IN-61-CR
12
p 25

JPL Publication 91-1

Algorithms for a Very High Speed Universal Noiseless Coding Module

Robert F. Rice
Jet Propulsion Laboratory

Pen-Shu Yeh
Warner Miller
Goddard Space Flight Center

February 15, 1991

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 91-1

Algorithms for a Very High Speed Universal Noiseless Coding Module

Robert F. Rice
Jet Propulsion Laboratory

Pen-Shu Yeh
Warner Miller
Goddard Space Flight Center

February 15, 1991

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and the Goddard Space Flight Center, under a contract with the National Aeronautics and Space Administration.

ABSTRACT

This publication provides the algorithmic definitions and performance characterizations for a high performance adaptive "coding module" currently under development in custom VLSI at two NASA centers. Operation of at least one of these (single chip) implementations is expected to exceed 500 Mbits/s under laboratory conditions. Operation of a companion "decoding module" should operate at up to half the coder's rate. The functionality provided by these modules should be applicable to most of NASA's science data.

The module incorporates a powerful adaptive noiseless coder for "Standard Form" Data Sources (i.e., sources whose symbols can be represented by uncorrelated non-negative integers where the smaller integers are more likely than the larger ones). Performance close to data entropies can be expected over a "Dynamic Range" of from 1.5 to 12-14 bits/sample (depending on the implementation).

This is accomplished by adaptively choosing the best of many "Huffman Equivalent" codes to use on each block of 16 samples. Because of the extreme simplicity of these codes, no table lookups are actually required in an implementation, thus leading to the expected very high data rate capabilities already noted. The "coding module" can be used directly on data which has been "pre-processed" to exhibit the characteristics of a Standard Form Source. Alternatively, a built-in Predictive Pre-processor can be used where applicable. This built-in Pre-processor includes the familiar one-dimensional predictor followed by a function which maps the prediction error sequences into the desired standard form. Additionally, an External Prediction can be substituted if desired (e.g., two-dimensional applications), further extending the module's generality.

CONTENTS

Table

Figures

ALGORITHMS FOR A VERY HIGH SPEED UNIVERSAL NOISELESS CODING MODULE

INTRODUCTION

References 1–4 provide the development and analysis of some practical adaptive techniques for efficient noiseless (lossless) coding of a broad class of data sources. These have been applied, in various forms, to numerous applications.

Those functions and algorithms most desirable for incorporation in a "coding module" which could be implemented using current custom VLSI capabilities were presented at the first NASA Data Compression Workshop at Snowbird, Utah, in 1988[5]. A workshop committee recommended that NASA should proceed and implement this "coding module." Since then, both the Jet Propulsion Laboratory (JPL) and the University of Idaho (in conjunction with the Goddard Space Flight Center) have implemented multiple custom VLSI versions of this coder in CMOS. Operation of at least one of these (single chip) coding modules is expected to exceed 500 Mbits/s under laboratory conditions. This far exceeds performance expectations anticipated in 1988. A companion single chip "decoding module" developed by the University of Idaho is expected to run at up to half the maximum rate of the coding module.

It is anticipated that the high performance functionality of these modules can serve most of NASA's science data where a lossless representation is appropriate.

The intent of this publication is to provide a concise description of the algorithmic and performance characteristics that are embodied in the coding modules of these VLSI implementations. A more general development, including extensive application notes can be found in Ref. 6. Details on these implementations are provided in Refs. 7–9.

THE CODING MODULE

A functional block diagram of a general purpose lossless coding "module" is shown in Fig. 1. A variation in Rice's original notation (of subscripting the Greek letter ψ) name various coding operations. Subsequent sections will quickly converge to more specific definitions that relate to the VLSI modules being implemented.

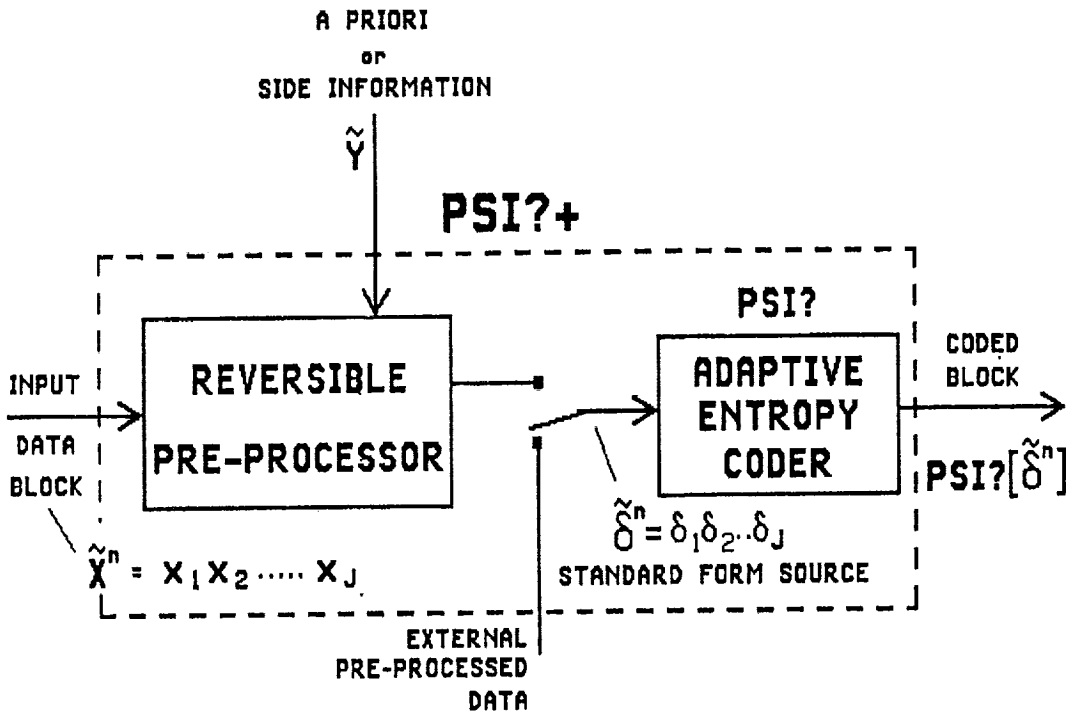

Fig. 1. General Purpose Noiseless Coding Module Block Diagram

The input to this coding module

$$\tilde{X}^n = x_1\, x_2 \ldots x_J \tag{1}$$

is a J sample block of n bit samples. $\tilde{Y}$ is a priori or Side Information that might help in the coding process.

The **overall** unspecific process of representing $\tilde{X}^n$ is named PSI?+ so that the actual coded result is

$$\text{PSI?} + [\tilde{X}^n, \tilde{Y}]$$

As Fig. 1 shows, the coding process is split into two independent steps as discussed below.

STEP 1

A Reversible Pre-processor is a process designed to convert the source represented by $\tilde{X}^n$ sequences (and $\tilde{Y}$) into a close approximation to a STANDARD FORM Data Source, represented by $\tilde{\delta}^n$ sequences. This process usually includes a de-correlation procedure (prediction).

The pre-processor converts each $\tilde{X}^n$ (and corresponding $\tilde{Y}$, if any) into

$$\tilde{\delta}^{n'} = \delta_1 \, \delta_2 \ldots \delta_J \tag{2}$$

a $J \geq 1$ sample sequence of n' bit samples. Usually $n = n'$, and we will henceforth assume that here.

Standard Form Source. Specifically,

a) Samples of $\tilde{\delta}^n$ are the non-negative integers 0, 1, 2, ..., q (3)

b) Samples of $\tilde{\delta}^n$ are independent (4)

c) With $p_i = \Pr[\delta_j = i]$, the probabilities are ordered so that the smaller integers occur more frequently, i.e.,

$$p_0 \geq p_1 \geq p_2 > \ldots \tag{5}$$

d) The source entropy is given as

$$\bar{H}_\delta = -\sum_i p_i \log_2 p_i \quad \text{bits/sample} \tag{6}$$

The best pre-processor will meet these conditions and produce the lowest $\bar{H}_\delta$.

STEP 2

An adaptive entropy coder, named PSI? for now, efficiently represents (Standard Source) pre-processed $\tilde{\delta}^n$ sequences with the coded result

$$\text{PSI?}\,[\tilde{\delta}^n] = \text{PSI?} + [\tilde{X}, \tilde{Y}]$$

This entropy coder is independent of the pre-processor. Its goal is to achieve performance that remains close to $\bar{H}_\delta$ as it varies with time.

Note that the coding module in Fig. 1 allows for coder PSI? to be used directly on externally supplied pre-processed data.

Prelude to the Details

A general form of an Adaptive Entropy coder (designed to efficiently represent Standard Data Sources), which chooses from multiple algorithm options on a block-by-block basis, will be identified in the next section. Specific sets of such code options will be defined and incorporated in this structure as a **parametrically** defined adaptive coder. In doing so, the unspecific "PSI?" will be turned into a specific coder called "PSIss."

Finally, the specific parameters of PSIss that are used in current VLSI implementations will be identified.

ADAPTIVE ENTROPY CODER FOR THE STANDARD SOURCE

PSI? of Fig. 1 represents the general purpose adaptive coder called PSI11 in Refs. 2-4. Basically, such a coder chooses one of a set of Code Options (coding algo-

rithms) to use to represent an incoming block of "**pre-processed**" data samples. A unique binary identifier precedes a coded block to tell a decoder which decoding algorithm to use. The following discussion will identify specific code options.

Code Options

Backup. When no coding of any form is performed on the data, we call this PSIbu

$$\text{PSIbu}\,[\tilde{\delta}^n] = \tilde{\delta}^n \tag{7}$$

This representation is used in an adaptive coder when all other available code options fail to compress $\tilde{\delta}^n$.

The Fundamental Sequence Code. Recall that the pre-processed samples of $\tilde{\delta}^n$ are the non-negative integers $i \geq 0$. A variable length "Fundamental Sequence Code, fs", is defined for each i as follows

$$\text{fs}[i] = \underbrace{0\,0\,0\,.\,.\,.\,.\,0\,0\,0}_{i \quad \text{zeroes}}\,1 \quad \text{for } i \geq 0 \tag{8}$$

That is, simply append a 1 to the end of a sequence of i zeroes.

The "Fundamental Sequence" itself is the application of fs[.] to all the samples of $\tilde{\delta}^n$. Following Rice's notation,[1]

$$\text{PSI1}[\tilde{\delta}^n] = \text{fs}[\delta_1] * \text{fs}[\delta_2] * \ldots . \text{fs}[\delta_J] \tag{9}$$

is the Fundamental Sequence. This defines Code Option, PSI1.

[1]An asterisk, *, is used to emphasize the concatenation of sequences.

Split-Sample Modes. The code option definitions here are basically to "split" off the k least significant bits of each $\tilde{\delta}^n$ sample and send them separately. The remaining n-k most significant bit samples are then coded using PSI1. Specifically, with

$$\delta^n = \delta_1\, \delta_2 \ldots \delta_J$$

Let

$$\tilde{M}^{n,k} = m_1\, m_2 \ldots . m_J \tag{10}$$

be the sequence of all the n-k most significant bit samples of δ^n and let

$$\tilde{L}_k = \ell sb_1 * \ell sb_2 * \ldots . \ell sb_J \tag{11}$$

denote the corresponding sequence of all the k-bit least significant bit samples of $\tilde{\delta}^n$.

That is

$$\delta_i = m_i * \ell sb_i \tag{12}$$

The "Split-Sample" Mode Code Option PSI1,k is defined by

$$PSI1,k[\tilde{\delta}^n] = PSI1[\tilde{M}^{n,k}] * \tilde{L}_k \tag{13}$$

Note that k = 0 is a special case where

$$PSI1,0 = PSI1$$

and when k = n

$$PSI1n, = PSIbu$$

The Individual Code Words. Note that **the individual code word assigned to** δ_j in (12) when code option PSI1,k is applied is given as

$$fs[m_j] * \ell sb_j \tag{14}$$

That is, the Fundamental Sequence Code, fs[·] in (9), is applied to the most significant n-k bits of δ_j, followed by the least significant k bits of δ_j. From this description, it should be easier to see that the **only variable length code operation ever required is the application of fs[·]**, since the "ℓsbs" can simply be shifted out, and fs[·] can be implemented **without any table lookups**.

Performance of the Individual PSI1,k Options. Under certain familiar assumptions on the type of data source (these assumptions will be described in a later section),

the individual "variable length codes" represented by (14) can be shown to be equivalent to Huffman Codes.[10] (15)

Thus they are not only extremely simple, they are **optimum** too.

But even more important for their application in an adaptive coder, the entropy where PSI1,k achieves its best performance is at

$$\bar{H}_{\delta}^{k} \approx k + 2 \text{ bits/sample} \tag{16}$$

and performance remains close to $\bar{H}_{\delta}^{k}$ over a range of ±0.5 bit/sample. Thus there is at least one PSI1,k option that should provide efficient coding for any

$$\bar{H}_{\delta} > 1.5 \text{ bits/sample} \tag{17}$$

Such conclusions can also be drawn directly from simulations using a broad range of data sources.

Split-Sample Adaptive Coder, PSIss

We can now use these Split-Sample Modes described above to replace the general coder, PSI? in Fig. 1 with a specific class of parametrically defined adaptive coders, named PSIss. PSIss is based on the following parameters:

$$\begin{aligned} &J = \text{block size} \geq 1 \\ &n = \text{Input Bits/Sample} \\ &N = \text{No. of Code Options} \\ &\lambda \geq 1 \text{ (Dynamic Range Parameter)} \end{aligned} \tag{18}$$

A functional block diagram is shown in Fig. 2.

The representation of J sample $\tilde{\delta}^n$, using an N option PSIss with parameter $\lambda \geq 1$ is given by

$$PSIss[\tilde{\delta}^n] = ID(id) * PSI1, k(id)[\tilde{\delta}^n] \tag{19}$$

where

$$id = 0, 1, 2, \ldots N\text{-}1 \tag{20}$$

is the integer value of a coder identifier for the options used, and ID(id) is its standard binary representation, requiring[2]

$$\lceil \log_2 N \rceil \text{ bits} \tag{21}$$

---

[2] $\lceil z \rceil$ is the smallest integer, greater than or equal to z.

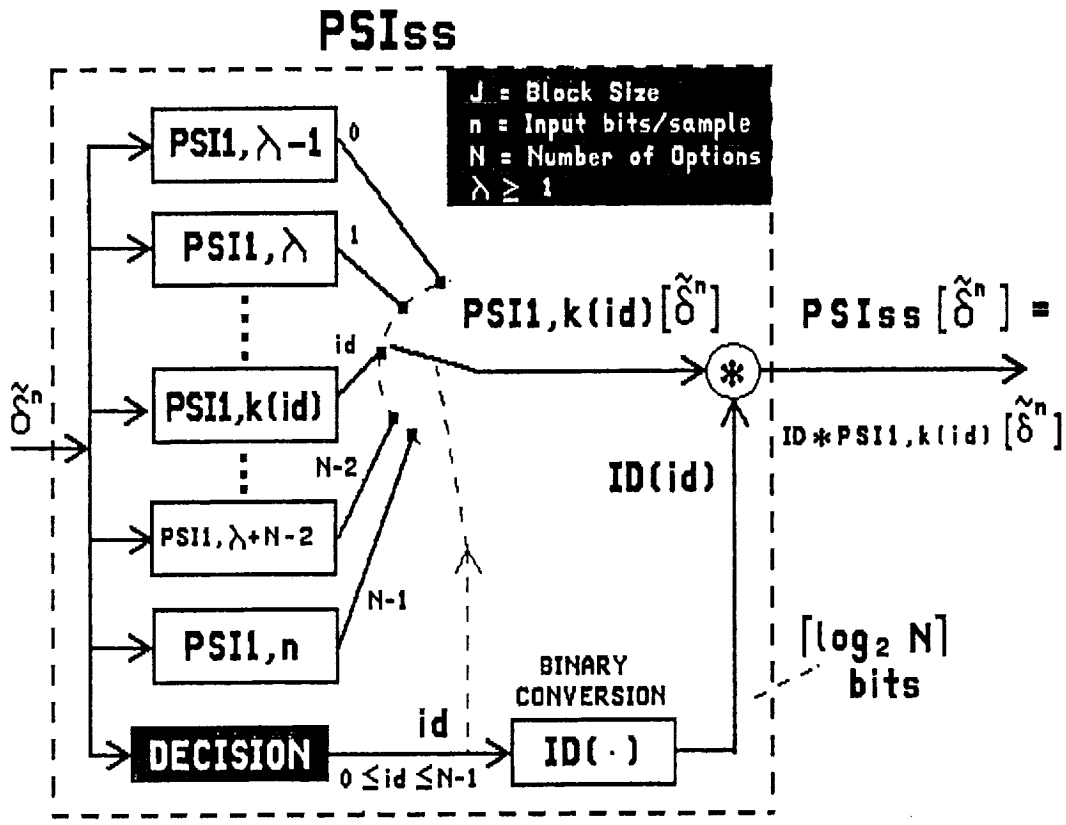

Fig. 2. Parametric Split-Sample Adaptive Coder Functional Block Diagram

PSI1,k(id) is the Split-Sample option specified by id, where

$$k(id) = \begin{cases} n \text{ for } id = N\text{-}1 \\ \lambda - 1 + id \quad \text{Otherwise} \end{cases} \tag{22}$$

for parameter $\lambda \geq 1$.

Dynamic Range. Except for limiting cases, the range of entropies where PSIss can be expected to efficiently represent pre-processed $\tilde{\delta}^n$ sequences has been shown to be closely specified by

$$\lambda + 0.5 \leq \bar{H}_\delta \leq \min \begin{cases} n \\ \lambda + N - 0.5 \end{cases} \tag{23}$$

A close look at this expression shows that **each increase in λ moves the Dynamic Range of efficient performance upwards by 1 bit/sample.**

Performance Graph. A graph of typical performance for PSIss with N = 12, λ = 1, n = 14 and J = 16 is shown in Fig. 3.

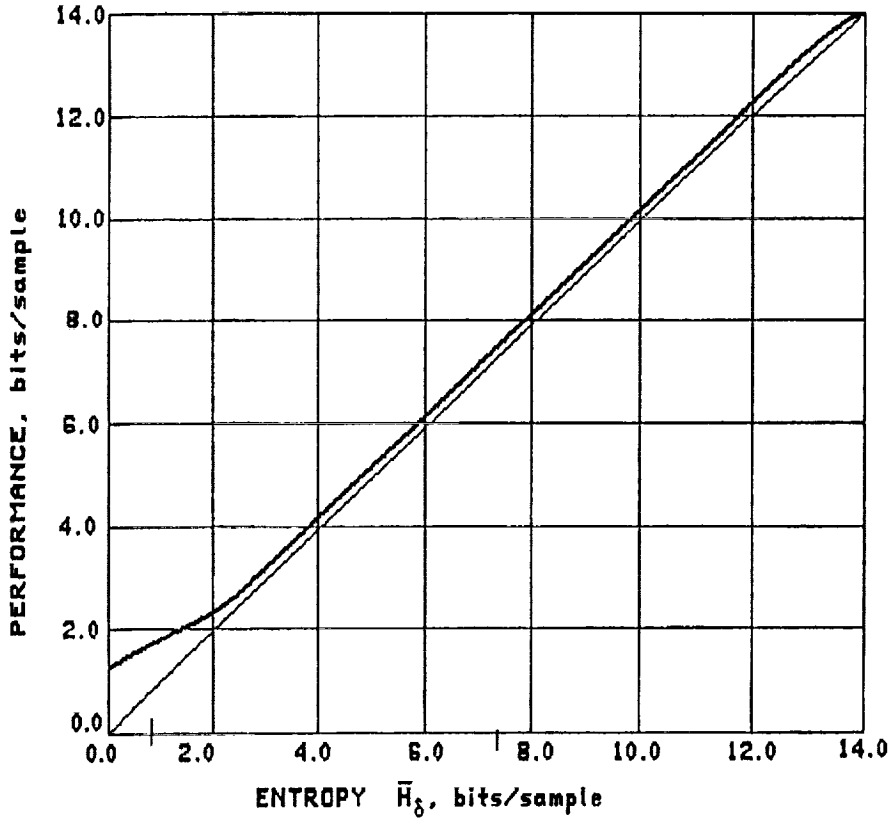

Fig. 3. PSIss Average Performance for N = 12, n = 14, λ = 1, J = 16.

Choosing the Right Option. The optimum criteria for selecting the best option to use to represent $\tilde{\delta}^n$ is to simply choose the one that produces the shortest coded sequence. That is,

choose $k = k^*$ if [3]

$$\mathscr{L}(\mathrm{PSI1},k^* [\tilde{\delta}^n]) = \min_k \{\mathscr{L}(\mathrm{PSI1},k [\tilde{\delta}^n])\} \tag{24}$$

Now, letting

$$F_k = \mathscr{L}(\mathrm{PSI1} [\tilde{M}^{n,k}])$$

we have from (12)

$$\mathscr{L}(\mathrm{PSI1},k [\tilde{\delta}^n]) = F_k + Jk \tag{25}$$

and from (8) - (10)

$$F_k = \sum_{i=1}^{J} m_i + J \tag{26}$$

(i.e., the sum of the most significant n-k bit samples plus the block size).

Thus (25) and (26) can simplify the decision making in (24) without actually coding the data. But this can be further simplified.

By taking advantage of the randomness in the least significant bits, the expected value of F_k can be related to F_0 by[4],[6]

$$E\{ F_k|F_0 \} = 2^{-k} F_0 + \frac{J}{2}(1 - 2^{-k}) \tag{27}$$

which we use as a estimate in (25). We have

[3] $\mathscr{L}(\tilde{Z})$ = length of sequence $\tilde{Z}$ in bits.

$$\gamma_{1,k}(\tilde{\delta}^n) = 2^{-k} F_0 + \frac{J}{2}\left(1 - 2^{-k}\right) + Jk$$
$$\approx \mathscr{L}\left(PSI1,k\,[\tilde{\delta}^n]\right) \tag{28}$$

We can then choose $k = k^*$ if

$$\gamma_{1,k^*}(\tilde{\delta}^n) = \min_k \left\{ \gamma_{1,k}(\tilde{\delta}^n) \right\} \tag{29}$$

and this leads to distinct decision regions based solely on F_0 (which by (26) can be determined by adding up the original samples since $m_j = \delta_j$). The boundaries to adjacent PSI1,k decision regions are given by

$$F_0 = \frac{J}{2} + J\left(2^{k+1}\right) \text{ bits} \tag{30}$$

Any PSI1,k option will generate more bits than PSIbu when

$$F_0 > \frac{J}{2}\left[(n-k)\,2^{k+1} + 1 - 2^k\right] \text{ bits} \tag{31}$$

A sample table of decision regions is shown in Table 1 for an N = 8 option PSIss.

Note that if the largest value of k used is $k = \ell$, then PSI1, ℓ will generate more bits than PSIbu when

$$F_\ell > (n - \ell)\,J \text{ bits} \tag{32}$$

which may be a simpler test than (31) provides in some cases.

Table 1. Decision Regions for an N = 8 Option PSIss

Code Option	F_0 Region in bits
PSI1,0	$F_0 \leq 5J/2$
PSI1,1	$5J/2 < F_0 \leq 9J/2$
PSI1,2	$9J/2 < F_0 \leq 17J/2$
PSI1,3	$17J/2 < F_0 \leq 33J/2$
PSI1,4	$33J/2 < F_0 \leq 65J/2$
PSI1,5	$65J/2 < F_0 \leq 129J/2$
PSI1,6	$129J/2 < F_0 \leq (128n-831)J/2$
PSIbu	$(128n-831)J/2 < F_0$

PSIss Implementation Parameters

The PSIss parameters used in the current VLSI implementations are as follows[8,9]:

a) J = Block size of $\tilde{\delta}^n$ = 15 or 16

b) n = Maximum Quantization of $\tilde{\delta}^n$ samples

= 12 for JPL chips

= 14 for University of Idaho chips

c) N = number of code options

= 11 for JPL chip

= 12 for University of Idaho chip

d) λ = Dynamic Range Parameter = 1 = Starting Option Parameter

THE PRE-PROCESSOR

The entropy coder, PSIss, as described in the previous section, was designed to efficiently represent (pre-processed) Standard Data Sources. PSIss doesn't need to know which pre-processor was used to produce its input. However, for an extremely broad set of real problems, the general pre-processing function of Fig. 1 can be replaced by the more specific Basic Predictive Pre-processor in Fig. 4. It is shown imbedded within the complete coding module for your convenience.

Standard Predictor

The first part of this pre-processor is a very simple predictor consisting of a single sample delay element. With x_i as the i^{th} sample in $\tilde{X}^n$, this delay element "predicts" that x_i equals the previous sample:

$$\hat{x}_i = x_{i-1} \tag{33}$$

The previous sample could be the last sample from a previous block when coding multiple blocks. It is assumed that the sample delay is always initialized with a prediction. But note also at this time that the module's design allows for an External Prediction to be supplied in place of this simple one-dimensional form.

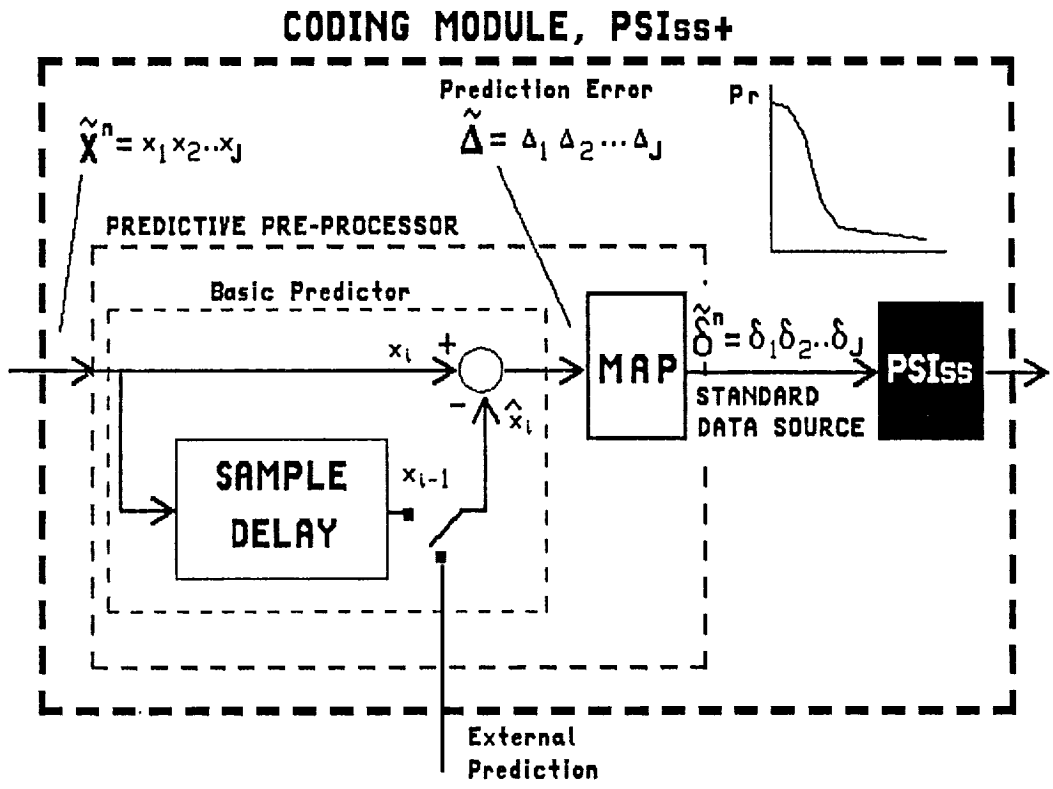

Fig. 4. Basic Predictive Pre-processor within Coding Module, PSIss+

The difference between any sample and its prediction produces the error signal

$$\Delta_i = x_i - \hat{x}_i \tag{34}$$

and the block of J error values

$$\tilde{\Delta} = \Delta_1\ \Delta_2 \ldots . \Delta_J \tag{35}$$

As a new data source, $\tilde{\Delta}$ sequences tend to be uncorrelated and display a unimodal distribution around zero (when data values are not near the boundaries of its dynamic range). That is[4]

$$\Pr[\Delta_i = 0] \geq \Pr[\Delta_i = -1] \geq \Pr[\Delta_i = +1] \geq \Pr[\Delta_i = -2] \geq \ldots . \tag{36}$$

The Mapper (into Standard Form Source)

When the latter condition in (36) is true, the following function will map each Δ_i into a corresponding Standard Source δ_i such that

$$p_0 \geq p_1 \geq p_2 \geq p_3 \geq \ldots .$$

Additionally, it will assure that an n-bit/sample x_i from $\tilde{X}^n$ produces an n-bit/sample δ_i. Further, the desired probability ordering of the δ_i is more closely approximated when x_i values are near 0 or $x_{max} = 2^n - 1$.

The Mapper is defined by:

$$\delta_i = \begin{cases} 2\Delta_i & 0 \leq \Delta_i \leq \theta \\ 2\,|\Delta_i| - 1 & -\theta \leq \Delta_i < 0 \\ \theta + |\Delta_i| & \text{Otherwise} \end{cases} \tag{37}$$

---

[4]An equally valid assumption is

$$\Pr[\Delta_i = 0] \geq \Pr[\Delta_i = +1] \geq \Pr[\Delta_i = -1] \geq \Pr[\Delta_i = +2] \geq \ldots .$$

but we will stick with (36). It appears to offer a slight advantage in the implementation of the mapping function described following (36).

where

$$\theta = \min (\hat{x}_i , x_{max} - \hat{x}_i)$$
$$x_{max} = 2^n - 1 \tag{38}$$

A Further Benefit. Suppose the pre-processor in Fig. 4 is set to receive an external prediction, and **fix** that prediction to

$$\hat{x}_i = 0 \tag{39}$$

Then, tracing through Eqs. (37) and (38), one finds that

$$\delta_i = x_i \tag{40}$$

That is, the input to the pre-processor, $\tilde{X}$, is passed directly through unchanged as $\tilde{\delta}$. This means that the separate desired external input line in Fig. 1 (to allow the pre-processor to be skipped) can be omitted.

The Ideal Case

If one assumes that the distribution of $\tilde{\Delta}$ samples in (36) fits the Laplacian form, then the code equivalence result in (14) can be proven.[10] That is,

the simple PSI1,k codes are equivalent to Huffman Codes for Laplacian distributions of prediction errors. (41)

Reference Sample

Most of those applications which make use of the built-in Predictive Pre-processor will occasionally need to incorporate a "Reference Sample" along with the coded prediction errors (e.g., at the start of an image line). Each of the VLSI

implementations[7–9] incorporates an optional feature to extract such a Reference Sample from an incoming data stream.

STANDARDIZATION OF THE PSIss+ MODULE

References 8 and 9 describe separate VLSI implementations of the coding module PSIss+. Both efforts represent significant achievements. However, as a result of earlier less specific definitions, these implementations have subtle variations in their algorithmic specification. To ensure compatibility in future applications, recent NASA efforts have focused on **standardizing** the definition of a PSIss+ Coding Module. This publication represents the most up-to-date statement of that effort.

REFERENCES

1) R. F. Rice and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," **IEEE Trans. on Communication Technology**, Vol. COM-19, Part I, Dec. 1971, pp. 889-897.

2) R. F. Rice, "Some Practical Universal Noiseless Coding Techniques," **JPL Publication 79-22**, Jet Propulsion Laboratory, Pasadena, California, March 15, 1979.

3) R. F. Rice, "Practical Universal Noiseless Coding," **1979 SPIE Symposium Proceedings**, Vol. 207, San Diego, California, August 1979.

4) R. F. Rice and J. Lee, "Some Practical Universal Noiseless Coding Techniques," Part II, **JPL Publication 83-17**, Jet Propulsion Laboratory, Pasadena, California, March 1983.

5) R. F. Rice, "Universal Noiseless Coding Techniques (Electronic Slideshows)," **NASA Workshop on Data Compression**, Snowbird, Utah, May 1988.

6) R. F. Rice, "Practical Universal Noiseless Coding Techniques," Part III, **JPL Publication 91-3**, Jet Propulsion Laboratory, Pasadena, California (to be published).

7) J. Venbrux and N. Liu, "Lossless Image Compression Chip Set," **Proceedings of Northcon**, Seattle, Washington, Oct. 9-11, 1990, pp. 145-150.

8) J. Venbrux and N. Liu, "A Very High Speed Lossless Compression/ Decompression Chip Set," **Proceedings IEEE Data Compression Conference**, Snowbird, Utah, April 1991; and JPL Publication 91-13, Jet Propulsion Laboratory, Pasadena, California (to be published).

9) J. Lee, et al., "A Very High Speed Noiseless Data Compression Chip for Space Imaging Applications," **Proceedings IEEE Data Compression Conference**, Snowbird, Utah, April 1991; and JPL Publication 91-12, Jet Propulsion Laboratory, Pasadena, California (to be published).

10) Pen-Shu Yeh, et al., "On the Optimality of Code Options for a Universal Noiseless Coder," **JPL Publication 91-2**, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.