ADVANCED TECHNIQUES AND TECHNOLOGY FOR EFFICIENT DATA STORAGE, ACCESS AND TRANSFER

Robert F. Rice
Jet Propulsion Laboratory
Califronia Institute of Technology
Pasadena, CA 91109

Warner Miller
Goddard Space Flight Center
Greenbelt, MD 20771

ABSTRACT

Advanced techniques for efficiently representing most forms of data are being implemented in practical hardware and software form through the joint efforts of three NASA centers. These techniques adapt to local statistical variations to continually provide near optimum code efficiency when representing data without error. Demonstrated in several earlier space applications, these techniques are the basis of initial NASA data compression standards specifications.

Since the techniques clearly apply to most NASA science data, NASA invested in the development of both hardware and software implementations for general use. This investment includes high-speed single-chip VLSI coding and decoding modules as well as machine-transferrable software routines. The hardware chips have been tested in the laboratory at data rates as high as 700 Mbits/s.

A coding module's definition includes a predictive preprocessing stage and a powerful adaptive coding stage. The function of the preprocessor is to optimally process incoming data into a standard form data source that the second stage can handle. The built-in preprocessor of the VLSI coder chips is ideal for high-speed sampled data applications such as imaging and high-quality audio, but additionally, the second stage adaptive coder can be used separately with any source that can be externally preprocessed into the "standard form." This generic functionality assures that the applicability of these techniques and their recent high-speed implementations should be equally broad outside of NASA.

INTRODUCTION

Science data returned from space instruments is often studied in great detail by many investigators. For some investigators the precise value of individual data samples can be crucial. Such "high fidelity criteria" led NASA to the development of efficient "lossless" techniques for representing such data without introducing error.

Increasing data rate requirements of many new instruments and a demonstrated capability to support a broad range of instrument data fostered the recent implementations of an important subset of these techniques as single coding and decoding modules. A somewhat broader set of algorithms is being incorporated into a software package written in C.

The intent of this paper is to introduce the underlying characteristics and algorithms associated with this hardware and software at a tutorial level. Details are provided by References 1-4.

The Standard Source

By various means, many data sources can be converted to one with the basic characteristics in Figure 1. Inactive sources will generate a greater occurrence of small data values than active sources. Conversely, active sources will generate a greater occurrence of larger data values than inactive sources. But in both cases (and all the cases inbetween), smaller data values occur more frequently than large.

Many real data sources can be preprocessed into the form described for Standard Sources.[1] So in subsequent discussions we will presume that this step has been done unless noted otherwise. Eventually, we will return to the preprocessing step.

The Variable Length Code

A codeword is a unique sequence of binary bits used to represent data values from this Standard Source. All the codewords together make up a "code." Basic data systems use a fixed-length code. That is, suppose there were 2^n data values. Then the fixed length code would consist of 2^n codewords, all n bits in length. Thus every data sample would require n bits/sample. By using a variable length we can usually improve on this by taking advantage of the differing frequency of occurrence shown in Figure 1.

A variable length code has different codeword lengths, as shown by the example in Table 1.

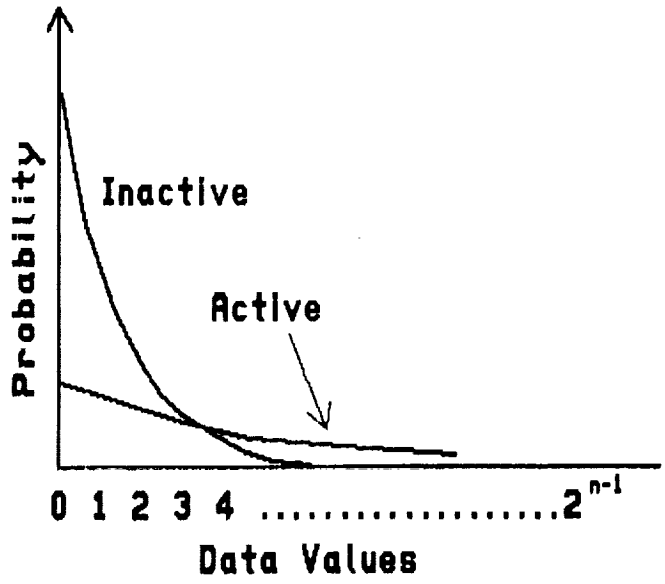

Figure 1. Standard Source Characteristics

[1]This preprocessing should also produce uncorrelated data values.

DATA VALUE	CODEWORD
0	1
1	01
2	001
3	0001
4	00001
•	•
•	•
•	•

Table 1. Simple Code

If the small data values occur very frequently, one could expect the average number of bits used to be less than n bits/sample (because the shorter codewords would be used more frequently). For example, if only the data value 0 occurred, the simple code of Table 1 would use only 1 bit/sample. But the opposite is true, too. The code in Table 1 might use a lot more than n bits/sample when used with a very active data source.

The Huffman Code

D. A. Huffman invented an algorithm for generating the best code for a "known" distribution of data values, as shown in Figure 1 [5]. This suggests a possible solution. Can we just use the right Huffman code? Unfortunately, such a direct application of the famous algorithm can have some practical difficulties:

1. Wrong Code: Data activities tend to vary with time, from instrument to instrument, and may not be known a priori at all.

2. Implementation Problem: Some new instruments have up to 2^{14} data values (e.g., 14 bits/sample fixed length code). Thus a single Huffman code could conceivably require a lookup table containing 2^{14} codewords, some of them quite lengthy.

The Adaptive Coder

We seek an answer to both of these problems with the structure of Figure 2. Consider functionally what this adaptive coder structure does. It operates on short blocks of data, X (e.g., 16 data samples), choosing the best of N coders to use for each block. A separate identifier, ID, precedes the chosen coded block, $C_{ID}[X]$, to tell a "decoder" which decoding algorithm it needs to use. The identifier penalty is small. For example, a coder with $9 \leq N \leq 16$ code options requires a 4-bit identifier (or 0.25 bits/sample for a 16-sample block). Thus with properly chosen code options, such an adaptive coder should be able to handle large variations in data activity.

This is in fact the case, and the implied complexity never materializes. The chosen codes are both equivalent to Huffman codes but require no table lookups at all.

ADAPTIVE VARIABLE-LENGTH CODER CHARACTERISTICS

References 1 and 3 provide several variations to the adaptive variable length coders (AVLC) that begin with the structure in Figure 2 as their starting point. We will focus here on the most useful variations which have recently been implemented as individual VLSI chips. The reader should consult the references for greater detail and a broader perspective.

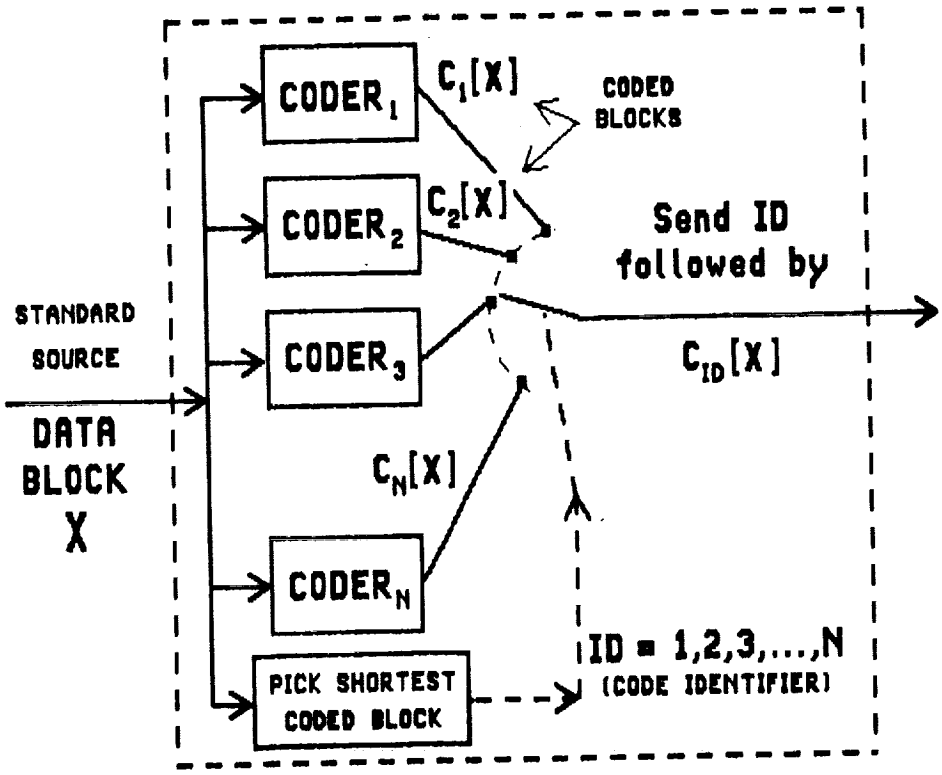

Figure 2. Adaptive Variable Length Coder Concept

Entropy

Information theory provides us with a mathematical measure of coding efficiency. For the case discussed here, entropy is a function of the probability distributions in Figure 1. Basically, entropy increases with activity. More specifically, a Standard Source with a fixed data value distribution and associated entropy, H bits/sample, cannot be coded with fewer bits/sample than H. A coder which codes close to the entropy, such as a Huffman code used on its design distribution, is said to be efficient.

AVLC Options

The simplest variable length code is the one shown in Table 1. It is defined for any number of data values. A codeword for data value j is j zeroes followed by a one. Clearly this code requires no table lookup. Further, it is an efficient coder over the entropy range $1.5 \leq H \leq 2.5$ bits/sample.

Other code options first split an n-bit data sample into its k least significant bits and its n-k most significant bits. Then the simple code of Table 1 is applied to the samples formed by the most significant bits.. Each value of k provides a code option which is efficient over an entropy range of about 1 bit/sample, centered on an entropy of k +2 bits/sample. More amazingly, it has been shown that these simple "Split-Sample" code options are equivalent to Huffman codes.[2]

Thus with enough of these easily implemented code options, the average performance of the adaptive coder in Figure 2 will look like that shown in Figure 3. That is, such an AVLC will be "efficient" everywhere except at very low entropies.

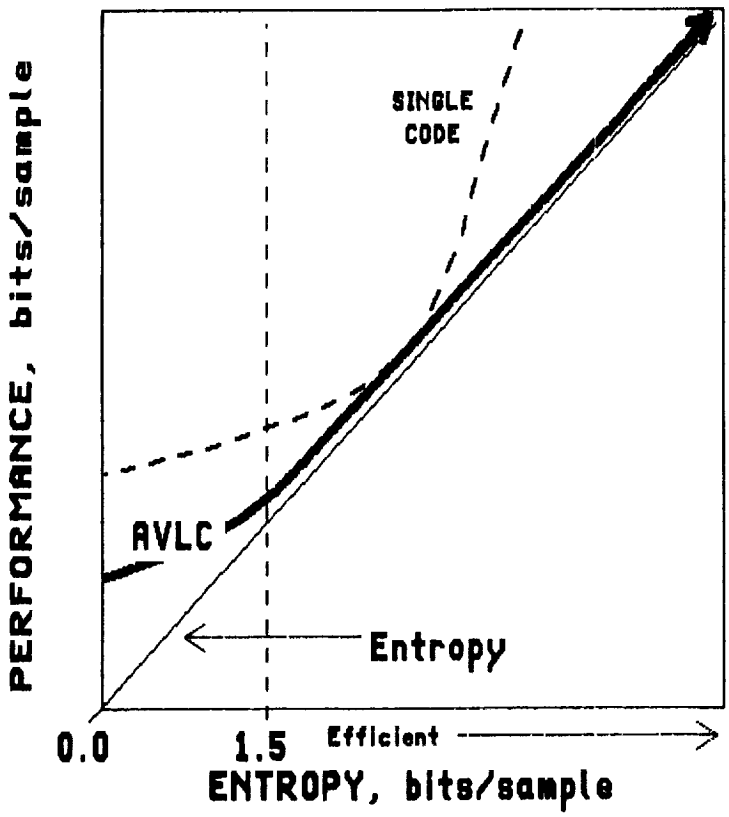

Figure 3. AVLC Performance Characteristics

CODING MODULE

For a vast number of real problems, the desired step to accomplish the conversion of an instrument data source into the desired standard source form can be accomplished by a simple predictive preprocessor. We then define an overall "coding module" in Figure 4 that incorporates this simple preprocessor along with an AVLC.

First note that a coding module includes a switch with positions C and D. In position C, the AVLC sees the output of the "built-in" preprocessor whereas in position D, the AVLC becomes available for directly coding the output of any external preprocessor.

Now consider the built-in preprocessor itself. For the moment assume that the switch with positions A and B is in position A. Then the output of the "sample delay" is the previous input sample. This acts as a sample prediction that the next sample equals the last. The result of differencing, Δ, is then the error in this prediction. For many problems this is a very good prediction and little can be gained by greater sophistication. But just in case, switch position B allows the module to use an arbitrary external prediction.

In either case, errors tend to be distributed around zero. That is, small differences are generally more likely than larger differences. The MAP function simply converts the differences into the integers expected for a Standard Source (0 maps to 0, -1 maps to 1, +1 maps to 2, -2 maps to 3, +2 maps to 4, and so on). See Refs. 1 and 3 for subtleties on this mapping.

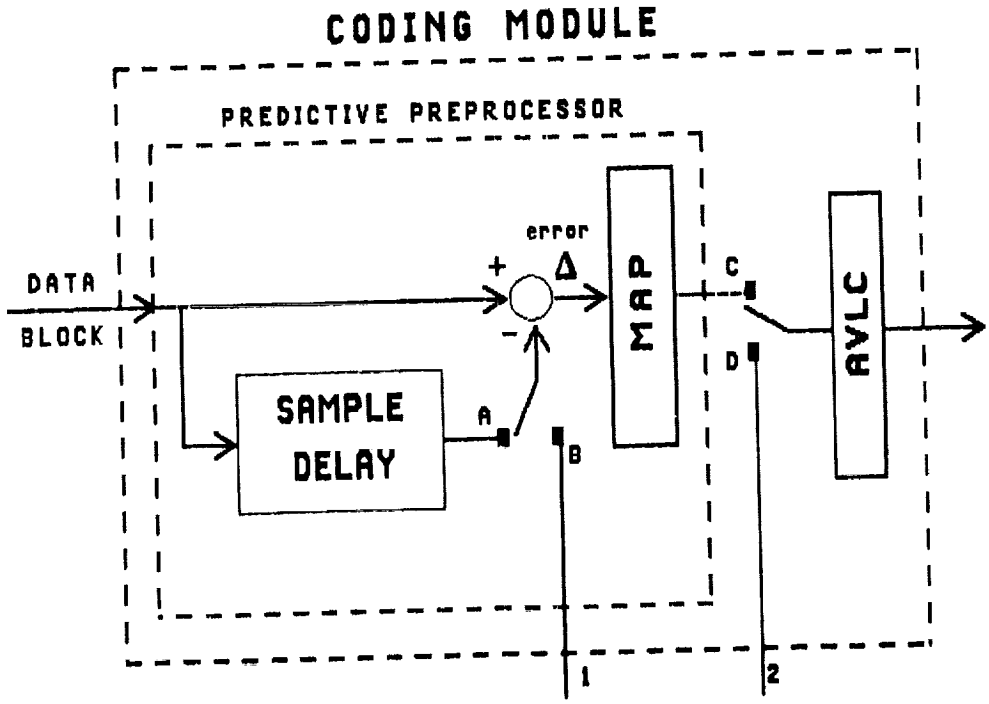

Figure 4. Coding Module

RECENT IMPLEMENTATIONS

Hardware

Two separate programs were recently completed to implement similar versions of the coding module of Figure 4 in high-speed, low-power CMOS VLSI.

The Jet Propulsion Laboratory (JPL) focussed on the design and fabrication of both gate array and standard cell coding module chips. Both were successfully tested in the laboratory in September 1990 at data rates up to 180 Mbits/s.

The JPL developments were driven by severe schedule constraints to meet the specific requirements of a then on-going project. These chips can operate on data quantized to 12 bits/sample (2^{12} data values) and incorporate an 11-option AVLC. The CRAF/Cassini project is currently funding a second-generation chip which will become flight qualified for these missions.

The University of Idaho, under the direction of Goddard Space Flight Center (GSFC) focussed on a more generic chip development unencumbered by the burdens of flight project schedules. A coding module and a companion "decoding" module chip set were recently tested in the laboratory at data rates up to 700 Mbits/s. These chips can operate on data quantized from 4 to 14 bits/sample and incorporate a 12-option AVLC. Both switches in Figure 4 are included in the design.

While not the most general coding module (see Ref. 3) this design includes the most important features needed to support a very broad range of problems. Consequently, its algorithmic specification is the basis of initial NASA

data compression standards efforts. A second-generation chip set (with some slight enhancements) is planned to provide this capability in space qualified form.

Software

The most general parameterized coding/decoding algorithms [3], of which the University of Idaho/GSFC chip set is a subset, have been implemented in C on a specific computer. Planned refinements to make the software more portable and callable will be made available to potential users.

ACKNOWLEDGEMENT

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, the Goddard Space Flight Center and the NASA Space Engineering Research Center for VLSI Design, University of Idaho, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. Robert F. Rice, Pen-shu Yeh, Warner Miller, "Algorithms for a Very High Speed Universal Noiseless Coding Module," JPL Publication 91-1, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

2. Pen-shu Yeh, Robert F. Rice, Warner Miller, "On the Optimality of Code Options for a Universal Noiseless Coder," JPL Publication 91-2, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

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

4. J. Venbrux, et al, "A Very High Speed Lossless Compression/Decompression Chip Set," JPL Publication 91-13, Jet Propulsion Laboratory, Pasadena, California, July 15, 1991.

5. D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," Proc. IRE, Vol. 40, pp. 1098-1101, 1952.