INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO-IEC/JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION

ISO-IEC/JTC1/SC29/WG11
MPEG92/
November 1992

CCITT
STUDY GROUP XV WP XV/1
Experts Group for ATM Video Coding
Ipswich October 1992

Document # AVC-392

English Version

SOURCE: A. Reibman, K. Matthews, J. Mailhot, R. Aravind: AT&T Bell Laboratories
TITLE: Correction: Eliminating the limit cycle in leaky prediction
PURPOSE: Informational

1 Introduction

The document AVC-349 (MPEG92/493) incorrectly described solution 2 to the limit cycle problem with leaky prediction (section 2.2). This document corrects the description, and should replace the previous section 2. We also include syntax recommendations for MPEG.

As described in section 2 of AVC-349, the solution 2 to the limit cycle is equivalent to adding a signal that varies between 0 and $(2^n - 1)/2^n$ after multiplication but before truncation. The simplified block diagram in Figure 1 illustrates this implementation. This block diagram can be used for any type of multiplication: floating-point, or shift-and-subtract.

Mathematically, the difference signal at the encoder is

$$y_t = x_t - T_n[\alpha x_{t-1} + b_t/2^n],$$

and the reconstructed signal at the decoder (less 128) is

$$w_t = y_t + T_n[\alpha w_{t-1} + b_t/2^n].$$

where b_t is a psuedo-random auxiliary signal that varies from 0 to $2^n - 1$, as in AVC-349. The signal x_t the original input signal minus 128.

Because the leak factors have been chosen to be of the form $(1 - 1/2^n)$, the multiplication is simply performed in hardware using shift-and-subtract. Figure 2 shows an alternate implementation of the limit cycle fix (solution 2) using shift-and-subtract multiplication.

For example, suppose the input value is constant at 40, so $x_t = x_{t-1} = 128 - 40 = -88$. Then, when $0 \le b_t < 8$, $y_t = -6$, while when $8 \le b_t < 16$, $y_t = -5$, if $\alpha = 15/16$. Without adding in the signal b_t, the prediction error $y_t = -6$ always (for truncation with either type of multiplication).

The signal b_t must be the same at the encoder and the decoder to ensure there are no limit cycles. Therefore, it should be sent in the picture layer immediately following the leak factor. This requires an additional 6 bits per picture since the maximum $n = 6$. Because b_t is constant throughout the picture, a DC input gets transmitted as successive DC values, with no additional high frequency components. The subscript t is a temporal index.

The choice of the signal b_t remains a subject of encoder design. However, a signal that works well is a ramp signal with the bits reversed (the most significant bit becomes the least significant bit, etc.).

2 Recommendations

Because the limit cycle problem appears with both floating-point multiplication and shift-and-subtract multiplication, we suggest that the shift-and-subtract multiplication be adopted for ease of implementation.

To eliminate the limit cycle problem with leaky prediction, the following pseudo-code can be used once per picture to generate the value of the auxiliary signal b_t.

```
ramp++;
ramp %= 64;
bt=0;
i=ramp;
for (k=0; k<n; k++) {
  bt = (bt<<1) + (i&0x01);
  i = i>>1;
}
```

The value of the variable ramp is initialized once to zero at the beginning of the encoding process. The variables bt and i are unsigned characters and k and ramp are integers. The variable bt contains the value of the signal b_t. Note that while ramp varies from 0 to 63, bt will vary from 0 to $2^n - 1$.

For ease of hardware implementation, shift-and-subtract multiplication should be used. The following pseudo-code performs the leaky prediction multiplication:

```
val = blk(i,j) -128;
val = val - ((val+bt) >> n);
blk(i,j) = val+128;
```

Before this operation, blk(i,j) contains the motion-compensated prediction pixel at location i,j in the block. After this operation, it contains the value with leaky prediction.

The leak factor and value of the signal b_t should be transmitted in the picture layer for every P picture. One appropriate location is after the **chroma_postprocessing_type** bit in the picture layer, with the syntax shown below.

if (picture_coding_type == 2) {		
leak_factor	3	**uimsbf**
leak_signal_b	6	**uimsbf**
}		

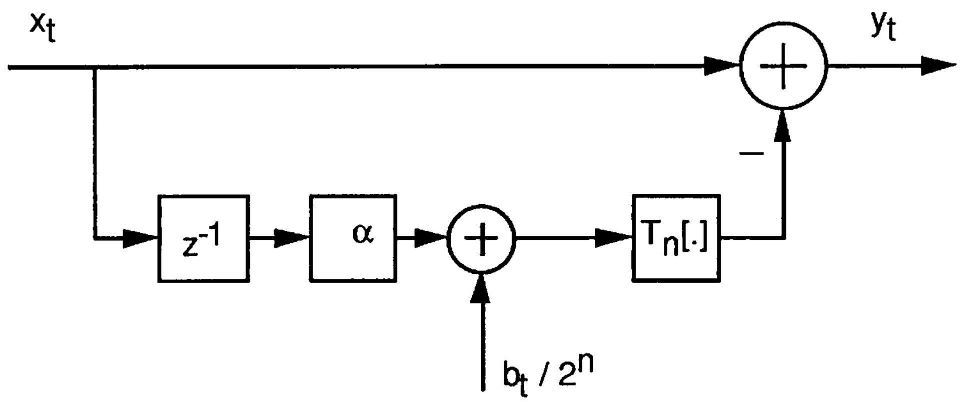

Figure 1(a). Encoder without limit cycles, any multiplier

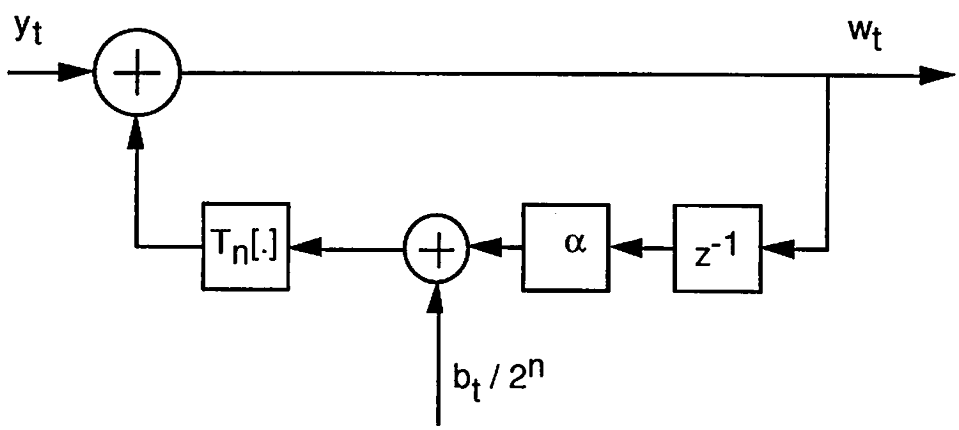

Figure 1(b). Decoder without limit cycles, any multiplier

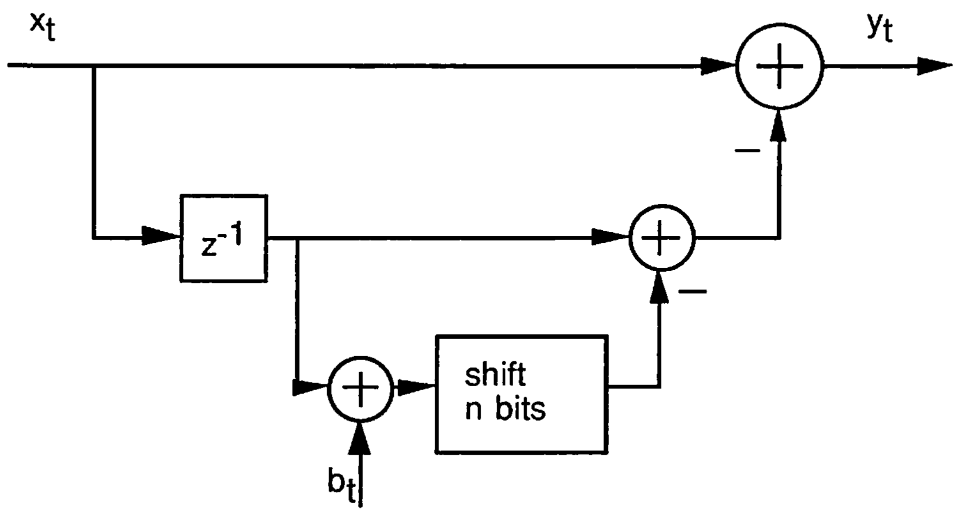

Figure 2(a). Encoder without limit cycles, using shift-and-subtract multiplier

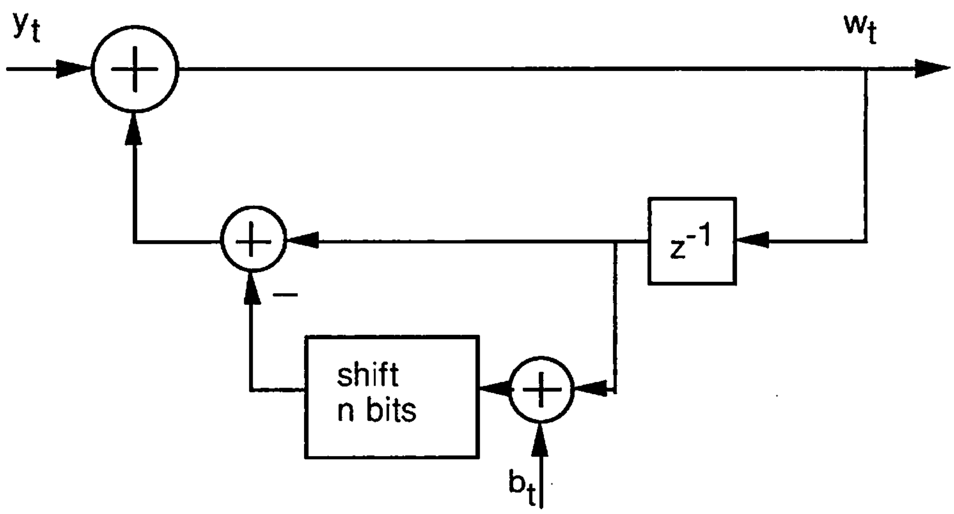

Figure 2(b). Decoder without limit cycles, using shift-and-subtract multiplier