International Journal of Electronics and Communication Engineering.
ISSN 0974-2166 Volume 4, Number 1 (2011), pp.83-94

http://www.irphouse.com

Transformation Techniques for Real Time High Speed Implementation of Nonlinear Algorithms

D. Madhavi, N. Jyothi and P.L.H. Varaprasad*

GITAM Institute of Technology, GITAM University,
Visakhapatnam, Andhra Pradesh, India-530045.
E-mail: madhavi336@yahoo.com, jyothi_nutakki@yahoo.com
**Department of Instrument Technology, AU College of Engineering,*
Andhra University, Visakhapatnam, India

Abstract

The goal of any digital design is to maximize the performance while keeping the cost down. In digital communication systems, high speed transmission requires implementation of high speed digital circuits which include adaptive equalizers, encoders and other signal processing algorithms. DSP algorithms are described using mathematical formulations at a higher level. For architectural design, these mathematical formulations need to be converted to graphical representations which include block diagram, signal flow graph, data flow graph and data graph. Examples of these algorithms are differential pulse code modulation (DPCM), adaptive differential pulse code modulation (ADPCM), decision feedback equalizers(DFE's). If the recursive structures are present in these algorithms high speed implementation is difficult, since these structures contain nonlinear operations (such as the quantization operation).This paper proposes different computation approaches for quantization algorithms, which can be easily pipelined. These approaches are suitable for real time high-speed implementation of quantizer loop operations. The performance of the proposed systems is better by the use of tri-state buffers and decoders in the implementation.

Introduction

Graphical representations are efficient for analyzing data flow properties of DSP algorithms and are used to map DSP algorithms to hardware implementations. These representations describe the algorithms at various levels of abstraction. Pipelining of these structures is difficult because of the nonlinear nature.

There are different approaches to overcome the problem of pipelining. One approach we use here is to transform the existing algorithms into equivalent forms which possess greater concurrency. The transformations proposed by [1] are modified by the use of tristate buffer and decoder circuits whose performance in terms of power consumption and speed is better compared to the earlier approach. The transformation approaches presented here can be applied either in the context of dedicated VLSI implementations, or multiprocessor implementations. This paper proposes transformation approaches to transform quantizer loop operations into equivalent forms so that they can exploit look-ahead and can be pipelined at finer levels.

The organization of the paper is as follows. Section II studies transformation of finite-level quantizer loops (first order two level ,first order four level and second order two level) into equivalent forms containing no loop quantizers. Section III proposes transformation of differential pulse code modulation encoders into equivalent forms that can be pipelined at higher levels.

Transformation of First Order and Second Order Quantizer Loops

This section presents how to transform first order two level, first order four level and second order two level quantizer loops into equivalent structures without quantizer loop operation. This reformulation allows pipelining the equalizer algorithms, which improves the speed of the processing.

First order two level Quantizer loop

Consider the first order loop update operation described by the equation

$$y(n+1) = Q[\alpha(n)\, y(n) + x(n)] \tag{1}$$

and it is shown in Fig.1.

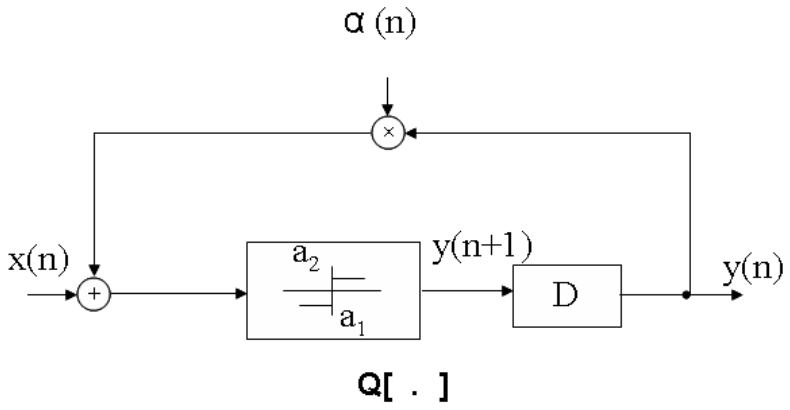

Figure 1: First order two level quantizer loop.

In equation (1), y (n) is the state which needs to be updated, x (n) is the input, $\alpha(n)$ is a time varying multiplier coefficient, and Q[.] is the Quantizer function. The quantized output y (n) at any time can be either a_2 or a_1 since it is a two level quantizer.

The minimum time required to update the loop in Fig.1 is equal to $(T_m+T_a+T_q)$, where, T_m ,T_a and T_q respectively, represent the computation times of a multiplier, an adder, and the quantization operation.

The input sequence x (n) cannot be processed with a sampling period less than $(T_m+T_a+T_q)$, since this time corresponds to the inherent iteration period bound imposed by the loop operation [4]. To improve the sample rate of this system, we can find the equivalent structure possessed with greater concurrency. In the equivalent system, we encode the quantized state y(n) using a binary state $y_1(n)$, and the loop update operation for y(n) is reformulated to update the binary state $y_1(n)$.

Define the binary state $y_1(n)$ as,

$$y_1(n) = 0 \text{ , if } y(n) = a_1$$
$$= 1 \text{ , if } y(n) = a_2 \tag{2}$$

Using the binary state, we can write

$$y(n) = a_1 \bar{y}_1(n) + a_2 y_1(n) \tag{3}$$

Define another binary q-function as

$$q_1(a) = 0 \text{ , if } Q[a] = a_1$$
$$= 1 \text{ , if } Q[a] = a_2 \tag{4}$$

The loop update operation in equation (1) can now be reformulated in terms of the binary state $y_1(n)$ and the binary function $q_1[.]$ as

$$\begin{aligned} y_1(n+1) = {} & y_1(n) q_1[\alpha(n) a_2 + x(n)] \\ & + \bar{y}_1(n) q_1[\alpha(n) a_1 + x(n)] \end{aligned} \tag{5}$$

where + is a binary OR operation, and the multiply is a binary AND operation. The equivalent reformulated quantizer loop is shown in fig.2.

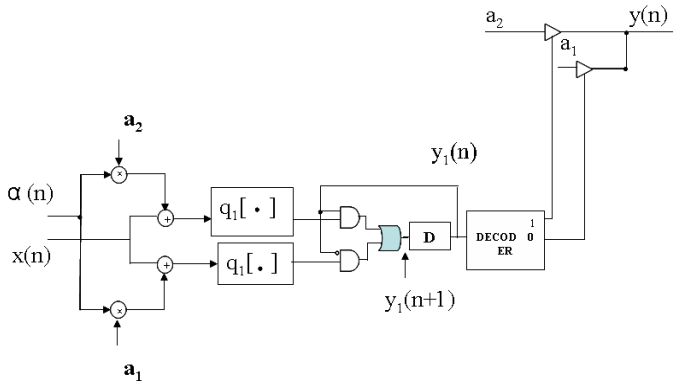

Figure 2: Equivalent transformed structure for first order two level quantizer loop.

The front and end computations in this structure are nonrecursive and can be pipelined at any level. The actual increase in speed is achieved after placing pipelining latches at appropriate locations.

First order four level Quantizer loop.

Consider the first order four level quantizer loop in Fig.3 .The quantized output y(n) can take four possible values a_1 , a_2 , a_3 , and a_4 .We encode the state y(n) using two binary states $y_1(n)$ and $y_2(n)$.

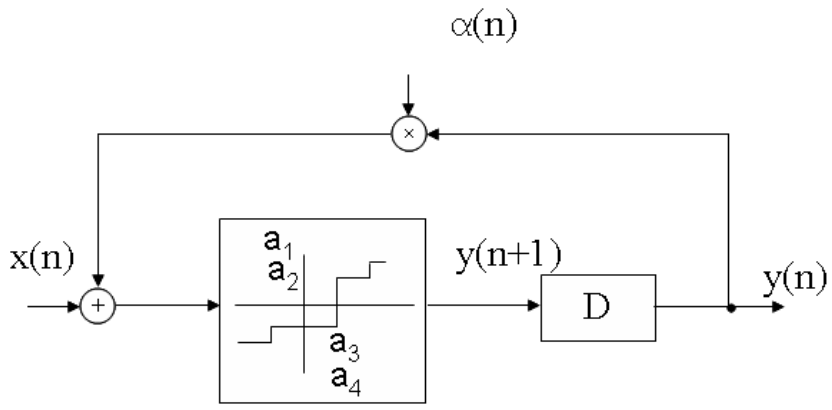

Figure 3: First order four level quantizer loop.

The state $y_1(n)$ is defined to be 1 if the output of the quantizer is a_1 or a_2,and is zero otherwise. The state $y_2(n)$ is defined to be 1 if the output of the quantizer is a_1 , or a_3 ,and is zero otherwise. Using these binary states, we can write

$$y(n) = a_4 \bar{y}_1(n) \bar{y}_2(n) + a_3 \bar{y}_1(n) y_2(n) + a_2 y_1(n) \bar{y}_2(n) + a_1 y_1(n) y_2(n) \quad (6)$$

Define the binary q-functions

$$q_1[a] = 1, \text{ if } Q[a] = a_1 \text{ or } a_2$$
$$= 0, \text{ if } Q[a] = a_3 \text{ or } a_4 \quad (7)$$

$$q_2[a] = 1, \text{ if } Q[a] = a_1 \text{ or } a_3$$
$$= 0, \text{ if } Q[a] = a_2 \text{ or } a_4 \quad (8)$$

We can then transform the original system into an equivalent system which updates the binary states $y_1(n)$ and $y_2(n)$ as

$$\begin{aligned} y_1(n+1) = {} & \bar{y}_1(n)\,\bar{y}_2(n) q_1[\alpha(n)a_4 + x(n)] \\ & + \bar{y}_1(n)\, y_2(n) q_1[\alpha(n)a_3 + x(n)] \\ & + y_1(n)\, y_2(n) q_1[\alpha(n)a_2 + x(n)] \\ & + y_1(n)\, y_2(n) q_1[\alpha(n)a_1 + x(n)] \end{aligned} \quad (9)$$

$$\begin{aligned} y_2(n+1) = {} & \bar{y}_1(n)\,\bar{y}_2(n) q_2[\alpha(n)a_4 + x(n)] \\ & + y_1(n)\, y_2(n) q_2[\alpha(n)a_3 + x(n)] \\ & + y_1(n)\,\bar{y}_2(n) q_2[\alpha(n)a_2 + x(n)] \\ & + y_1(n)\, y_2(n) q_2[\alpha(n)a_1 + x(n)] \end{aligned} \quad (10)$$

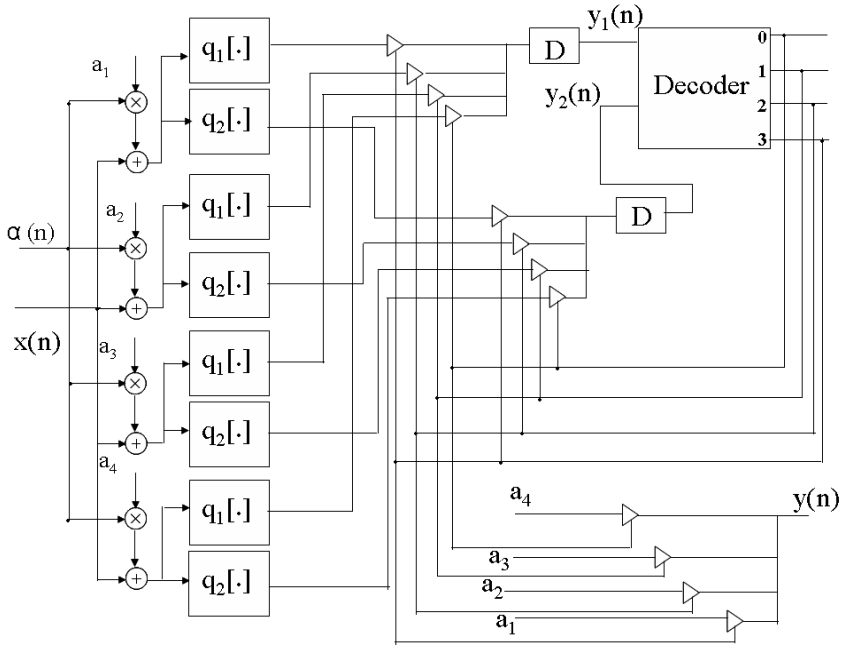

Figure 4: Equivalent transformed structure for first order four level quantizer loop.

The reformulated system is implemented and shown in fig.4.This reformulated system can operate at a very high speed.

Second order two level Quantizer loop

Consider the second order loop containing two level quantizer shown in fig.5

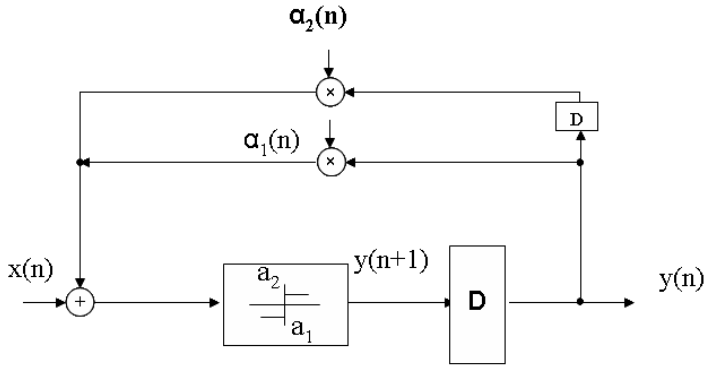

Figure 5: Second order two level quantizer loop.

The loop update operation can be described by the equation

$$y(n+1) = Q[\alpha_1(n)y(n) + \alpha_2(n)y(n-1) + x(n)] \tag{11}$$

Define the binary state $y_1(n)$ as,

$$y_1(n) = 0 \text{ , if } y(n) = a_1$$
$$= 1 \text{ , if } y(n) = a_2 \tag{12}$$

Using the binary state, we can write

$$y(n) = a_1 \bar{y}_1(n) + a_2 y_1(n) \tag{13}$$

Define another binary q-function as

$$q_1(a) = 0 \text{ , if } Q[a] = a_1$$
$$= 1 \text{ , if } Q[a] = a_2 \tag{14}$$

The update operation of the binary state $y_1(n)$ in the reformulated system is described by

$$y_1(n+1) = \bar{y}_1(n)\,\bar{y}_1(n-1)q_1[\alpha_1(n)a_2 + \alpha_2(n)a_2 + x(n)]$$

$$+ \bar{y}_1(n)\,y_1(n-1)q_1[\alpha_1(n)a_1 + \alpha_2(n)a_2 + x(n)]$$

$$+ y_1(n)\,\bar{y}_1(n-1)q_1[\alpha_1(n)a_2 + \alpha_2(n)a_1 + x(n)]$$

$$+ y_1(n)\,y_1(n-1)q_1[\alpha_1(n)a_1 + \alpha_2(n)a_1 + x(n)] \quad (15)$$

The reformulated system is implemented in fig.6.

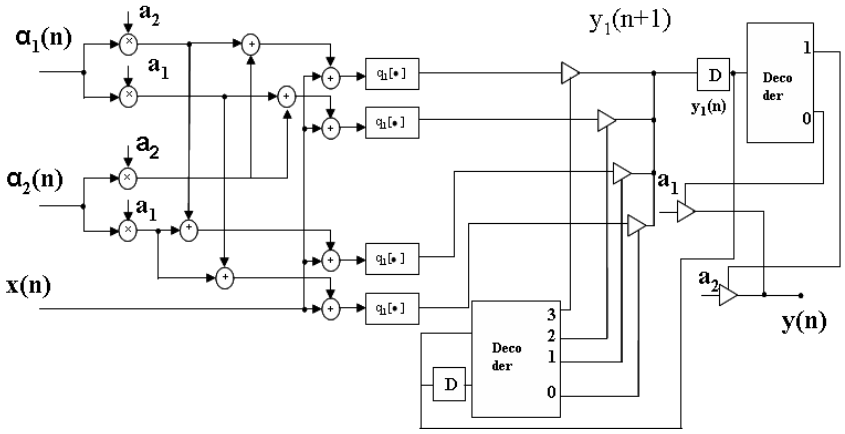

Figure 6: Equivalent transformed structure for second order two level quantizer loop.

The performance of this structure is improved by the use of tristate buffers and decoders. The power consumption of the proposed system is less by the use of buffers in the implementation. The propagation delay of the equivalent transformed structure is greatly reduced by the use of decoder s and buffers in the implementation.

Transformation of DPCM Algorithms

In this section, we obtain the transformed structures for differential pulse code modulation (DPCM) algorithms [2], [3]. The loop update equation in DPCM algorithm is much more complicated than those in section 2.The transformed structures obtained for DPCM loops contain multiplication operation. Look-ahead computation technique can be applied to the transformed structures to pipeline these operations at finer levels.

First order two level Quantizer DPCM Algorithm

Consider the first order prediction-based DPCM algorithm containing a two level quantizer loop described by

$$y(n+1) = Q[\tau(n) + x(n)] \quad (16)$$

$$\tau(n) = \alpha[\tau(n-1) + y(n)] \quad (17)$$

and it is shown in fig.7.

The quantized output y(n) can be either a_2 or a_1. The system state y(n) is encoded using the binary state $y_1(n)$.

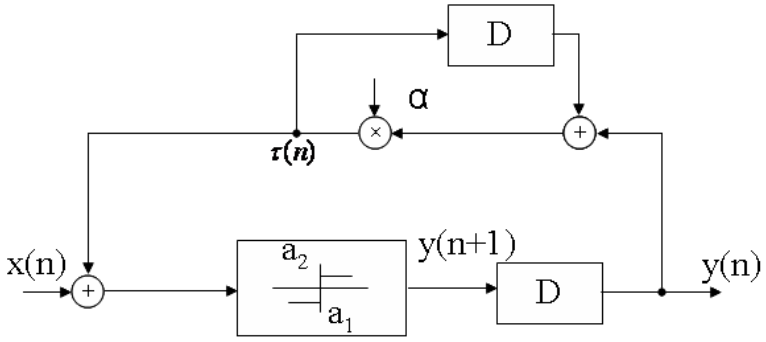

Figure 7: DPCM loop containing two level quantizer and a first order predictor.

The reformulated system using the binary state $y_1(n)$ can be obtained to be

$$\tau(n) = \alpha\tau(n-1) + \alpha a_1 \bar{y}_1(n) + \alpha a_2 y_1(n) \tag{18}$$

$$y_1(n+1) = y(n)q_1[\alpha\tau(n-1) + \alpha a_2 + x(n)]$$

$$+ \bar{y}_1(n)q_1[\alpha\tau(n-1) + \alpha a_1 + x(n)] \tag{19}$$

The clock speed for this system is given by $(T_m + T_a + T_q + T_d)$,where T_d represents the time required for the decoder operation. The clock speed can actually be improved using the technique of retiming, which involves redistribution of the latches in the system [5],[6],[7]. The reformulated figure is shown in fig.8

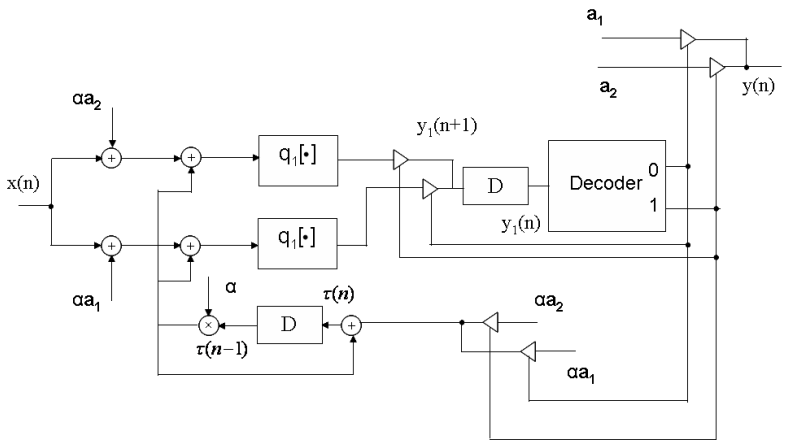

Figure 8: Equivalent transformed DPCM loop containing two level quantizer and a first order predictor.

For some applications, the throughput of this system may be inadequate. To meet the real-time requirements of such systems, we need to create additional concurrency and exploit this concurrency to pipeline the multiplier at finer levels.

First order four level Quantizer DPCM Algorithm

Consider the first order predictor based DPCM loop containing a four level quantizer shown in fig 9. The state update operation τ(n) is obtained as

$$\tau(n) = \alpha\tau(n-1) + \alpha y(n)$$

$$= \alpha\tau(n-1) \; + \alpha a_4 \, \bar{y}_1(n)\, \bar{y}_2(n) \; + \alpha a_3 \, \bar{y}_1(n)\, y_2(n)$$

$$+ \alpha a_2 \, y_1(n)\, \bar{y}_2(n) \; + \alpha a_1 \, y_1(n)\, y_2(n) \qquad (20)$$

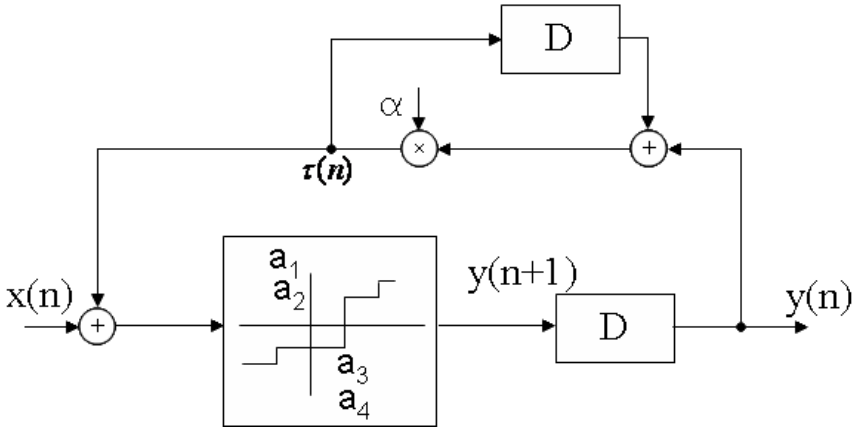

Figure 9: DPCM loop containing four level quantizer and a first order predictor.

Here we encode the state y(n) using two binary states $y_1(n)$ and $y_2(n)$ as

$$y_1(n+1) = y_1(n)\, y_2(n) \;\; q_1[\alpha\tau(n-1) + \alpha a_1 + x(n)]$$

$$+ \; y_1(n)\, \bar{y}_2(n) \;\; q_1[\alpha\tau(n-1) + \alpha a_2 + x(n)]$$

$$+ \; \bar{y}_1(n)\, y_2(n) \;\; q_1[\alpha\tau(n-1) + \alpha a_3 + x(n)]$$

$$+ \; \bar{y}_1(n)\, \bar{y}_2(n) \;\; q_1[\alpha\tau(n-1) + \alpha a_4 + x(n)]$$

$$y_2(n+1) = y_1(n)\, y_2(n) \;\; q_2[\alpha\tau(n-1) + \alpha a_1 + x(n)]$$

$$+ \; y_1(n)\, \bar{y}_2(n) \;\; q_2[\alpha\tau(n-1) + \alpha a_2 + x(n)]$$

$$+ \; \bar{y}_1(n)\, y_2(n) \;\; q_2[\alpha\tau(n-1) + \alpha a_3 + x(n)]$$

$$+ \bar{y}_1(n)\,\bar{y}_2(n)\;\; q_2[\alpha\tau(n-1)+\alpha a_4+x(n)] \tag{21}$$

The reformulated system is shown in Fig. 10 which operates at very high speed.

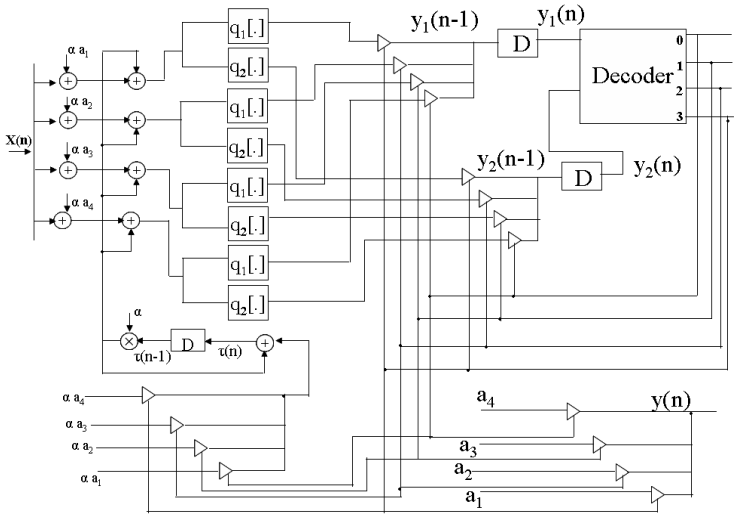

Figure 10: Equivalent transformed DPCM loop containing four level quantizer and a first order predictor.

The achievable sample rate of the system may still be inadequate for some applications. To meet the speed requirements in such systems, we can create concurrency by using look-ahead computation in these systems.

Second order two level Quantizer DPCM Algorithm

Consider the second order predictor based DPCM loop containing a two level quantizer shown in fig 11.

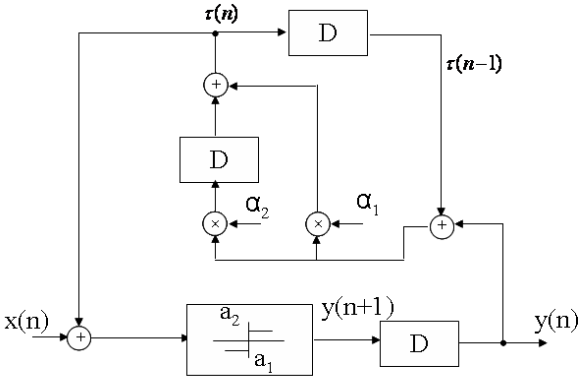

Figure 11: DPCM loop containing two level quantizer and a second order predictor.

The state update operations τ(n) and y (n) are given by

$$\tau(n) = \alpha_1 \tau(n-1) + \alpha_1 y(n)$$
$$+\alpha_2 \tau(n-2) + \alpha_2 y(n-1) \quad (22)$$

$$y(n+1) = Q[\tau(n) + x(n)]$$
$$= Q[\alpha_1 \tau(n-1) + \alpha_2 \tau(n-2)$$
$$+\alpha_1 y(n) + \alpha_2 y(n-1) + x(n)] \quad (23)$$

The state y (n) can be encoded using a single binary state $y_1(n)$.The above state equations can be reformulated in terms of binary state as

$$\tau(n) = \alpha_1 \tau(n-1) + \alpha_2 \tau(n-2) + \alpha_1 a_2 y_1(n)$$
$$+\alpha_1 a_1 \overline{y_1}(n) + \alpha_2 a_2 y_1(n-1) + \alpha_2 a_1 \overline{y_1}(n-1) \quad (24)$$

$$y_1(n+1) = y_1(n) y_1(n-1)\ q_1[\alpha_1 \tau(n-1) + \alpha_2 \tau(n-2)$$
$$+\alpha_1 a_2 + \alpha_2 a_2 + x(n)]$$
$$+ y_1(n)\, \bar{y}_1(n-1)\ q_1[\alpha_1 \tau(n-1) + \alpha_2 \tau(n-2)$$
$$+\alpha_1 a_2 + \alpha_2 a_1 + x(n)]$$
$$+ \bar{y}_1(n) y_1(n-1)\ q_1[\alpha_1 \tau(n-1) + \alpha_2 \tau(n-2)$$
$$+\alpha_1 a_1 + \alpha_2 a_2 + x(n)]$$
$$+ \bar{y}_1(n)\, \bar{y}_1(n-1)\ q_1[\alpha_1 \tau(n-1) + \alpha_2 \tau(n-2)$$
$$+\alpha_1 a_1 + \alpha_2 a_1 + x(n)] \quad (25)$$

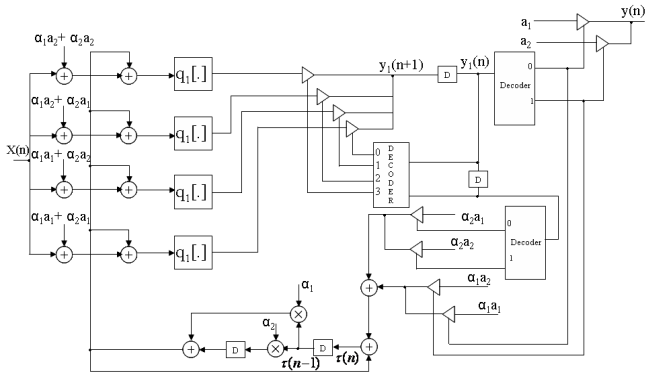

Figure 12: Equivalent transformed DPCM loop containing two level quantizer and a second order predictor.

The reformulated structure is implemented as shown in fig.12.The speed of the system may still be inadequate for some applications. To meet the speed requirements in such systems, we can exploit look-ahead computation technique in these systems.

Conclusions

This paper presents different transformation techniques to pipeline the recursive loop algorithms using decoders and three state buffers. These approaches are suitable for real time high-speed implementation. Note that the proposed architectures improve the performance of the algorithm in terms of propagation delay and power consumption, even though it suffers from large amount of hardware circuitry.

References

[1] K.K. Parhi, "Pipelining in algorithms with quantizer loops," IEEE Trans. Circuits Syst., vol 37, no.7, pp.745-754,jul.1991.
[2] M.L. Honig and D.G. Messerschmitt, Adaptive Filters: Structures, Algorithms, and Applications. Boston: Kluwer, 1984.
[3] N.S. Jayant and P. Noll, Digital coding of waveforms. Englewood Cliffs, NJ:Prentice-Hall, 1984.
[4] M. Renfors and Y. Neuvo, "The maximum sampling rate of digital filters under hardware speed constraints," IEEE Trans. Circuits Syst., vol. CAS-28,pp.196-202, Mar.1981.
[5] S.Y. Kung, "On supercomputing with systolic/wavefront array processors," Proc. IEEE, vol.72, July 1984.
[6] C.E. Leiserson, F. Rose, and S. Saxe, "Optimizing synchronous circuitry by retiming," in proc. Third Caltech Conf. VLSI, Mar.1983.
[7] K.K. Parhi, "Pipeline interleaving and parallelism in recursive digital filters, part 1:Pipelining using scattered look-ahead and decomposition," IEEE Trans. Acoust., Speech, Signal Processing, pp.1099-1117, July 1989.
[8] J. G. Proakis, Digital Communications, 4th edition. New York: McGraw-Hill, 2001.
[9] K. K. Parhi and D. G. Messerschmitt, "Concurrent cellular VLSI adaptive filter architectures," IEEE Trans. Circ. Syst., vol. CAS-34, pp.1141-1151, Oct. 1987.
[10] A. Gatherer and T. H.-Y. Meng, "High sampling rate adaptive decision feedback equalizer," lEEE Trans. Signal Processing**,** vol. 41. pp. 1000-1005, Feb. 1993.
[11] K.K. Parhi, "Design of Multigigabit Multiplexer–Loop–Based Decision Feedback Equalizers", IEEE Trans. VLSI Systems, vol.13, no.4, April.2005.