because it is pipelined to reduce carry propagation delay. To guarantee high speed operation of this adder, for large moduli, the gate delay should be considered in the design of overflow control circuit. A design example of the VMA architecture is given in figure 4. Design of the modulo multipliers is based on a pipelined square law multiplier which uses two ROMs and 3 modulo adders for each modulo multiplier [3, 6]. With a slight modification this system can be implemented with 2 ROMs and 1 modulo adder. Taking values of multiplicands as the input to the ROMs with precomputed residues of values of square-of-sums ($|(A+B)^2|_{m_i}$) and square-of-differences ($|(A-B)^2|_{m_i}$), and adding outputs of ROMs with a modulo adder, produces the modulo multiplication. But the output of this multiplier is four times larger than the expected value ($4AB$). A block diagram is shown in figure 5. The binary multiplier needed for the coefficient update block is implemented using AND gates and adders, because the value of the step size is usually kept small for stability of the system. Implementing the multiplier using a combination of logic and adders can save considerable area and speed compared to a conventional multiplier design. FIR filter sections are designed by the pipelined FIR structures. To compensate all delays caused by pipelining, special attention is required. In figure 6 a structure of a pipelined FIR filter is given. To implement the update algorithm, modulo multipliers, modulo adders and delays are used. To initialize the filter coefficients, external initialization inputs are provided. Because each modulo multiplier and each modulo adder has different numbers of delays, also delay compensations are required. This is given in figure 7.

5. SUMMARY

This paper presents the design details of an experimental ASIC for an all-digital adaptive equalizer. Since the system level designs based on an RNS architecture are already established in a previous publication [5], this paper focuses on the design details of the RNS adaptive equalizer arithmetic units. The designs of all units, i.e. an RNS multiplier, an RNS adder, an RNS filter, a binary-to-residue converter, a residue-to-binary converter, and an update algorithm for the adaptive equalizer based on an RNS architecture are discussed in this paper, demonstrating the feasibility of an RNS implementation of the LMS algorithm. Design verification is performed by timing and logic simulation using a high level hardware simulation tool.

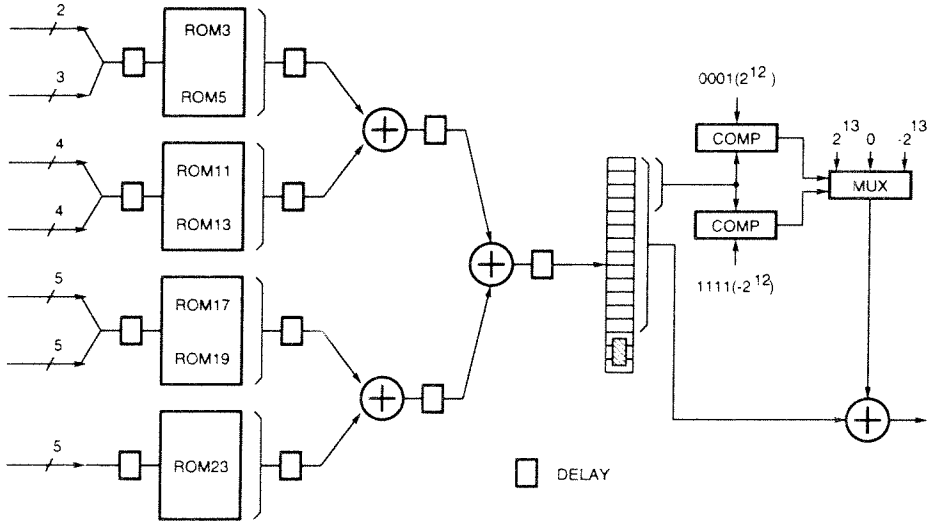

Figure 2: Block diagram of an ACRT for 16 bit precision.

6. REFERENCES

[1] C. S. H. Wong, J. C. Rudell, G. T. Uehara, and P. R. Gray, "A 50 MHz Eight-Tap Adaptive Equalizer for Partial-Response Channel," *IEEE J. of Solid-State Circuits*, vol.30, no.3, pp. 228-234, March 1995.

[2] S. Haykin, *Adaptive Filter Theory* 3rd Ed. New Jersey: Prentice Hall, 1996.

[3] N. R. Shanbhag and R. E. Siferd, "A Single-Chip Pipelined 2-D FIR Filter Using Residue Arithmetic," *IEEE J. of Solid-State Circuits*, vol.26, no.5, pp. 796-805, May 1991.

[4] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, New York, IEEE Press, 1986.

[5] Inseop Lee and W. Kenneth Jenkins, "VLSI Design for an Adaptive Equalizer Using a Residue Number System Architecture for Magnetic Channels," *Proceedings of 40th Midwest Symposium on Circuits and Systems*, Sacramento, CA, August, 1997, to appear.

[6] W. K. Jenkins and B. J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters," *IEEE Trans. on Circuits and Systems*, vol.CAS-24, no.4, pp. 191-201, April 1977.

[7] J. Y. Kim, K. H. Park, and H. S. Lee, "Efficient Residue-to-Binary Conversion Technique with Rounding Error Compensation," *IEEE Trans. on Circuits and Systems*, vol.38, no.3, pp. 315-317, March 1991.

Acknowledgment

This work is supported in part by the Joint Services Electronic Program (JSEP) under grant number N0001A-96-0129. The opinions expressed herein are not necessarily those of the sponsoring agency.

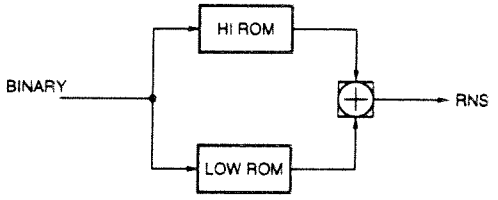

Figure 3: Block diagram of a binary-to-residue converter.

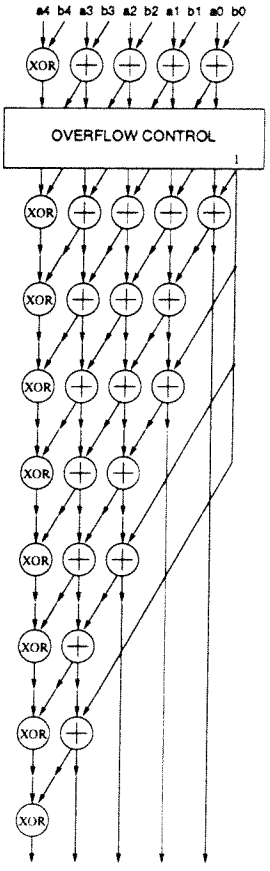

Figure 4: An example of VMA architecture for modulo 17 adder.

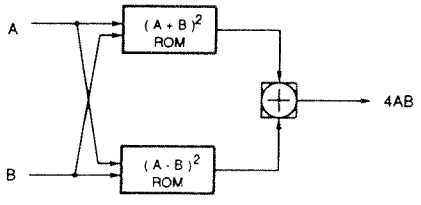

Figure 5: Block diagram of a modulo multiplier.

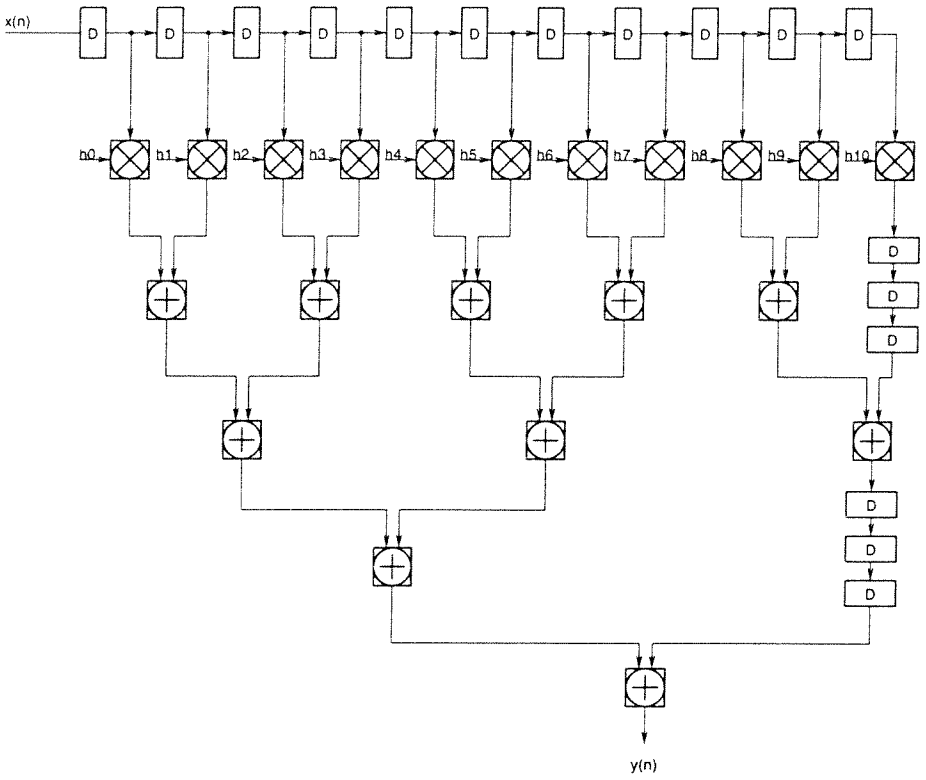

Figure 6: Block diagram of FIR filter.

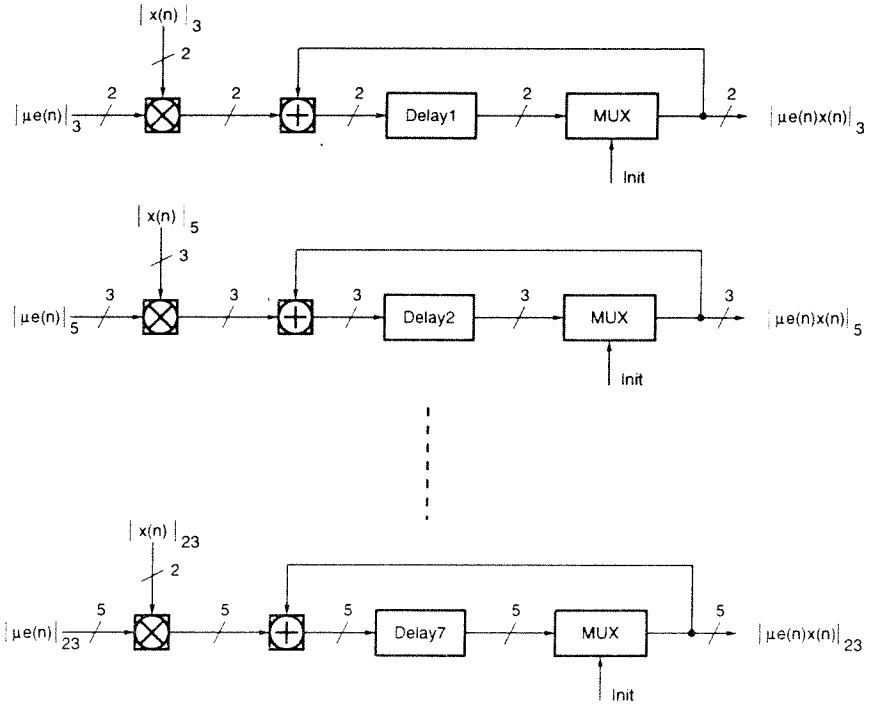

Figure 7: Implementation of update algorithm.