



A New Class of $(2p)B(2p+1)B$ d.c. Balanced Line Codes

Eric Deliot, Alistair N. Coles
Extended Enterprise Laboratory
HP Laboratories Bristol
HPL-98-46
March, 1998

E-mail: ed@hplb.hpl.hp.com

block code,
d.c. balance,
4B5B

A block code of the form $(2p)B(2p+1)B$ cannot provide both d.c. balance and redundant codewords for control signals. We describe a new set of additional rules that may be applied to groups of consecutive codewords in order to provide both d.c. balance and a large control space, and show how 256 10-bit control symbols may be achieved with a d.c. balanced 4B5B code.

A new class of $(2p)B(2p+1)B$ d.c. balanced line codes

Eric Deliot, Alistair N. Coles

Hewlett-Packard Laboratories
Filton Road
Bristol, BS12 6QZ
UK
e-mail: ed@hplb.hpl.hp.com

Abstract: A block code of the form $(2p)B(2p+1)B$ cannot provide both d.c. balance and redundant codewords for control signals. We describe a new set of additional rules that may be applied to groups of consecutive codewords in order to provide both d.c. balance and a large control space, and show how 256 10-bit control symbols may be achieved with a d.c. balanced 4B5B code.

Introduction: $nBmB$ block coding, whereby n bits of data are mapped to $m > n$ bits, is often used in data communication links in order to provide some redundant signal space which may be used for control signals. The transmission overhead introduced by the block code should be kept minimal, and so codes of the type $nB(n+1)B$ are popular. It is also desirable that the size of the input data field (n) is as small as possible and is a factor of 8. These two properties enable low complexity designs for byte-oriented data. The 4B5B code used by FDDI [1] and 100BaseTX [2] is an example of an $nB(n+1)B$ code that meets these criteria.

A further desirable property of a block code is that of d.c. balance. Data is often transmitted through a.c. coupled links (e.g. copper twisted pairs or optical fibres), and any d.c. content in the coded data may cause signal distortion, commonly referred to as baseline wander. A d.c. balanced code reduces baseline wander by ensuring the disparity of coded data (i.e. the difference between the number of ones and the number of zeros) is zero.

For an $nB(n+1)B$ code with n even, i.e. a $(2p)B(2p+1)B$ code, d.c. balance can only be achieved by using two alphabets, for example one containing all codewords having positive disparity and one containing all codewords having negative disparity (there are no codewords with zero disparity). The running disparity (rd) is constrained by choosing codewords from the appropriate alphabet: if $rd > 0$, select a codeword having negative disparity; if $rd \leq 0$, select a codeword having positive disparity. Unfortunately, such a scheme results in all codewords being used in the two data alphabets leaving no redundant codewords for control signalling. In practice, codes such as the FDDI 4B5B must provide some control codewords and are therefore not d.c. balanced. When n is odd, zero disparity codewords may be shared between the two alphabets leaving some redundant codewords. An example of this is the 5B6B code used by IEEE 802.12 networks [3]. However, these codes are harder to implement with byte-oriented data.

One way to provide distinct control signals with a balanced $(2p)B(2p+1)B$ code is for control signals to be represented by codeword sequences (2 or more codewords) that violate the disparity constraining rule described above. This method provides for a relatively small set of control states to be signalled (16 in the case of a balanced 4B5B code for sequences of length 2). This letter presents a new simple method of constructing a d.c. balanced $(2p)B(2p+1)B$ code which provides a much larger control space, from which control sequences may then be selected to satisfy other constraints such as spectral content, runlength, transition density or error detection ability.

A new class of codes: The basic idea used to create the new family of $(2p)B(2p+1)B$ codes is simple: the disparity constraining rule should not be used after every codeword but only after l codewords. The new codes are constructed following these steps:

- 1) Split the total codeword space into two alphabets: an obvious split is positive/negative disparity but as long as an alphabet does not contain a codeword and its complement then any combination is valid. In other words, one alphabet must be the complement of the other alphabet.

- 2) For a sequence of l blocks of $2p$ data bits, each block is coded using a predetermined alphabet, according to its position in the sequence, i.e. from 2^l possible alphabet combinations, select $q < 2^{l-1}$ which are allowed for data.

- 3) Control disparity by complementing codeword sequences (rather than individual codewords), as appropriate. Thus a total of $2q$ alphabet combinations are used for data (q and complements).

For example, consider the case when sequences consist of $l=3$ data blocks which are coded using a single alphabet combination ($q=1$). Out of 8 possible alphabet combinations, suppose $(\mathbf{a}, \mathbf{b}, \mathbf{a})$ has been chosen as a valid data sequence, where \mathbf{a} is an element of alphabet A and \mathbf{b} of alphabet B. Therefore, the first $2p$ data bits are mapped into a codeword from alphabet A, the following $2p$ bits to a codeword from alphabet B and the last $2p$ bits to a codeword from alphabet A again. Then, if required by the disparity rule (step 3), the sequence may be complemented to $(\mathbf{b}, \mathbf{a}, \mathbf{b})$.

The structure given to the data sequences causes the space left for control signalling to increase considerably. The ratio of the total number of alphabet combinations to the number of combinations used for data signalling is $2^{l-1}/q$ and grows quickly with l . Control sequences may use any spare alphabet combination, but also need to be d.c. balanced which implies that they also follow the disparity rule and can be complemented. Therefore in the example above, out of the 6 possible alphabet combinations available for control signalling, if $(\mathbf{a}, \mathbf{a}, \mathbf{a})$, say, was chosen then $(\mathbf{b}, \mathbf{b}, \mathbf{b})$ would also be used. It is assumed that the decoder can not only recognise the boundary between codewords but also recognise the boundary between consecutive groups of l codewords.

As l increases, creating more choice for selecting control sequences, the delay introduced at the coding stage also increases. It is therefore sensible to keep the values of l as low as possible (an example with $l=2$ is detailed in the next section). The method described here may alternatively be seen as a way of designing low complexity

$(2p)B((2p+1)l)B$ block codes. Each group of $2p$ bits can be coded in parallel by coders, the concatenation of the l coders leading to a very simple overall design.

A d.c. balanced 4B5B code with a large control space: Using $p=2$, $l=2$ (a delay of only 10 bits) and $q=1$, a very simple d.c. balanced 4B5B code can be achieved with a large set of control sequences (10 bit long).

Alphabet A is formed from all 16 codewords with positive disparity; alphabet B being its complement contains all negative disparity codewords. Valid data sequences are **(a, b)** and **(b, a)**. The mapping from bits to codewords is not specified since it does not affect the properties of the code. This leaves **(a, a)** and **(b, b)** as possible control sequences (see Figure 1), a choice of 512 control sequences providing 256 control symbols - to constrain disparity, 2 sequences are needed to represent 1 control symbol (sequences **(a, a)** have positive disparity and should be complemented to **(b, b)** sequences as required by the disparity rule).

With such a large control space, extra requirements can be considered. For instance, control sequences can be selected to be Hamming distance 2 from data sequences so that one single error can be detected. Out of the possible 512 sequences, 70 can be found satisfying this requirement, enough to provide 35 control symbols with increased error detection ability.

Summary: A simple method to design d.c. balanced $(2p)B(2p+1)B$ line codes for both data and control has been presented. This technique can provide a very large control space at the cost of introducing some delay at the coding stage. However, the size of the control space grows quickly with the delay and it is clear that constraining this delay to small values will still give very good results. This was illustrated by an example showing the design of a d.c. balanced 4B5B code having 256 control symbols for a delay of only 2 codewords (10 bits).

References

[1] American National Standards Institute, X3T12, "FDDI twisted pair physical layer medium dependent (TP-PMD)", American National Standard, 1994.

[2] IEEE Standard 802.3u-1995, "CSMA/CD Access Method, Type 100Base-T".

[3] IEEE Standard 802.12, "Demand Priority Access Method, Physical Layer and Repeater Specification for 100 Mbit/s Operation".

Figure 1: Example of possible d.c balanced 4B5B coded sequences

