RC23409 (W0411-033) November 3, 2004
Electrical Engineering

IBM Research Report

5B/6B-T(urbo), 3B/4B-T, and Partitioned 8B-/10B-T Transmission Code, and Its Implementation for High Operating Rates

Albert X. Widmer
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Oct. 22, 2004

5B/6B-T(urbo), 3B/4B-T, and Partitioned 8B/10B-T Transmission Code, and its Implementation for High Operating Rates

Albert X. Widmer

IBM T.J. Watson Research Center
1101 Kitchawan Rd. Route 134, Yorktown Heights, NY 10598-0218
914-945-2047, widmer@us.ibm.com

Abstract: A variant of the well known dc-balanced partitioned 8B/10B Transmission code is presented. The encoding and decoding circuits can operate at higher speed or lower power and pipe-lining can be easier implemented. Up to 19 control characters can be defined, two of them are comma characters. The transmission characteristics are virtually identical, but the coding tables are different. Applications which need not be backwards compatible with the traditional code can benefit from this new code.

Field of Subject-Matter

The subject-matter of this report relates to transmission codes and, more particularly, relates to methods and apparatuses used to produce and interpret 5B/6B, 3B/4B, 8B/10B, and 10B/12B transmission codes.

The expression ‘Turbo’ was chosen to express the speed aspects of the code and to set it apart from the well known partitioned 8B/10B code of Ref. 1, and 2. There is no relation whatsoever to the class of transmission codes commonly referred to as ‘Turbo Codes’.

Background

In a partitioned 8B/10B code, an input vector having eight bits is partitioned into two smaller vectors having three and five bits, respectively. Coded vectors having four and six bits, respectively, are created from the partitioned vectors through use of 3B/4B and 5B/6B vector sets. The resultant coded vectors then form a single ten-bit coded vector suitable for transmission. Generally, a control input generates control characters readily recognized as other than the 256 data characters in an 8B/10B transmission code.

The original partitioned 8B/10B code of Ref. 1 and 2, introduced more than 20 years ago, was designed to minimize the number of gates required for encoding and decoding. The original circuitry was implemented with emitter coupled logic (MECL), some versions were also done in bipolar transistor/transistor (TTL) logic, and this was followed by several complementary metal oxide silicon (CMOS) designs. One recent design example is described in Ref. 3.

Lately, design efforts have concentrated on high operating rates. Traditional means for achieving higher operating rates for transmission codes, such as the partitioned 8B/10B transmission code, have involved the parallel operation of several encoders and decoders. Current important applications for the partitioned 8B/10B transmission code and its 5B/6B component are for very wide high speed busses using sets of parallel links similar to Ref. 5, with each serial link operating at several Gbaud and expected to be pushed into the tens of Gbaud. These applications are generally short links which require short latencies for system performance reasons and improvements in this regard are documented in Ref. 3. Operation with a single CoDec (coder/decoder) circuit for each serial link, or a reduction in the multiplexing ratios required at both ends to accommodate parallel CoDec circuits required to serve a single link, is desirable to improve the latency aspect.

Although conventional 8B/10B encoding and decoding work well for a large number of applications, the conventional codes could be improved, particularly in operating rates and latency. Additionally, some applications are compatible with 5-bit and other data units. It would be beneficial to enable the use of the 5B/6B code part alone or in pairs to form a 10B/12B code or any other combination of multiple 5B/6B, 3B/4B, and 1B/2B codes which are all compatible with each other. Such combinations require different comma sequences for boundary demarcation.

In the mean time, several more complicated codes (Ref. 4) with better coding efficiencies have been developed together with improved techniques and insights on how to assign the set of source vectors to a given set of encoded vectors. Based on these improved design techniques, new assignments have been chosen for the nearly identical 8B/10B encoded set with the top consideration of high speed operation and better compatibility with pipelined solutions for the encoder and decoder circuits. The new code can better support the short latency goal both as an 8B/10B code and in a stand alone 5B/6B version as described in Ref. 5.

General Improvements for Higher Speed

The new code retains the 5B/6B and the 3B/4B partitions. Most of the changes are in the 5B/6B domain. For instance, for both the 5B/6B-T encoding and decoding described herein, fewer modifications of bit-positions, in fewer vectors, are performed as compared to conventional 5B/6B techniques. As another example, the S-Function which has an important purpose of preventing false commas, has been reduced to the minimum required to maintain the singularity of the comma at the expense of more frequent single runs of five in random data. A comma generally indicates proper byte boundaries and can be used for instantaneous acquisition or verification of byte synchronization. The K28.7 comma character of the traditional code has been swapped with a formerly invalid control character K3.7 (‘1100001110’ and its complement) which is *not* a comma character but has no sequence restrictions. The reduction of the S-Function vector set allows the definition of seven additional control characters which are listed in Table 3B.

Notation

The signal names used in the equations of this document do not reflect any logic levels. Instead, they should be interpreted as abstract logic statements. However, in the circuit diagrams, the signal names may be prefixed with the letter P or N to indicate whether the function is true at the upper or lower level, respectively. The P and N prefixes are normally not used for net names which start with P and N, respectively. Net numbers starting with 'n' or 'm' are true at the lower level and take the P prefix if true at the upper level. In the logic equations, the symbols •, +, and ⊕ represent the Boolean AND, OR, and EXCLUSIVE OR functions, respectively. The apostrophe (') represents negation.

5B/6B-T Encoding, Table 1

For purposes of this design, the 33 primary 6B vectors are classified into five groups as illustrated by the trellis diagrams of FIGS. 1A, 1B, 2, 3, and 4. All the coded vectors of FIG. 2 through FIG. 4 have alternate, complementary versions assigned to an identical source vector.

Conceptually, coding is done in two steps. We first make the translation to a primary vector. For the coded vectors of FIGS. 1A, 2, 3, and 4, the first five bits of the encoded vector are identical to the source vector and the sixth and last bit assumes a default value of zero. Any vectors with changes in individual bits of the source vector belong to the balanced, disparity independent class of FIG. 1B. A second step for the subset of the disparity dependent coded vectors of FIGS. 2, 3, and 4 determines whether the alternate, complemented vector must be used to meet the disparity rules. Disparity dependent vectors have a plus sign or a minus sign in the DR column of the tables.

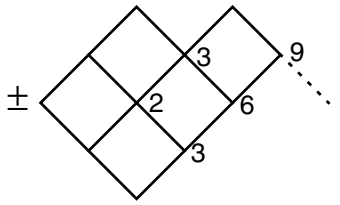


FIG. 1A

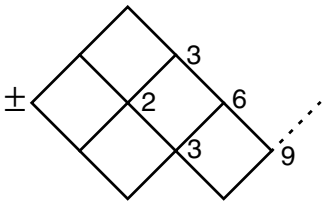


FIG. 1B

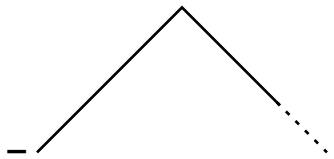


FIG. 2

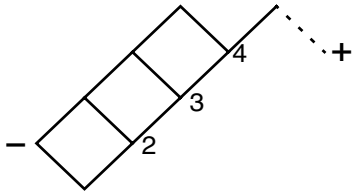


FIG. 3

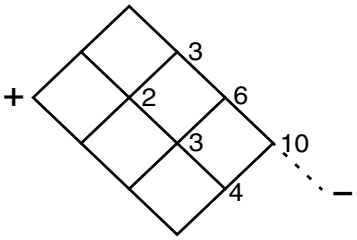


FIG. 4

Generation of Primary Vectors

The logic equations necessary for the translation to the primary vectors can be read directly from the columns 'Primary abcdei' and 'Primary fghj' of Table 1 and Table 2,

respectively. In the ‘Primary’ columns, all plain coded bits are the same as the corresponding input bit values ABCDE or FGH, respectively. The i-bit and the j-bit have a default value of zero. As mentioned above, for the new code, only the nine 6B-T vectors of FIG. 1B require any changes in individual bit values as indicated by the bold and underlined bits which are forced to the complemented value indicated. The complemented bits are also explicitly listed in the column ‘Inverted Bits’.

The encoding equations are extracted from the encoding tables in methodical steps as described below. For each column of a coded bit such as ‘a’, the vectors which require changes are listed, the bits to be complemented are underlined, and the bits which can be used to classify the source vector sets are generally presented in boldface type.

In the Coding Labels and equations, some bit values are included redundantly to allow more circuit sharing for the coding of several bits. Redundant bit values are underlined and redundant vector names are preceded by an asterisk. This redundancy adds to the fan-in and increases the circuit delay. Therefore, if timing margins are not sufficient, the circuit delay can be reduced by removing the redundancies if present in a critical path.

Generation of Primary 6b-T Vectors

The ‘a’ column has bold entries for D0, D15, D16, and D31. The respective uncoded bits ABCDE are listed, the A-bit is underlined, and common patterns are marked to logically classify the vectors by simple expressions.

Table 1.1. a-bit Encoding

Name	ABCDE	Coding Label	a	Name	ABCDE	Coding Label	a
D0	<u>0</u>**000**0	<u>A</u>’•B’•C’•D’	1	D15	<u>1</u>**111**0	<u>A</u>•B•C•D	0
D16	<u>0</u>**000**1	<u>A</u>’•B’•C’•D’	1	D31	<u>1</u>**111**1	<u>A</u>•B•C•D	0

Ignoring the ‘A’ bit for minimum delay, D0 and D16 can be identified as a class by B’•C’• D'; D15 and D31 are identified by B•C•D. Using these labels, the encoding equation for bit ‘a’ can be written as follows:

a = A•(B•C•D)’ + B’•C’• D'

Adding the value of bit ‘A’ redundantly to the first term allows circuit sharing with b-bit encoding which are both never in the critical path. Adding the value of bit ‘A’ redundantly to the second term allows circuit sharing with i-bit encoding, which might be in a critical path. The equation with redundancy for circuit sharing is as follows:

a = A•(<u>A</u>•B•C•D)’ + <u>A</u>’•B’•C’• D'

In the following, only the equations with the redundancy will be shown.

Table 1. 5B/6B-T Encoding

Name	ABCDE K	Inverted Bits	Primary a b c d e i	Alternate a b c d e i	DR Class	DR	DB Class	DB
D0	00000 0	ADI	**1**00**1**0**1**			±	C'•D'•E'•K'	0
D1	10000 0	CI	10**1**00**1**			±	C'•D'•E'•K'	0
D2[2]	01000 0	EI	0100**11**			±	C'•D'•E'•K'	0
D3	11000 0	I	110001			±	C'•D'•E'•K'	0
D4	00100 0	BI	0**1**100**1**			±	ZB6	0
D5	10100 0		101000	010111	PDRS6	+		–
D6	01100 0		011000	100111	PDRS6	+		–
D7	11100 0		111000	000111	NDRS6	–	ZB6	0
D8	00010 0	BI	0**1**0101			±	XB6	0
D9	10010 0		100100	011011	PDRS6	+		–
D10	01010 0		010100	101011	PDRS6	+		–
D11	11010 0		110100			±	XB6	0
D12	00110 0		001100	110011	PDRS6	+		–
D13[1]	10110 0		101100			±	XB6	0
D14[1]	01110 0		011100			±	XB6	0
D15	11110 0	ABI	**00**1101			±	ZB6	0
D16[2]	00001 0	AI	**1**0001**1**			±	XB6	0
D17	10001 0		100010	011101	PDRS6	+		–
D18	01001 0		010010	101101	PDRS6	+		–
D19	11001 0		110010			±	XB6	0
D20	00101 0		001010	110101	PDRS6	+		–
D21	10101 0		101010			±	XB6	0
D22	01101 0		011010			±	XB6	0
D/K23	11101 x		111010	000101	NDRS6	–		+
D24	00011 0		000110	111001	PDRS6	+		–
D25	10011 0		100110			±	YB6	0
D26	01011 0		010110			±	YB6	0
D/K27	11011 x		110110	001001	NDRS6	–		+
D28	00111 0		001110			±	YB6	0
D/K29	10111 x		101110	010001	NDRS6	–		+
D/K30	01111 x		011110	100001	NDRS6	–		+
D31	11111 0	ABDI	**00**1**0**1**1**			±	YB6	0
K3	11000 1		110000	001111	PDRS6	+		–

1. S1 = 1 for PDFS6•C•D•E'•(A≠B)
2. S2 = 1 for NDFS6•A'•C'•D'•(B≠E)

The ‘b’ column has bold entries for D4, D8, D15, and D31.

Table 1.2. b-bit Encoding

Name	ABCDE	Coding Label	b	Name	ABCDE	Coding Label	b
D4	**0**0**1**0**0**	A’•B’•E’•C≠D	1	D15	**1**1**111**0	A•B•C•D	0
D8	**0**0**0**1**0**	A’•B’•E’•C≠D	1	D31	**1**1**111**	A•B•C•D	0

The encoding equation for bit ‘b’ can be written as follows:

$b = B•(A•B•C•D)' + A'•B'•E'•C≠D$

The ‘c’ column has a single bold entry for D1 with a value of one.

D1 1 0 0 0 0

The encoding equation for bit ‘c’ can be written as follows:

$c = C + A•B'•D'•E'$

The ‘d’ column has bold entries for D0 and D31.

Table 1.3. d-bit Encoding

Name	ABCDE	Coding Label	d	Name	ABCDE	Coding Label	d
D0	0000**0**	A’•B’•C’•E’	1	D31	1111**1**	A•B•C•E	0

The encoding equation for bit ‘d’ can be written as follows:

$d = D•(A•B•C•E)' + A'•B'•C'•E'$

The ‘e’ column has a single bold entry for D2 with a value of one.

D2 0 1 0 0 0

The encoding equation for bit ‘e’ can be written as follows:

$e = E + A'•B•C'•D'$

The ‘i’ column has nine entries with a value of one. All are marked in bold because the default value for the i-bit is zero.

Table 1.4. i-bit Encoding

Name	ABCDE K	Coding Label	i	Name	ABCDE	Coding Label	i
D0	00**000 x**	C’•D’•E’•K’	1	*D0	**0000**0	A’•B’•C’•D’	1
D1	10**000 x**	C’•D’•E’•K’	1	D16	**0000**1	A’•B’•C’•D’	1
D2	01**000 x**	C’•D’•E’•K’	1	D15	**1111**0	A•B•C•D	1
D3	11**000 0**	C’•D’•E’•K’	1	D31	**1111**1	A•B•C•D	1
*D0	**00**000 **x**	A’•B’•E’•(C•D)’	1				
D4	**00**10**0 x**	A’•B’•E’•(C•D)’	1				
D8	**00**01**0 x**	A’•B’•E’•(C•D)’	1				

Although the expression A'•B'•E'•(C•D)' for the set of vectors (*D0, D4, D8) is simpler, the expression A'•B'•E'•C≠D without the redundant vector D0 is used instead so it can be shared with circuits required for b-bit encoding. The encoding equation for bit 'i' can be written as follows:

i = A'•B'•E'•C≠D + C'•D'•E'•K' + A'•B'•C'•D' + A•B•C•D

The principal purpose for the S-function (S1 and S2 annotation in Table 1) is to prevent false commas in the bits 'cdeifgh'. In the original 8B/10B code of Ref. 1 and 2, the vectors D11 and D31 (D20 in the original code) were included in the S-function only to lower the incidence of runs of five. They are left out now for reduced logic delay in the path for the encoding of the f and j bits which have now delays comparable with the other bits. This simplifies and increases the effectiveness of pipe-lining which is preferred for some high speed applications over parallel operation of several encoders for a single serial lane. The incidence of runs of five is now an average of about once per 256 bytes of random data, but there are still no contiguous runs of five.

For a *positive* running disparity at the front of the 6B vector (PDFS6), the S function must be asserted for the two balanced vectors for which the encoded pattern ends with cdei=1100.

	Source Vector	Encoded Vector
D13	1 0 **1 1 0**	1 0 1 1 0 0
D14	0 1 **1 1 0**	0 1 1 1 0 0

S1 = PDFS6•C•D•E'•(A≠B)

For a *negative* running disparity at the front of the 6B vector (NDFS6), the S function must be asserted for the two balanced vectors for which the encoded pattern ends with cdei=0011.

	Source Vector	Encoded Vector
D2	**0** 1 **0 0** 0	0 1 0 0 1 1
D16	**0** 0 **0 0** 1	1 0 0 0 1 1

S2 = NDFS6•A'•C'•D'•(B≠E)

3B/4B-T Encoding, Table 2

New 3B/4B-T vector assignments have been made following similar guidelines as used for the 5B/6B-T part. Changes are also required to accommodate the control character swap referred to above. The 3B/4B-T encoding equations can be directly derived from Table 2.

For the 'f' column, the coding equation is as follows:

f = F • [F •G•H•(S+Kx)]' = F•[G•H•(S+Kx)]'

where S = S1 + S2.

Just below, the 5B part of all control characters of Tables 3A and 3B is listed. Kx is an abbreviation for the 5B or 6B part of all control characters other than K3. If the set of valid control characters does not include K11.7 of Table 3B, then Kx = K•E is true. If K11.7 is included, then Kx = K•(D+E) is true.

	Source Vector		Source Vector	
K23	1 1 1 0 1	1	K19	1 1 0 0 1 1
K27	1 1 0 1 1	1	K21	1 0 1 0 1 1
K29	1 0 1 1 1	1	K22	0 1 1 0 1 1
K30	0 1 1 1 1	1	K25	1 0 0 1 1 1
K3	1 1 0 0 **0**	1	K26	0 1 0 1 1 1
K11	1 1 0 1 0	1	K28	0 0 1 1 1 1

Table 2. 3B/4B-T Encoding

Name	F G H K	Coding Class	Primary f g h j	Alternate f g h j	DR Class	DR	DB Class	DB
Dx.0	0 0 0 0	F'•G'•H', G'•H'	0 **1** 0 **1**			±	G'•H'	0
K3.0	0 0 0 1	F'•G'•H', G'•H'	0 **1** 0 **1**	1 0 1 0	K•(F•G)'	+	G'•H'	0
Dx.1	1 0 0 0	G'•H'	1 0 0 **1**			±	G'•H'	0
K3.1	1 0 0 1	G'•H'	1 0 0 **1**	0 1 1 0	K•(F•G)'	+	G'•H'	0
Dx/K3.2	0 1 0 x		0 1 0 0	1 0 1 1	F'•(G≠H)	+		
Dx/K3.3	1 1 0 x		1 1 0 0	0 0 1 1	F•G	–	F•(G≠H)	0
Dx/K3.4	0 0 1 x		0 0 1 0	1 1 0 1	F'•(G≠H)	+		
Dx.5	1 0 1 0		1 0 1 0			±	F•(G≠H)	0
K3.5	1 0 1 1		1 0 1 0	0 1 0 1	K•(F•G)'	+	F•(G≠H)	0
Dx.6	0 1 1 0		0 1 1 0			±	F'•G•H	0
K3.6	0 1 1 1		0 1 1 0	1 0 0 1	K•(F•G)'	+	F'•G•H	0
Dx/K3.P7	1 1 1 x		1 1 1 0	0 0 0 1	F•G	–		
Dx.A7[1]	1 1 1 0	F•G•H•S	**0** 1 1 **1**	1 0 0 0	F•G	–		
Kx.A7[2]	1 1 1 1	Kx	**0** 1 1 **1**	1 0 0 0	F•G	–		

1. S = 1 for PDFS6•C•D•E'•(A≠B) + NDFS6•A'•C'•D'•(B≠E)
2. If the set of valid control characters does not include K11.7 of Table 3B, then Kx = K•E is true. If K11.7 is included in the set of control characters, then Kx = K•(D+E)

Because the primary interest here is minimum circuit delay rather than circuit area, the equation for bit 'f' is transformed as follows to reduce the logic depth:

f' = F'+G•H•S1+G•H•S2+G•H•K•E

It is assumed that only valid values for control are presented at the input to the encoder, i.e. all control characters are either K3.y where y has any value from 0 to seven, or Kx.7. With this assumption, the last term G•H•K•E above can be reduced to K•E.

f'=F'+G•H•C•D•E'•(A≠B)•PDFS6+G•H•A'•C'•D'•(B≠E)• NDFS6+ K•E

For the actual circuit implementation, the term G•H is expanded back to F•G•H because the full term is required for the j-bit encoding anyway.

For the 'g' column, the coding equation is as follows:

$g = G + F' \bullet G' \bullet H' = G + F' \bullet H'$

For the 'h' column, the coding equation is as follows: $h = HK \bullet E$

For the 'j' column, the coding equation is as follows:

$j = G' \bullet H' + F \bullet G \bullet H \bullet S + Kx$

$j = G' \bullet H' + F \bullet G \bullet H \bullet S1 + F \bullet G \bullet H \bullet S2 + Kx$

$j = G' \bullet H' + F \bullet G \bullet H \bullet C \bullet D \bullet E' \bullet (A \neq B) \bullet PDFS6 + F \bullet G \bullet H \bullet A' \bullet C' \bullet D' \bullet (B \neq E) \bullet NDFS6 + K \bullet E$

Control Characters

A basic set of 12 control characters is listed in Table 3A. The coded format of all characters except K3.7 is identical to the traditional partitioned 8B/10B code. However, the source vector K28 is replaced by its complement K3 and so the name for the respective identical coded vectors is changed. K3.1 and K3.5 are comma characters and the comma sequence is printed in bold type. Note that because of the run length limit of 5, the second bit (b) of the sequence can be left out for purposes of comma search circuits which limits the search to 1x00000 or 0x11111.

In Table 3A, the column heading 'Primary (6B)' refers to the fact that the 6B part of the 10-bit control vectors of that column are primary vectors. The 4B part may be the alternate vector. This distinction has no significance beyond semantics.

Table 3A. Basic Set of Control Characters

Name	ABCDE FGH K	Primary (6B) DR abcdei fghj DB	Alternate DR abcdei fghj DB
K3.0	11000 000 1	+ 110000 1010 –	– 001111 0101 +
K3.1	11000 100 1	+ **110000 0**110 –	– **001111 1**001 +
K3.2	11000 010 1	+ 110000 1011 0	– 001111 0100 0
K3.3	11000 110 1	+ 110000 1100 –	– 001111 0011 +
K3.4	11000 001 1	+ 110000 1101 0	– 001111 0010 0
K3.5	11000 101 1	+ **110000 0**101 –	– **001111 1**010 +
K3.6	11000 011 1	+ 110000 1001 –	– 001111 0110 +
K3.7	11000 111 1	+ 110000 1110 0	– 001111 0001 0
K23.7	11101 111 1	– 111010 1000 0	+ 000101 0111 0
K27.7	11011 111 1	– 110110 1000 0	+ 001001 0111 0
K29.7	10111 111 1	– 101110 1000 0	+ 010001 0111 0
K30.7	01111 111 1	– 011110 1000 0	+ 100001 0111 0

An additional set of seven control characters are defined in Table 3B. All these characters use the alternate A7 coding in the 4B domain when following a 6B vector which does not require the alternate vector for purposes of compliance with the coding constraints. These balanced 6B vectors use the primary P7 vector in the 4B domain for data and the alternate vector A7 for control characters. For control characters, the 6B part is a disparity dependent balanced vector which is complemented by analogous rules as are the balanced 4B vectors when the K-bit has a value of one. Because 12 control characters are enough for most applications, these seven extra characters are not implemented in the circuit diagrams attached and treated as invalid characters. Where appropriate, guidance is given on how to modify the logic equations for their implementation.

Table 3B. Additional Control Characters

Name	ABCDE FGH K	Primary (6B) DR abcdei fghj DB	Alternate DR abcdei fghj DB
K11.7	11010 111 1	+ 110100 1000 –	– 001011 0111 +
K19.7	11001 111 1	+ 110010 1000 –	– 001101 0111 +
K21.7	10101 111 1	+ 101010 1000 –	– 010101 0111 +
K22.7	01101 111 1	+ 011010 1000 –	– 100101 0111 +
K25.7	10011 111 1	+ 100110 1000 –	– 011001 0111 +
K26.7	01011 111 1	+ 010110 1000 –	– 101001 0111 +
K28.7	00111 111 1	+ 001110 1000 –	– 110001 0111 +

Implementation of 8B/10B-T Encoding

A first circuit, T_s10encode, illustrated in FIGS. 5A and 5B shows the bit encoding and identification of the disparity attributes, respectively. Because both figures represent a single circuit, there is sharing of nets a among them. A symbol of the circuit with all input and output signals is shown in FIG. 6. The disparity attributes generated by the circuit of FIG. 5B are used in a variety of ways depending on the required operating speed (relative to the performance of the logic circuit family used for implementation). Representative examples are shown in FIGS. 6, 7, 8, and 9.

8B/10B-T Bit Encoding Circuit, FIG. 5A

An implementation according to the above tables, equations and design principles is illustrated in the circuit diagram 8B/10B-T Bit Encoding, T-s10encode, of FIG. 5A. The design presented here assumes that all inputs are available in complementary form. If this is not true, minor restructuring of the circuit is needed for minimum delay.

Notation for net names in the encoding circuit diagrams: The letters ‘a’ and ‘o’ within net-names refer to the Boolean AND and OR functions, respectively. The letter ‘n’ within a name negates the preceding parameter. The letters ‘e’ and ‘ue’ represent the symbols ‘=’ and ‘≠', respectively. The capital letters ABCDEFGHK represent the uncoded input bits and the lower case letters abcdeifghj represent the coded format. These clumsy notations

have been adopted because of the limitations of the logic design system which was used for the generation of the logic diagrams.

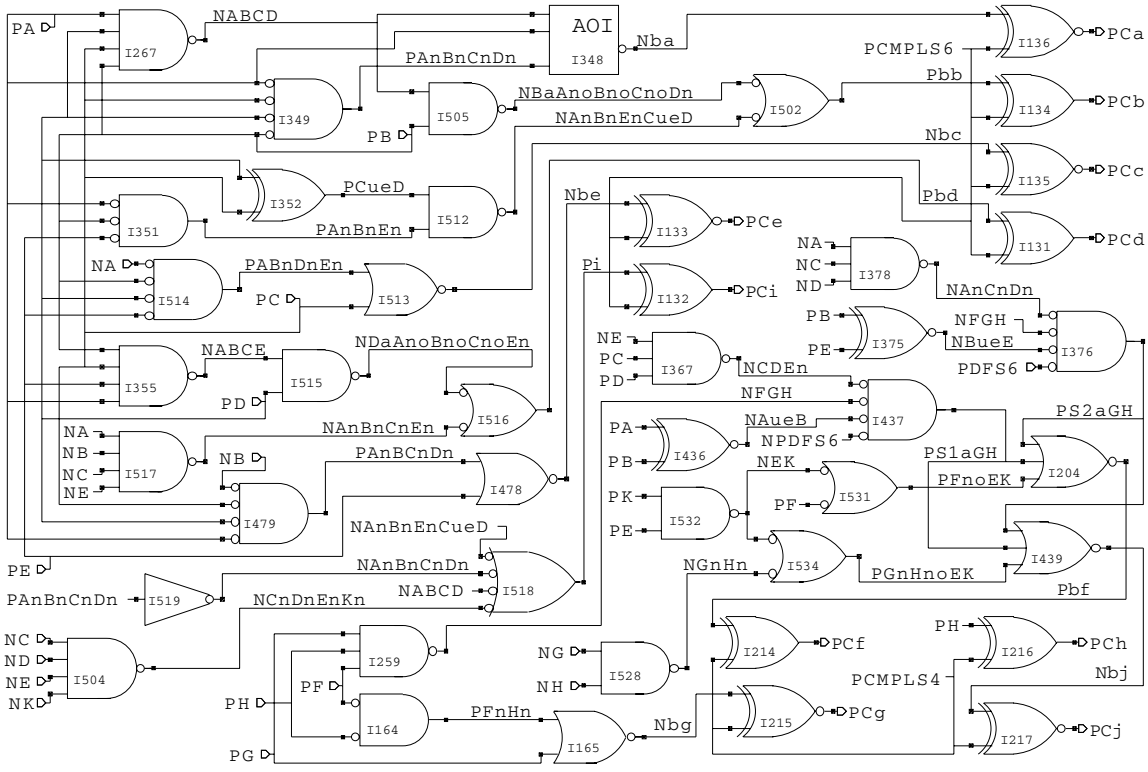


FIG. 5A. 8B/10B-T BIT ENCODING

8B/10B-T Disparity Attributes, FIG. 5B

The column *'DR Class'* (DR = Required Running Disparity) in Table 1and 2 classifies the vectors according to the plus sign and the minus sign entries in the DR column which indicates the required disparity at the front of the primary encoded vector. The expressions PDRS6, PDRS4 and NDRS6, NDRS4 represent a positive or negative required disparity, respectively, at the start of the 6B or 4B vectors. These signal names do not appear in the circuit diagram, because the gating required for CMPLS6 and CMPLS4 described below has been merged with said functions in order to eliminate one gating level.

PDRS6

The set of ten primary 6B vectors with a negative block disparity and a plus sign in the DR column of Table 1 is referred to as PDRS6. All are illustrated in the trellis diagram of FIG. 4. They are generated by appending a zero bit to the following source vectors:

D5	1 0 1 0 **0**	D12	**0** 0 1 1 0	D17	1 **0 0** 0 **1**
D6	0 1 1 0 **0**	D18	**0** 1 0 0 1	D24	0 **0 0** 1 **1**
D9	1 0 0 1 **0**	D20	**0** 0 1 0 1	K3	1 1 0 0 0 K=1
D10	0 1 0 1 **0**	*D10	**0** 1 0 1 0		

- The vectors D5, D6, D9, and D10 can be identified by the expression E'•(A≠B)•(C≠D).
- The vectors D12, D18, D20, and D10 (redundant) are identified by A'•(B≠C)•(D≠E).
- The vectors D17 and D24 are identified by B'•C'•E•(A≠D).
- The control vector K3 can be identified by E'•K since an examination of Table 1 shows, that all the 6B control vectors of Table 3A with a positive block disparity (K23, K27, K29, and K30) have all a value of one in bit position E.

The equation for PDRS6 can thus be expressed as follows:

PDRS6 = E'•(A≠B)•(C≠D) + A'•(B≠C)•(D≠E) + B'•C'•E•(A≠D) + E'•K

If all the control characters of Table 3B are included, the last term E'•K must be replaced by:

K•(A•B•D•E+A•B•C+A•C•D+B•C•D)'

If K11.7 is not included in the valid set, the above expression is simplified to:

K•(A•B•D+A•B•C+A•C•D+B•C•D)'

NDRS6

The set of five 6B vectors with a minus sign in the DR column of Table 1 is referred to as NDRS6. Four of these vectors have a positive block disparity and are illustrated in the trellis diagram of FIG. 3. The balanced vector D7 (111000) of FIG. 2 also requires a negative entry disparity.

D7	**1 1 1 0** 0	D/K29	1 0 **1 1 1**	D/K27	1 1 0 1 1
D/K23	**1 1 1 0** 1	D/K30	0 1 **1 1 1**		

- The vectors D7 and D/K23 can be identified by A•B•C•D'.
- The vectors D/K29and D/K30 are identified by C•D•E•(A≠B).
- The vector D/K27 is identified by A•B•C'•D•E.

So the equation for NDRS6 is as follows:

NDRS6 = A•B•C•D' + C•D•E•(A≠B) + A•B•C'•D•E

PDRS4

Table 2 shows a plus sign in the DR column for the following six vectors shown with their uncoded values FGH K:

K3.0	0 0 0	**1**	Dx/K3.2	**0** 1 0	x
K3.1	1 0 0	**1**	Dx/K3.4	**0** 0 1	x
K3.5	1 0 1	**1**			
K3.6	0 1 1	**1**			

- The four vectors in the left column above can be identified by K•(F•G)' since none of the control characters have a DR entry of ± and all control characters with a negative DR have bit values of one for both the F and the G bit as shown in Table 2.
- The two vectors of the right column are identified by F'•(G≠H).

The equation for PDRS4 is as follows:

PDRS4 = K•(F•G)' + F'•(G≠H)

NDRS4

There are four rows in Table 2 with a minus entry in the DR column. They all can be uniquely identified by F•G. Therefore:

NDRS4 = F•G

CMPLS6 and CMPLS4

If the running disparity DF in front of the vector does not match the required entry disparity DR, a complement signal is generated which selects the alternate vector.

CMPLS6 = NDFS6•PDRS6 + PDFS6•NDRS6

CMPLS4 = NDFS4•PDRS4 + PDFS4•NDRS4

In the circuit diagrams, the signal names PDFS6, PDFS4 and NDFS6 and NDFS4 represent the actual running disparity at the front of the 6B and 4B vectors, respectively. Note that in the above two equations, the signals NDFS and PDFS are complementary and the signals PDR and NDR are orthogonal, i.e. only one can be true, but both can be false.

BALS6

The set of 19 primary 6B vectors of FIGS. 1A, 1B, and 2 are balanced and identified by a 0 in the column *'DB Class'* (Block Disparity). This set of vectors is referred to as BALS6 and can be grouped as shown below:

D0 0 0 **0 0 0**	D8 0 0 0 **1 0**	D25 1 0 0 **1 1**	*D0 **0 0** 0 **0 0**
D1 1 0 **0 0 0**	D11 1 1 0 **1 0**	D26 0 1 0 **1 1**	D4 **0 0** 1 **0 0**
D2 0 1 **0 0 0**	D13 1 0 1 **1 0**	D28 0 0 1 **1 1**	
D3 1 1 **0 0 0** K=0	D14 0 1 1 **1 0**	D31 1 1 1 **1 1**	D7 **1 1 1** 0 **0**
	D16 0 0 0 **0 1**		D15 **1 1 1** 1 **0**
	D19 1 1 0 **0 1**		
	D21 1 0 1 **0 1**		
	D22 0 1 1 **0 1**		

- The 4 vectors D0, D1, D2, and D3 are identified by C'•D'•E'•K'.
- The 8 vectors D8, D11, D13, D14, D16, D19, D21, and D22 are identified by XB6 = (D≠E)•(A'•B'•C'+A•B•C'+A•B'•C+A'•B•C) = (D≠E)•(A⊕B⊕C)'.

- The 4 vectors D25, D26, D28, and D31 are identified by
  YB6 = D•E•(A•B•C+A•B'•C'+A'•B•C'+A'•B'•C) = D•E•(A⊕B⊕C).
- The 2 vectors D0 (redundant from Column 1) and D4 are identified by
  WB6 = A'•B'•D'•E'.
- The 2 vectors D7 and D15 are identified by
  ZB6 = A•B•C•E'.

So the equation for BALS6 can be expressed as:

BALS6=(D≠E)•(A⊕B⊕C)'+D•E•(A⊕B⊕C)+A'•B'•D'•E'+C'•D'•E'•K'+A•B•C•E'

In the usual circuit implementations, the signal BALS6 is in the critical delay path and required in true and complement form which requires an inversion with extra delay. This problem can be side-stepped by generating the UNBALS6 signal directly from the inputs similar to the circuit for BALS6. The UNBALS6 signal requires only 9 gates and is slightly less complex and has a little less delay, so if only one of the signals is generated and then inverted, preference should be given to the UNBALAS6 signal. It is derived from the following grouping of all the vectors which have an entry other than 0 in the column DB of Table 1.

D5	**1 0** 1 0 0	D12	**0 0** 1 1 0	D/K23	**1 1** 1 0 **1**	K3	1 1 0 0 0 0 K=**1**
D9	**1 0** 0 1 0	D20	**0 0** 1 0 1	D/K27	**1 1** 0 1 **1**		
D17	**1 0** 0 0 1	D24	**0 0** 0 1 1				
D/K29	**1 0** 1 1 1						
D6	**0 1** 1 0 0						
D10	**0 1** 0 1 0						
D18	**0 1** 0 0 1						
D/K30	**0 1** 1 1 1						

- The 8 vectors D5, D9, D17, D/K29, D6, D10, D18, and D/K30 in the left column of the list below are identified by:
  XUB6 = (A≠B)•(C•D'•E'+C'•D•E'+C'•D'•E+C•D•E) = (A≠B)•(C⊕D⊕E).
- The three vectors D12, D20, and D24 in the second column can be identified by:
  YUB6 = A'•B'•(A•B•C'+A•B'•C+A'•B•C).
- The two vectors D/K23 and D/K27 in the third column can be identified by:
  ZUB6 = A•B•E•(C≠D).
- The single vector K3 in the right column can be uniquely identified by K=1 since all encoded 6B K vectors of Table 3A are unbalanced.

So the equation for UNBALS6 can be expressed as:

UNBALS6 = (A≠B)•(C⊕D⊕E) + A'•B'•(A•B•C'+A•B'•C+A'•B•C) + A•B•E•(C≠D) + K

If all the control characters of Table 3B with a balanced 6B part are included, the K3 vector must be identified by: K•D'•E'. If K11.7 is excluded, this is simplified to K•E'.

BALS4

The set of source vectors with a zero in the DB column of Table 2 is referred to as BALS4.

Dx/K3.0 0 **0 0**	Dx/K3.3 **1** 1 0	Dx/K3.6 0 1 **1**
Dx/K3.1 1 **0 0**	Dx/K3.5 **1** 0 1	

$$BALS4 = G'\bullet H' + F\bullet(G \neq H) + F'\bullet G\bullet H$$

Additional circuits governed by the signals BALS6 and BALS4 indicate the balance of a byte by the signal PBALBY which assumes the upper level for a balanced byte.

The technique for reducing the delay for the disparity function extending over one or more bytes as taught in Ref. 3 (FIGS. 3B, 5, 6, 7, and 8) apply equally to the code presented here and is summarized briefly again. A reduction in the combined delay of a 5B/6B and a 3B/4B encoder or of several 8B/10B encoders operating in parallel results from the methodology used to determine the disparity at any vector boundary as shown at the bottom of circuit diagram of FIG. 5B and on the diagram of FIG. 6. Given a starting disparity such as PDFBY (Positive Disparity in Front of a Byte), the running disparity at any subsequent vector boundary remains unchanged if the combined number of balanced S6 and S4 vectors between the two points is even, otherwise it assumes the complementary polarity. This is in contrast to the more obvious techniques which observe the disparity as it propagates from vector to vector. The expressions PDFS6 and PDFS4 represent a positive running disparity in front of a 6B and a 4B vector, respectively.

The 8B/10B encoder has an output which indicates whether the coded 10-bit byte is balanced or not, but there is no output to indicate the ending disparity. Generally, the starting disparity for a vector is determined from the disparity of a prior reference point and the odd or even number of balanced vectors in between.

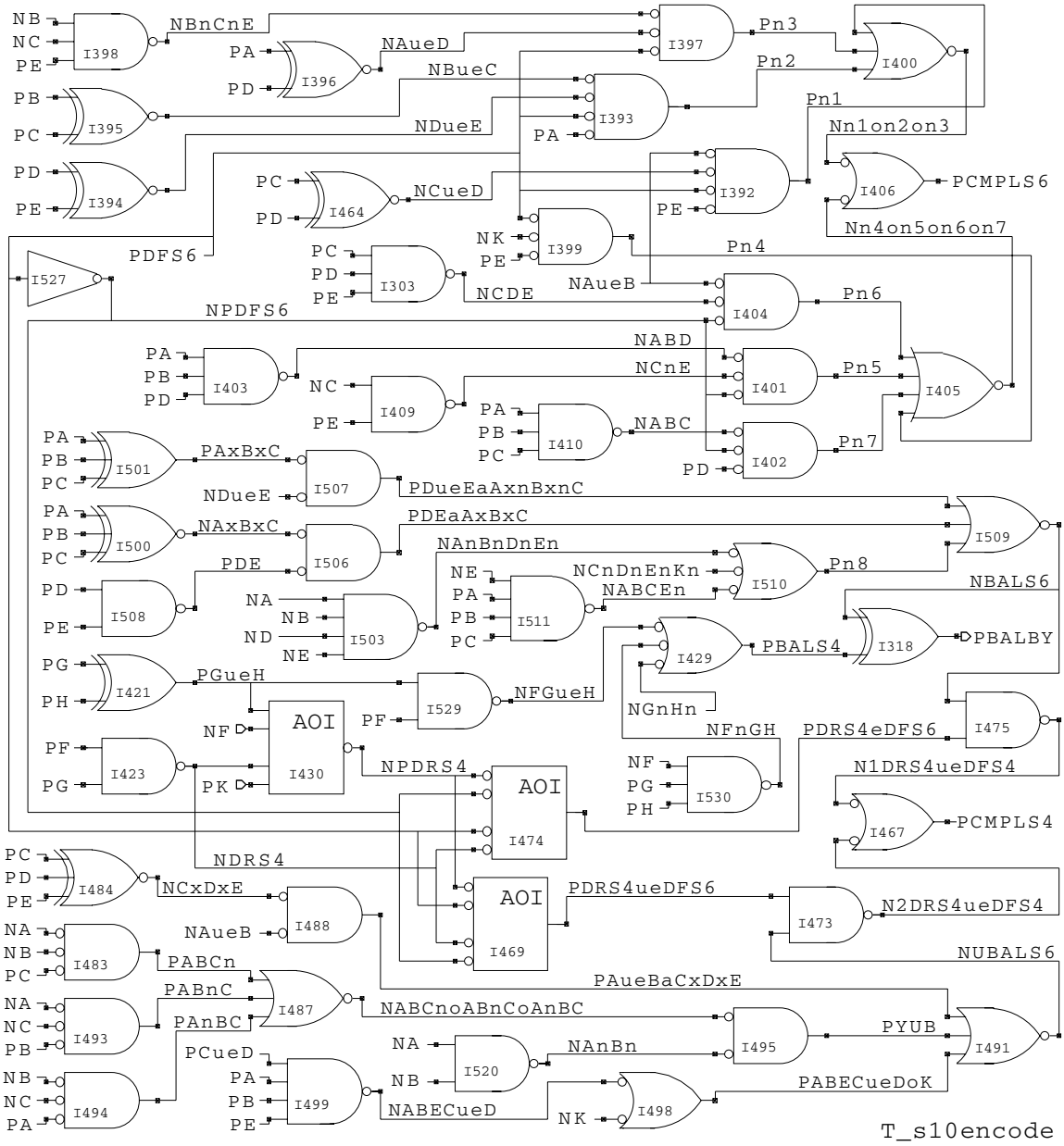


FIG. 5B. 8B/10B-T DISPARITY CONTROL 12-05-03

In the circuit of FIG. 5B, the following abbreviations are used for net names:

n1 = NDFS6•E'•A≠B•C≠D

n2 = NDFS6•A'•B≠C•D≠E

n3 = NDFS6•B'•C'•E•A≠D

n4 = NDFS6•K•E'

n5 = PDFS6•A•B•C'•D•E

n6 = PDFS6•C•D•E•A≠B

n7 = PDFS6•A•B•C•D'

Disparity Circuits

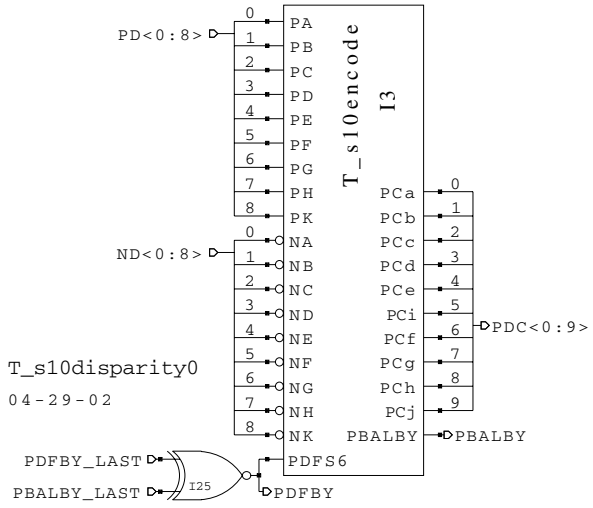


FIG.6. DISPARITY FOR 1-BYTE ENCODER
Fast Version

Several disparity circuits are presented now suitable for slower or faster operation with single byte encoders or four encoders operating in parallel to generate a single coded bit stream.

Disparity Circuit for 1-Byte Encoder, Fast Version, FIG. 6

The first disparity circuit that will be described allows faster operation relative to a slower version (described below). The fast version of FIG. 6 takes advantage of the fact that it is not necessary that the starting disparity must be known immediately for the encoding process. Since the evaluation of the running disparity at the end of a byte may be in the critical delay path, the final operations for determining the starting disparity PDFBY of the next byte are deferred to the next byte interval to be executed while initial bit encoding independent of the running disparity is performed. The cost of doing this is to pass along two parameters rather than just one to the next byte interval. The method increases the timing margin by an amount equal to the delay of the XNOR2 gate I25 in FIG. 6.

The circuit of FIG. 6 uses the encoding circuitry T_s10encode of FIGS. 5A and 5B to generate the encoded byte PDC<0:9> and a signal PBALBY, which is at the upper level when the coded byte is balanced. At the end of each byte cycle, the signals PDFBY and PBALBY are stored in two latches with outputs PDFBY_LAST and PBALBY_LAST, respectively, for use in the next cycle. The respective latches are not shown since their timing is identical to or closely related to the timing for the data output latches. The XNOR2 gate I25 provides the starting disparity PDFBY for the current byte based on the starting disparity PDFBY and the PBALBY value carried over from the preceding byte (PDFBY_LAST, PBALBY_LAST). The signal PDFBY is at the upper level for a positive running disparity in front of the new byte.

There are 2 coded vectors per byte (6B, 4B). So if there is an odd number of *balanced or unbalanced* vectors between the start of the current byte and a previous byte boundary, the starting disparity for the current byte is the complement of the disparity at the reference point, otherwise it is the same.

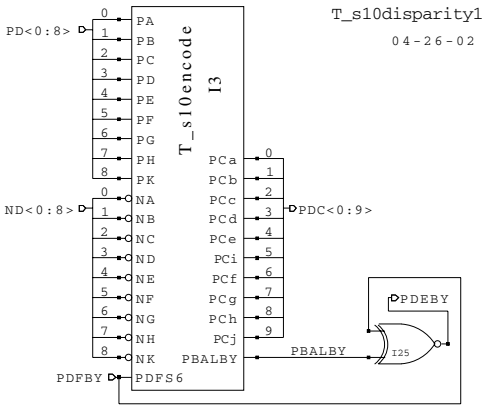


FIG. 7. DISPARITY FOR 1-BYTE ENCODER
Slower Version

Disparity Circuit for 1-Byte Encoder, Slower Version, FIG. 7

To better illustrate the fast approach to disparity operations, the more traditional way is also shown in FIG. 7 and is applicable where the higher performance is not needed. In traditional circuits, the ending disparity PDEBY is derived within one and the same encoding cycle. Then only one parameter must be passed on to the next cycle with a single latch. The data input of this latch is PDEBY and the output is PDFBY, the disparity at the front of the next byte.

Disparity Circuit for 4-Byte Encoder, Fast Version, FIG. 8

A disparity circuit is shown in FIG. 8 for determining disparity for four bytes. The circuit shows four encoders operating in parallel on a 4-byte word. The starting disparity of the block and of the first byte is given by the input PDFW (Pos. Disp. in Front of the Word). It is generated from signals carried over from the preceding word cycle by a pair of latches (not shown), i.e the running disparity PDF3_LAST in front of the last byte (#3) and the balance signal PBALBY3_LAST of byte #3. The starting disparity for each of the remaining 3 bytes is obtained by circuits operating in parallel from this reference point and the number of balanced bytes in between using a set of XOR and XNOR gates. The signal PBAL012 is at the upper level if the block comprising the first 3 bytes is balanced.

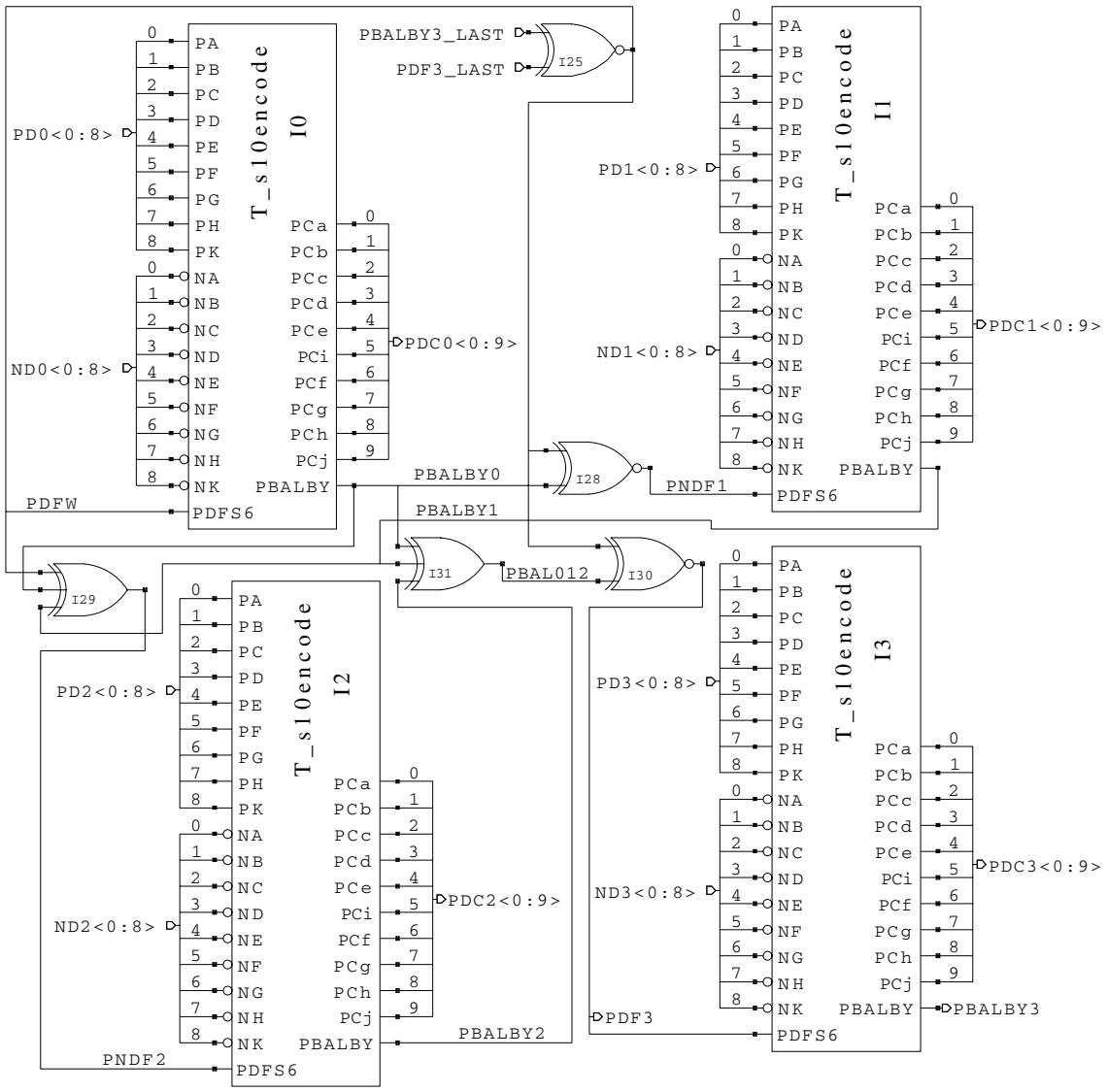


FIG. 8. DISPARITY FOR 4-BYTE ENCODER
Fast Version

T_s40disparity0
12-11-03

Disparity Circuit for 4-Byte Encoder, Slower Version, FIG. 9

A disparity circuit is shown in FIG. 9 for determining disparity for four bytes. The circuit shown in FIG. 9 is slower than the circuit shown in FIG. 8. In FIG. 9 the ending disparity PDEW of the word is generated within a single clock cycle. A single latch (not shown) with the signal PDEW at the data input passes along to the next cycle the ending disparity for the word. The output of this latch is the starting disparity PDFW for the next cycle.

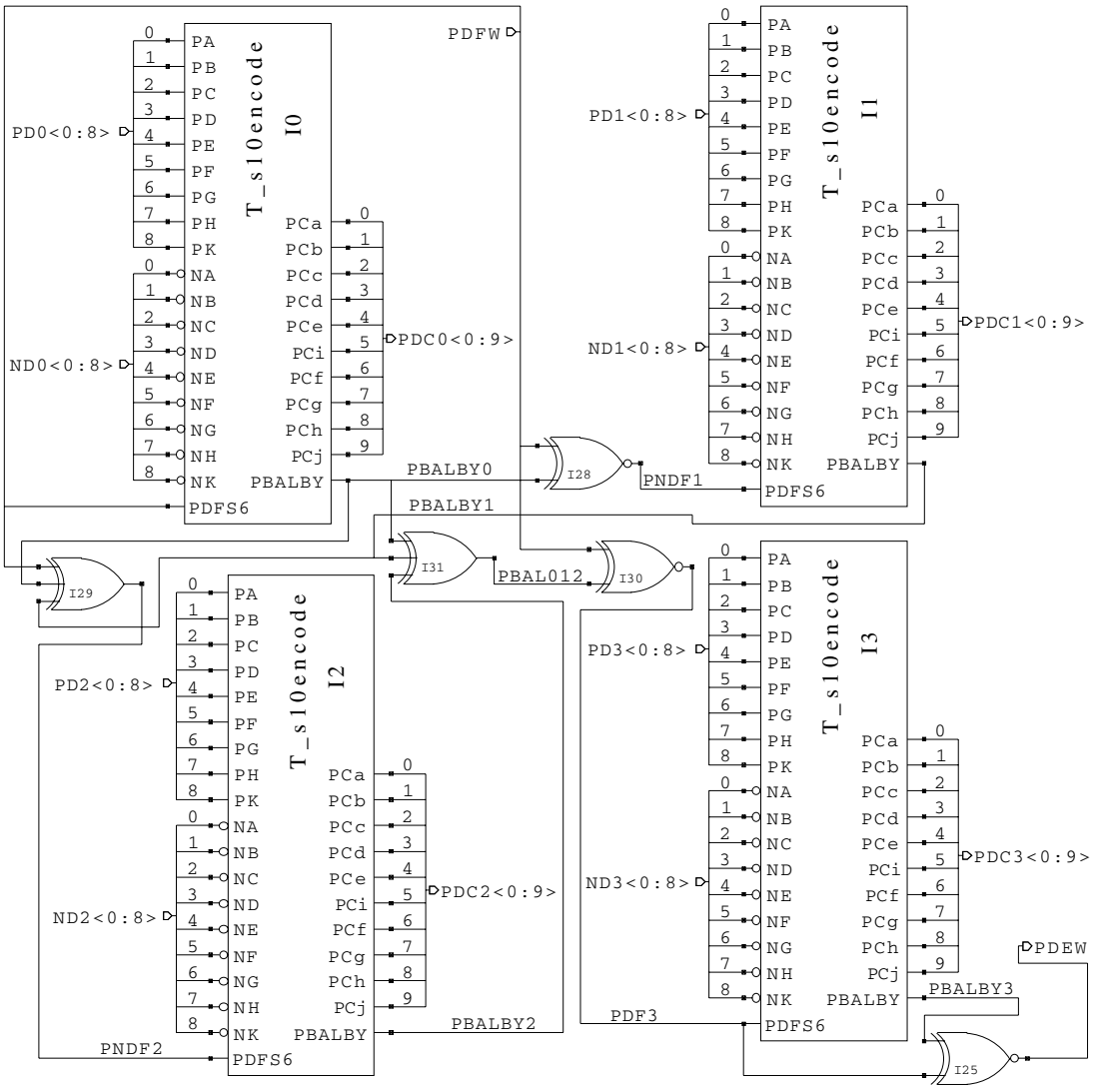


FIG.9. DISPARITY FOR 4-BYTE ENCODER
Slower Version

T_s40disparity1
12-11-03

10B/8B-T Decoder

The decoder comprises circuits to restore the original byte ABCDEFGH K, and circuits to indicate all transmission errors to the extent that they are detectable by the transmission code. For decoding, the trailing bits ‘i’ and ‘j’ are just dropped but their value guides some of the decoding functions.

The tables 4 and 5 (described below) show the relationships between the coded and decoded vectors for the 6B/5B-T and 4B/3B-T decoding, respectively. For some vector names, there are several rows to represent the true and complement version and to show the different rules for the decoding of specific bits. If for a row, the equation in the column ‘Decoding Class’ is true, the bold, underlined bits in the column ‘ABCDE K’ are

complemented. The suffix ‘P’ or ‘A’ of a vector name refers to the primary or alternate version of a vector where applicable.

A set of circuits for 10B/8B decoding is illustrated in FIGS. 10A, 10B, 11, and 12. To avoid confusion with some of the lower case letters abcdeifghj which represent the coded bits, the notation for Boolean operators in the decoding circuits uses capital letters (A, O, N, E, UE) instead of the lower case letters used in the encoder circuits.

6B/5B-T Decoder, Table 4

The relationship between the coded 6B vectors and the corresponding decoded 5B vectors is shown in Table 4. The circuit T_r6decode illustrated in FIGS. 10A and 10B executes the transformations and classifications of the table.

Table 4. 6B/5B-T Decoding

Name	abcdei	Decoding Class	ABCDE K	DR Class	DR	DU Class	DU
D0	100101	b’•i•(a≠c)•(d≠e)	**0**0000 0		±		
D0	100101	b’•i•(a•c’•d•e’)	000**0**0 0		±		
D1	101001	a•b’•c•d’•e’•i	10**0**00 0		±		
D2	010011	a’•b•c’•d’•e•i	0100**0** 0		±		
D3	110001		11000 0		±		
D4	011001	a’•b•e’•i•(c≠d)	0**0**100 0		±		
D5P	101000		10100 0	PDRR6	+	NDUR6	–
D5A	010111	d•e•i	**10100** 0	NDRR6	–	PDUR6	+
D6P	011000		01100 0	PDRR6	+	NDUR6	–
D6A	100111	d•e•i	**01100** 0	NDRR6	–	PDUR6	+
D7P	111000		11100 0	NDRR6	–	NDUR6	–
D7A	000111	d•e•i	**11100** 0	PDRR6	+	PDUR6	+
D8	010101	a’•b•e’•i•(c≠d)	0**0**010 0		±		
D9P	100100		10010 0	PDRR6	+	NDUR6	–
D9A	011011	c•i•(a+b)•(d+e)	**10010** 0	NDRR6	–	PDUR6	+
D10P	010100		01010 0	PDRR6	+	NDUR6	–
D10A	101011	c•i•(a+b)•(d+e)	**01010** 0	NDRR6	–	PDUR6	+
D11	110100		11010 0		±		
D12P	001100		00110 0	PDRR6	+	NDUR6	–
D12A	110011	a•b•i•(c+d+e)	**00110** 0	NDRR6	–	PDUR6	+
D13	101100		10110 0		±		
D14	011100		01110 0		±		
D15	001101	b’•i•(a≠c)•(d≠e)	**1**1110 0		±		
D15	001101	a’•b’•c•i•(d≠e)	1**1**110 0		±		
D16	100011	b’•i•(a≠c)•(d≠e)	**0**0001 0		±		
D17P	100010		10001 0	PDRR6	+	NDUR6	–
D17A	011101	c•i•(a+b)•(d+e)	**10001** 0	NDRR6	–	PDUR6	+
D18P	010010		01001 0	PDRR6	+	NDUR6	–
D18A	101101	c•i•(a+b)•(d+e)	**01001** 0	NDRR6	–	PDUR6	+
D19	110010		11001 0		±		
D20P	001010		00101 0	PDRR6	+	NDUR6	–
D20A	110101	a•b•i•(c+d+e)	**00101** 0	NDRR6	–	PDUR6	+
D21	101010		10101 0		±		
D22	011010		01101 0		±		
D/K23P	111010		11101 x	NDRR6	–	PDUR6	+

Table 4. 6B/5B-T Decoding

Name	abcdei	Decoding Class	ABCDE K	DR Class	DR	DU Class	DU
D/K23A	000101	a'•b'•e'•(c'+d')	**11101** x	PDRR6	+	NDUR6	–
D24P	000110		00011 0	PDRR6	+	NDUR6	–
D24A	111001	a•b•i•(c+d+e)	**00011** 0	NDRR6	–	PDUR6	+
D25	100110		10011 0		±		
D26	010110		01011 0		±		
D/K27P	110110		11011 x	NDRR6	–	PDUR6	+
D/K27A	001001	a'•b'•e'•(c'+d')	**11011** x	PDRR6	+	NDUR6	–
D28	001110		00111 0		±		
D/K29P	101110		10111 x	NDRR6	–	PDUR6	+
D/K29A	010001	c'•d'•e'•(a'+b')	**10111** x	PDRR6	+	NDUR6	–
D/K30P	011110		01111 x	NDRR6	–	PDUR6	+
D/K30A	100001	c'•d'•e'•(a'+b')	**01111** x	PDRR6	+	NDUR6	–
D31	001011	b'•i•(a≠c)•(d≠e)	**1**1111 0		±		
D31	001011	a'•b'•c•i•(d≠e)	1**1**111 0		±		
D31	001011	b'•i•(a'•c•d'•e)	111**1**1 0		±		
K3P	110000	c'•d'•e'•i'	11000 **1**	PDRR6	+	NDUR6	–
K3A	001111	d•e•i	**11000** **1**	NDRR6	–	PDUR6	+

6B/5B-T Bit Decoding

The circuit for 6B/5B-T decoding following the equations below is shown in FIG. 10A

Inversion of the Five Leading Bits, CMPL5

A received vector is identified as an alternate vector, if the trailing bit 'i' has a value of one and either the vector is not balanced or all three trailing bits have a value of one. In this case, all the leading five bits are complemented. The complements of all valid vectors (primary vectors) falling into this category are illustrated in FIGS. 2, 3, and 4 and their true values are listed in the 'Alternate' column of Table 1.

It is assumed that invalid vectors with five or six ones originated from vectors with four ones and they will be complemented as though the extra ones were not present, i.e. it is not necessary to include the zeros in the Boolean expressions for the 11 vectors below with positive disparity.

D5A	0 1 0 **1 1 1**	D12A	**1 1** 0 0 1 **1**	D9A	0 1 **1** 0 1 **1**
D6A	1 0 0 **1 1 1**	D20A	**1 1** 0 1 0 **1**	D10A	1 0 **1** 0 1 **1**
D7A	0 0 0 **1 1 1**	D24A	**1 1** 1 0 0 **1**	D17A	0 1 **1** 1 0 **1**
K3A	0 0 1 **1 1 1**			D18A	1 0 **1** 1 0 **1**

Similarly, it is assumed that invalid vectors with five or six zeros originated from vectors with four zeros and they will be complemented as though the extra zeros were not present, i.e. it is not necessary to include the ones in the Boolean expressions for the four vectors below with negative disparity.

D/K23A	**0 0** 0 1 **0** 1	D/K29A	0 1 **0 0 0** 1
D/K27A	**0 0** 1 0 **0** 1	D/K30A	1 0 **0 0 0** 1

A Boolean expression CMPL5 for the complementation of the leading five bits of the fifteen 6B vectors above is developed below:

$$CMPL5 = d \bullet e \bullet i + a \bullet b \bullet i \bullet (c+d+e) + c \bullet i \bullet (a+b) \bullet (d+e) + a' \bullet b' \bullet e' \bullet (c'+d') + c' \bullet d' \bullet e' \bullet (a'+b')$$

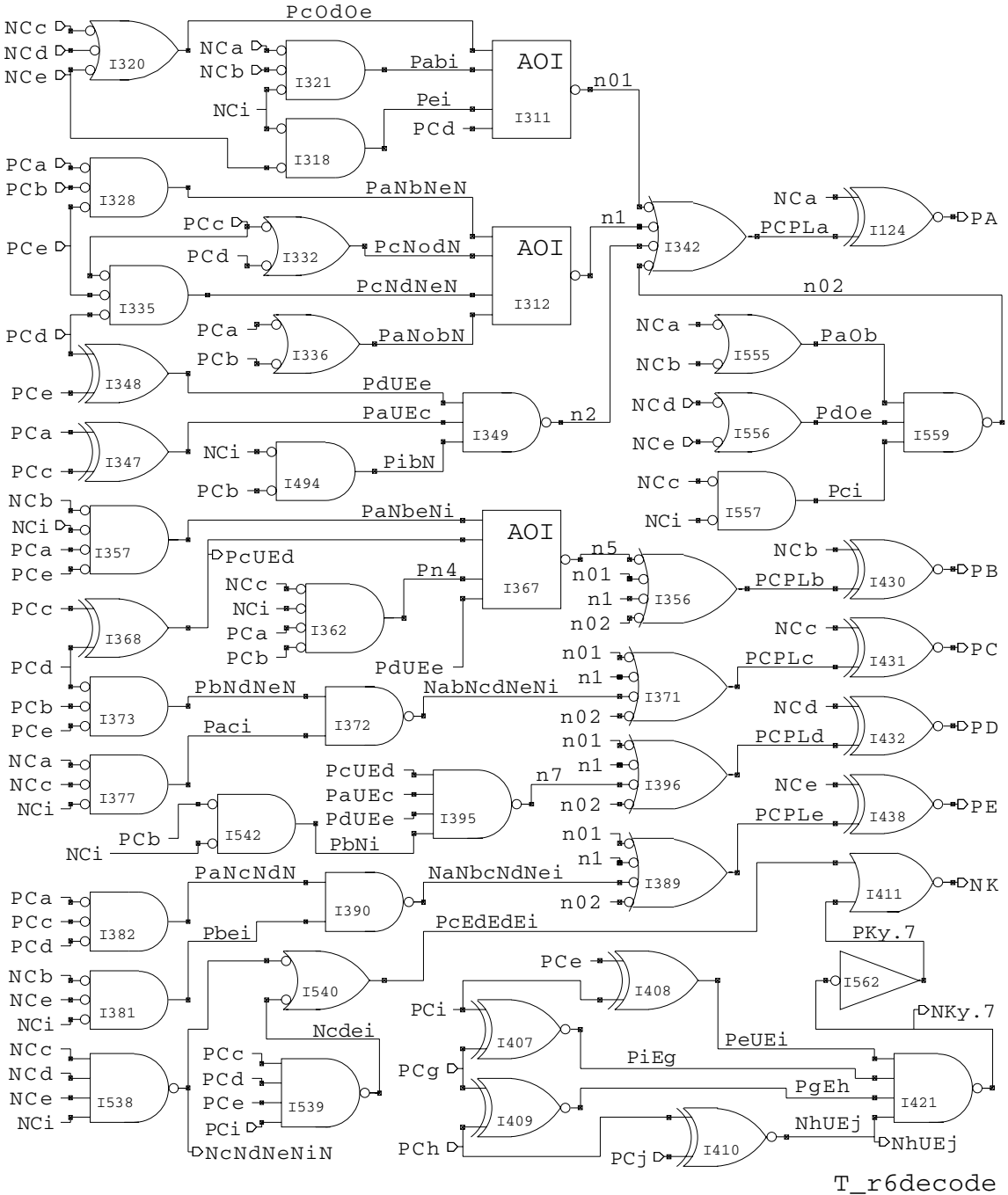


FIG. 10A. 6B/5B-T DECODER

If the control characters of Table 3B are included, the term +g•h•j must be added to CMPL5. It is recommended to expand this to +f'• g•h•j to reduce error spreading.

In the circuit diagram FIG. 10A, the following net names are used:

n01 = a•b•i•(c+d+e) + d•e•i n02 = c•i•(a+b)•(d+e)

n1 = a'•b'•e'•(c'+d') + c'•d'•e'•(a'+b')

The signal COMPL5 = n01 + n02 + n1 does not appear explicitly in the circuit, because the OR function is merged with the complementation function specific to a particular bit position.

Selected Bit Inversions

There are nine disparity independent primary vectors with an i-bit value of 1 as shown in FIG. 1B. For all of these, one or more bits in the leading 5 positions must be changed for decoding.

Bit A

Table 4 shows for which of these vectors, the first bit must be complemented:

D0 1 **0** 0 1 0 **1**

D15 0 **0** 1 1 0 **1**

D16 1 **0** 0 0 1 **1**

D31 0 **0** 1 0 1 **1**

CMPLa = CMPL5 + b'•i•(a≠c)•(d≠e)

n2 = b'•i•(a≠c)•(d≠e)

Bit B

D4 **0 1** 1 0 **0 1** D15 **0 0 1** 1 0 **1**

D8 **0 1** 0 1 **0 1** D31 **0 0 1** 0 1 **1**

CMPLb = CMPL5 + a'•b•e'•i•(c≠d) + a'•b'•c•i•(d≠e)

n4 = a'•b'•c•i

n5 = a'•b•e'•i•(c≠d) + a'•b'•c•i•(d≠e)

Bit C

D1 1 0 1 0 0 1

CMPLc = CMPL5 + a•b'•c•d'•e'•i

Bit D

D0 1 **0** 0 1 0 **1**

D31 0 **0** 1 0 1 **1**

CMPLd = CMPL5 + b'•i•(a≠c)•(c≠d)•(d≠e)

$n7 = b' \bullet i \bullet (a \neq c) \bullet (c \neq d) \bullet (d \neq e)$

Bit E

D2 0 1 0 0 1 1

$CMPLe = CMPL5 + a' \bullet b \bullet c' \bullet d' \bullet e \bullet i$

Bit K

The value of K is one for any bit pattern listed in Table 3A. K3.x is the set of the first eight characters where x is any number from 0 to 7: $K3.x = (c=d=e=i)$. Ky.7 stands for the ten vectors of the Tables 3A and 3B with $ei=10$ (K23, K27, K29, K30, K19, K21, K22, K25, K26, or K28): $Ky.7 = (e \neq i) \bullet (i=g=h=j)$

$K = (c=d=e=i) + (e \neq i) \bullet (i=g=h=j)$

The above equation is implemented as follows, because the term $(c' \bullet d' \bullet e' \bullet i')$ is also required in the 4B/3B decoder:

$K = c \bullet d \bullet e \bullet i + c' \bullet d' \bullet e' \bullet i' + (e \neq i) \bullet (i=g) \bullet (g=h) \bullet (h=j)$

If the last six control characters of Table 3B are not included in the valid set, they are simply declared invalid by some expanded expression described further down. If K11.7 is also included, the term $+c=e=i=g=h=j$ must be appended to the expression for K.

6B Validity and Disparity Checks

The circuit of FIG. 10B checks the validity of the 6B vectors and checks whether there are any disparity violations at the front or back end. The circuit also determines the exit disparity of the disparity dependent 6B vectors for the use by other checking circuits of the 4B/3B-T decoder shown in FIG. 11.

Logic Equation for invalid Vectors R6, INVR6

There are a total of 16 invalid R6 vectors:

$INVR6 = a \bullet b \bullet c \bullet d + a' \bullet b' \bullet c' \bullet d' + P3x \bullet e \bullet i + Px3 \bullet e' \bullet i'$

$P3x = P31 + P40 = a \bullet b \bullet c + a \bullet b \bullet d + a \bullet c \bullet d + b \bullet c \bullet d$

$Px3 = P13 + P04 = a' \bullet b' \bullet c' + a' \bullet b' \bullet d' + a' \bullet c' \bullet d' + b' \bullet c' \bullet d'$

The column 'DR' of Table 4 lists the required disparity at the start of the respective 6B vectors and the column 'DR Class' identifies the respective input bit patterns. The old column DB of Ref. 1 and 2 has been changed to DU as was done for the decoder of Ref. 3 and lists a positive or negative exit disparity for the disparity dependent vectors only. Disparity independent vectors have no entry in the DU column. In the old design, disparity independent vectors passed the input disparity to the output. In the new design, disparity independent vectors are ignored and bypassed for disparity purposes for shorter delay. Short delay is especially important for the DU outputs PDUR6 and NDUR6. To achieve

this goal, a dedicated column ‘DU Class’ has been added which sorts the received vectors into DU classes in the most efficient way.

Logic Equations for Required Input Disparity DRR6

The terms PDRR6 and NDRR6 represent the R6 vectors which require a positive or negative running disparity, respectively, at the start of the vector.

All invalid vectors with five or six zeros are also assigned a positive required entry disparity. Therefore, any vector with three leading zeros or four or more zeros requires a positive entry disparity. The valid vectors with *positive* required front end disparity are listed below.

D7A	**0 0 0** 1 1 1	D5P	1 0 1 **0 0 0**	D9P	1 0 0 1 0 0
D/K23A	**0 0 0** 1 0 1	D6P	0 1 1 **0 0 0**	D10P	0 1 0 1 0 0
D24P	**0 0 0** 1 1 0	K3P	1 1 0 **0 0 0**	D12P	0 0 1 1 0 0
				D17P	1 0 0 0 1 0
				D18P	0 1 0 0 1 0
				D20P	0 0 1 0 1 0
				D/K27A	0 0 1 0 0 1
				D/K29A	0 1 0 0 0 1
				D/K30A	1 0 0 0 0 1

Making allowance for invalid vectors, the three vectors in the left column can be identified by the Boolean expression a’•b’•c’. The three vectors in the center column are identified by d’•e’•i’•(a•b•c)’. The nine vectors in the right column have two zeros in both the leading and trailing three bit positions. Because of the inclusion of the vectors with more than four zeros, all the one bits can be ignored and the remaining vectors (valid or invalid) can be identified by the expression:

(a’•b’+a’•c’+b’•c’) • (d’•e’+d’•i’+e’•i’)

Therefore:

PDRR6 = a’•b’•c’ + d’•e’•i’•(a•b•c)’ + (a’•b’+a’•c’+b’•c’) • (d’•e’+d’•i’+e’•i’)

In the circuit diagram DES1B, the following net names are used:

n9 = (a’•b’+a’•c’+b’•c’)

n10 = (d’•e’+d’•i’+e’•i’)

n11 = n9 • n10

n12 = d’•e’•i’•(a•b•c)’

All invalid vectors with five or six ones are also assigned a negative required entry disparity. Therefore, any vector with three leading ones or four or more ones requires a

negative entry disparity. The vectors with *negative* required front end disparity are listed below.

D7P	**1 1 1** 0 0 0	D5A	0 1 0 **1 1 1**	D9A	0 1 1 0 1 1
D/K23P	**1 1 1** 0 1 0	D6A	1 0 0 **1 1 1**	D10A	1 0 1 0 1 1
D24A	**1 1 1** 0 0 1	K3A	0 0 1 **1 1 1**	D12A	1 1 0 0 1 1
				D17A	0 1 1 1 0 1
				D18A	1 0 1 1 0 1
				D20A	1 1 0 1 0 1
				D/K27P	1 1 0 1 1 0
				D/K29P	1 0 1 1 1 0
				D/K30P	0 1 1 1 1 0

The three vectors D7P, D/K23P, and D24A in the left column above can be identified by the Boolean expression a•b•c. The three vectors in the center column are identified by d•e•i•(a+b+c). The nine vectors in the right column are identified by the expression: (a•b+a•c+b•c) • (d•e+d•i+e•i).

Therefore:

NDRR6 = a•b•c + d•e•i•(a+b+c) + (a•b+a•c+b•c) • (d•e+d•i+e•i)

n13 = (a•b+a•c+b•c)

n14 = (d•e+d•i+e•i)

n15 = d•e•i•(a+b+c)

n16 = n13 • n14

Logic Equation for Monitoring Byte Disparity Violations DVBY

Bytes with only disparity independent vectors R6 and R4 are ignored for disparity checking purposes. We have a disparity violation DVBY at a specific byte under the following conditions:

1. The required entry disparity of the R6 vector does not match the running disparity at the front of the byte.

   PDFBY•NDRR6 + NDFBY•PDRR6

2. The required entry disparity of the R4 vector does not match the running disparity in front of the byte *and* the R6 vector does *not* have a required entry disparity which is the complement of the required entry disparity for R4.

   n17 = NDRR4 • PDFBY • PDRR6’

   n18 = PDRR4 • NDFBY • NDRR6’

A disparity violation internal to a byte from a disparity dependent R4 vector mismatched to a disparity dependent R6 vector is included in the set of invalid bytes, but not in DVBY. The disparity violation at a byte DVBY is thus given by the equation:

DVBY = PDFBY•NDRR6 + NDFBY•PDRR6 + n17 + n18

The terms PDFBY and NDFBY represent a positive or negative running disparity, respectively, at the front of the byte and one or the other function is always true. However, for PDRR6 and NDRR6, none of the functions is true for the case of most balanced vectors.

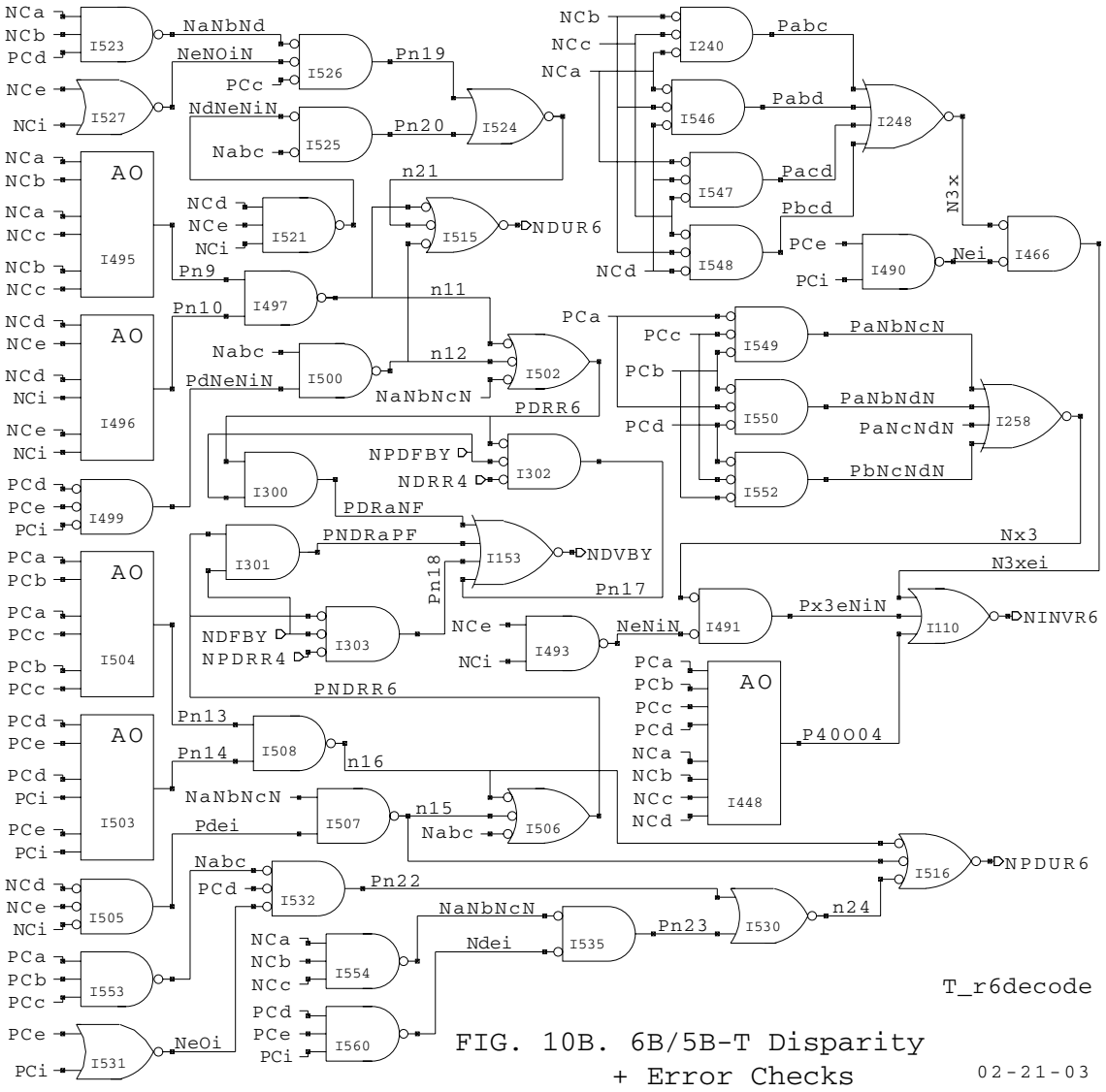


FIG. 10B. 6B/5B-T Disparity + Error Checks

02-21-03

Logic Equations for the assumed ending Disparities PDUR6 and NDUR6

Four leading ones or zeros in the encoded domain are invalid vectors and can be generated only by at least one error. For the case of a single error and e=i, the R6 vector was obviously one of the initially balanced vectors 011100, 101100, 110100, 111000, or their complement. Of these, all except 111000 and 000111 should not generate PDUR6 or NDUR6 which would generate a superfluous code violation at the next disparity dependent vector in addition to the invalid vector at the actual error location.

Therefore, the signal **PDUR6** should be asserted in response to the following 6B inputs:

1. All bits with a value of one (1).
2. Five bits with a value of one (6).
3. Four bits with a value of one except the pattern ‘111100’ (14).
4. The pattern ‘000111’

Again, the list of these vectors is almost identical to the list above for NDRR6 except that the pattern for D7 is the complement, i.e. 000111, so the left column looks as follows:

D7A **0 0 0** 1 1 1

D/K23P **1 1 1 0** 1 0

D24A **1 1 1 0** 0 1

The three vectors above are defined by the expression n24. The following net name abbreviations are used in the circuit diagram DES1B:

n22 = a•b•c•d’•(e+i) *n23 = a’•b’•c’•d•e•i* *n24 = n22 + n23*

The term n24 replaces the term a•b•c in the NDRR6 equation, therefore:

PDUR6 = n24 + d•e•i•(a+b+c) + (a•b+a•c+b•c) • (d•e+d•i+e•i)

PDUR6 = n24 + n1 + n16

The signal **NDUR6** should be asserted in response to the following 6B inputs:

1. All bits with a value of zero (1).
2. Five bits with a value of zero (6).
3. Four bits with a value of zero except the pattern ‘000011’ (14).
4. The pattern ‘111000’

The list of these vectors is almost identical to the list above for PDRR6 except that the pattern for D7 is the complement, i.e. 111000, so the modified left column looks as follows:

D7P **1 1 1** 0 0 0

D/K23A **0 0 0 1** 0 1

D24P **0 0 0 1** 1 0

The vectors D/K23A and D24P above are defined by the expression:

n19 = a’•b’•c’•d•(e’+i’)

The vector D7P is defined by the expression:

n20 = a•b•c•d’•e’•i’

The following net name abbreviation is used in the circuit diagram of FIG.10B:

n21 = n19 + n20

The expression for n21 replaces the term a'•b'•c' in the PDRR6 equation, therefore:

NDUR6 = n21 + d'•e'•i'•(a•b•c)' + (a'•b'+a'•c'+b'•c') • (d'•e'+d'•i'+e'•i')

NDUR6 = n21 + n12 + n11

Circuit Simplification

The first term (D7) in each of the equations for PDUR6 and NDUR6 prevents double counts for some type of errors. However, overall some double counts are unavoidable and the added term improves the accuracy of the error count by a minuscule amount. It is debatable whether it should be dropped. A further more significant simplification is possible if the first vector (D7) is also dropped from PDRR6 and NDRR6. This may delay the error detection for some patterns by a very few bytes but does not degrade error detection per se. The circuit advantage is that with these two simplifications PDRR6 is equal to NDUR6, and NDRR6 is equal to PDUR6. Analogous simplifications are possible for the 4B/3B error detection, but there the circuit simplification is less compelling. The circuit of FIG. 10B does not use these simplifications.

4B/3B-T Decoder, Table 5

Table 5 shows the relationships between the coded 4B vectors and the corresponding decoded 3B vectors. The circuit T_r4decode illustrated in FIG. 11 executes the transformations and classifications of the table.

Table 5. 4B/3B-T Decoding

Name	f g h j	Decoding Class	FGH K[1]	DR Class	DR	DU Class	DU
Dx/K3.0P	0101	m5	0**0**0 x		±		
K3.0A	1010	m0, m0	**0**0**0** **1**				
Dx/K3.1P	1001		100 x		±		
K3.1A	0110	m0, m0•(f•h)', m0	**100** **1**				
Dx/K3.2P	0100		010 x	PDR4	+		–
Dx/K3.2A	1011	m2, m5, g'•h•j	**010** x	NDR4	–		+
Dx/K3.3P	1100		110 x	NDR4	–		–
Dx/K3.3A	0011	m2, m5, g'•h•j	**110** x	PDR4	+		+
Dx/K3.4P	0010		001 x	PDR4	+		–
Dx/K3.4A	1101	m1, m5, m1	**001** x	NDR4	–		+
Dx/K3.5P	1010		101 x		±		
K3.5A	0101	m0, m3, m0	**101** **1**				
Dx/K3.6P	0110		011 x		±		
K3.6A	1001	m0, m3, m0	**011** **1**				
Dx/K3.7P	1110		111 x	NDR4	–		+
Dx/K3.7A	0001	m1, m4, m1	**111** x	PDR4	+		–
Dx/Kx.7P	0111	m2	**1**11 x	NDR4	–		+
Dx/Kx.7A	1000	m4, m4	1**11** x	PDR4	+		–

[1] Kx.7 = (e≠i)•(i=g=h=j)

Logic Equations for the Generation of the decoded Bits F, G, H, K

Generally, F = f, G = g, H = h, except for the conditions listed below for which the complement of the respective unencoded bit is generated, e.g. H = h'. No changes are required for the inclusion of the control characters of Table 3B.

Bit F

For 4B/3B decoding, the f-bit is complemented for the vectors listed below. For the four vectors in the left column, complementation is applicable only if the vectors are preceded by K3 with negative ending disparity, i.e. if c'•d'•e'•i' is true.

K3.0A	1 0 1 0	Dx/K3.2A	1 0 **1** **1**	Dx/K3.7A	0 0 **0** **1**
K3.1A	0 1 1 0	Dx/K3.3A	0 0 **1** **1**	Dx/K3.4A	1 1 **0** **1**
K3.5A	0 1 0 1	Dx/Kx.7P	0 1 **1** **1**		
K3.6A	1 0 0 1				

- The left column can be characterized by *m0 = (c'•d'•e'•i')•(f≠g)•(h≠j).*
- The center column can be characterized by *m2 = h•j•(f•g)'.*
- The right column can be characterized by *m1 = (f=g)•h'•j.*

The Boolean expression CMPLf to complement the f-bit is thus:

CMPLf = (c'•d'•e'•i')•(f≠g)•(h≠j) + h•j•(f•g)' + (f=g)•h'•j = m0 + m1 + m2

Bit G

For 4B/3B decoding, the g-bit is complemented for the vectors listed below. For the three vectors in the left column, complementation is applicable only if the vectors are preceded by K3 with negative ending disparity, i.e. if c'•d'•e'•i' is true.

K3.1A	0 1 1 0	Dx/K3.2A	1 **0** **1** **1**	Dx/K3.7A	0 **0** **0** 1
K3.5A	0 1 0 1	Dx/K3.3A	0 **0** **1** **1**	Dx/Kx.7A	1 **0** **0** 0
K3.6A	1 0 0 1	Dx/K3.0P	0 **1** **0** **1**		
		Dx/K3.4A	1 **1** **0** **1**		

- The left column can be characterized by *(c'•d'•e'•i')•(f≠g)•(h≠j)•(f•h)' = m0•(f•h)' = m3.*
- The center column can be characterized by *m5 = g'•h•j + g•h'•j.*
- The right column can be characterized by *m4 = (f≠j)•g'•h'.*

The Boolean expression CMPLg to complement the f-bit is thus:

CMPLg = (c'•d'•e'•i')•(f≠g)•(h≠j)•(f•h)' + g'•h•j + g•h'•j + (f≠j)•g'•h'

CMPLg = m3 + m4 + m5

Bit H

For 4B/3B decoding, the h-bit is complemented for the vectors listed below. For the four vectors in the left column, complementation is applicable only if the vectors are preceded by K3 with negative ending disparity, i.e. if c'•d'•e'•i' is true.

K3.0A	1 0 1 0	Dx/K3.2A	1 **0 1 1**	Dx/K3.7A	0 0 **0 1**
K3.1A	0 1 1 0	Dx/K3.3A	0 **0 1 1**	Dx/K3.4A	1 1 **0 1**
K3.5A	0 1 0 1			Dx/Kx.7A	1 **0 0** 0
K3.6A	1 0 0 1			*Dx/K3.7A	0 **0 0** 1

- The left column is identical to what is listed under Bit F above and can be characterized by *m0 = (c'•d'•e'•i')•(f≠g)•(h≠j)*.
- The center column can be characterized by g'•h•j.
- The right column can be characterized by *(f=g)•h'•j + (f≠j)•g'•h' = m1 + m4*, where the second term includes Dx/K3.7 redundantly to reuse an expression already available from bit g decoding.

The Boolean expression CMPLh to complement the h-bit is thus:

CMPLh = (c'•d'•e'•i')•(f≠g)•(h≠j) + g'•h•j + (f=g)•h'•j + (f≠j)•g'•h'

CMPLh = m0 + m1 + m4 + g'•h•j

Bit K

The derivation of the K-bit has been described above in the section on 6B/5B decoding. A single K-bit is applicable to the entire byte. While K3 can be recognized as a control vector from the value of the 6B vector alone, all other control characters can be recognized only by an examination of the combination of a 6B and a 4B vector.

Logic Equations for the required Disparity at the Front of the R4 Vector

The terms PDRR4 and NDRR4 represent the required positive or negative disparity, respectively, at the front of the R4 vector.

A total of six 4B vectors require a positive disparity PDRR4 at the front:

1. All four bits have a zero value which is an invalid vector.
2. There is a single one bit in the 4-bit vector.
3. The vector 0011.

The first and the third condition are met by f'•g' which also overlaps the second condition. The second condition is met by (f•g)'•h'•j'. Therefore,

PDRR4 = f'•g' + (f•g)'•h'•j'

A total of six 4B vectors require a negative disparity NDRR4 at the front:

1. All four bits have a one value which is an invalid vector.

2. There is a single zero bit in the 4-bit vector.
3. The vector 1100.

$NDRR4 = f•g + (f+g)•h•j$

Logic Equations for the assumed ending Disparities PDUR4 and NDUR4

A total of six 4B vectors have a positive ending disparity PDUR4:

1. All four bits have a one value which is an invalid vector.
2. There is a single zero bit in the 4-bit vector.
3. The vector 0011.

$PDUR4 = h•j + f•g•(h+j)$

A total of six 4B vectors have a negative ending disparity NDUR4:

1. All four bits have a zero value which is an invalid vector.
2. There is a single one bit in the 4-bit vector.
3. The vector 1100.

$NDUR4 = h'•j' + f'•g'•(h•j)'$

Logic Equation for invalid Vector R4, INVR4

There are a total of two inherently invalid R4 vectors: all ones or all zeros (f=g=h=j). Some combinations of an R6 vector with the primary or alternate version of FGH= 111 encoding can be recognized as invalid and can be generated by errors. A first error involves an actual false comma sequence cdeifgh = 1x00000 or 0x11111 can be detected by the expression c≠e=i=f=g=h. A second error involves a potential for a false comma sequence across the trailing end such as ifghj = 1x000 pr 0x111 and can be detected by the expression i≠g=h=j. These invalid bit patterns are lumped together in the signal INVR4:

$INVR4 = (f=g=h=j) + (c≠e=i=f=g=h) + (i≠g=h=j)$

Below the net names Ky.7 and Kz.7 are used. The letter 'y' stands for the ten values in the 'Name' column of the Tables 3A and 3B with ei=10 (K23, K27, K29, K30, K19, K21, K22, K25, K26, or K28), the letter 'z' is restricted to the seven values listed in Table 3B (K11, K19, K21, K22, K25, K26, or K28). The Kz.7 control characters are not implemented in the circuit designs shown and therefore classified as invalid characters.

The first character K11.7 can be identified by c=e=i=g=h=j. The values of the f-bit and the d-bit can be ignored for this purpose, because either value generates an invalid character detected by this or the other expressions.

The R6 part of the last six characters of Table 3B have complementary bits in the last two positions, the same as the last four control characters of Table 3A. However, the last six characters of Table 3B have two ones and two zeros in the first four bit positions and the other four characters do not. So these six control characters can be identified by

(i=g=h=j) • (e≠i)•P22, where P22 = (a≠b)•(c≠d) + (a=b)•(c=d)•(b≠c).

So all seven control characters Kz.7 of Table 3B can be identified by:

Kz.7 = (c=e=i=g=h=j) + (i=g=h=j) • (e≠i)•P22

If the required entry disparity DRR4 does not match the exit disparity DUR6 of a disparity dependent R6 vector, an invalid character is flagged by the signal DV64 under the following conditions:

DV64 = PDRR4 • NDUR6 + NDRR4 • PDUR6

The signal INVR4V64 is the OR function of INVR4 and DV64.

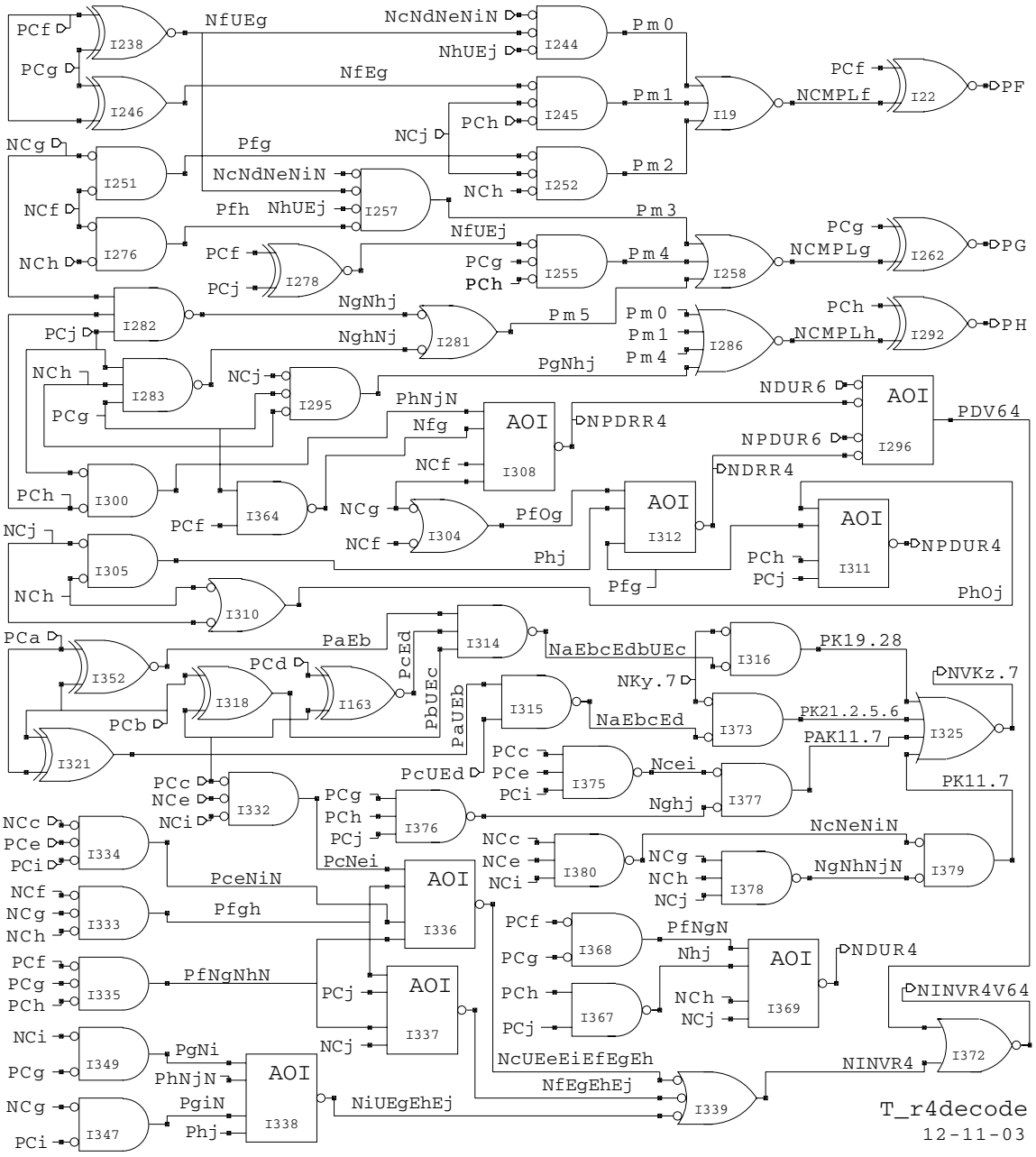


FIG. 11. 4B/3B-T DECODER, Error Checks

Circuit Simplifications

Some of the above error and validity checks can be simplified or eliminated without great harm, depending on the application which often provides other means to detect the same errors. Some simplifications just postpone error detection to a succeeding byte. Simplification should be considered primarily if the checks are in the critical timing path and are the major reason for complicated measures to work around it.

10B/8B-T Decoder, Error Checks

The circuit T_r10decode shown in FIG. 12 merges the 6B/5B and 4B/3B decoders into a byte decoder and monitors errors at the byte level based on outputs from the two circuits T_r6cecode and T_r4decode described above. The circuit also monitors whether the running disparity at the front end of the byte matches the required entry disparity and it establishes a running disparity value at the tail end of bytes with disparity dependent component vectors.

Byte Validity Check

This circuit generates the signal INVBY which indicates an inherently invalid byte which includes disparity violations DV64 which are evident from an examination of just the 10 coded bits of the current byte. The signal VIOL signals either an invalid byte or a disparity violation DVBY detected at this location which may result from an error in this or a preceding byte. The output INVBY is provided for support of certain error correction techniques which require the identification of the erroneous byte. The circuit delays for DVBY and VIOL are the longest for the decoder. Fortunately, for many applications, these outputs can be reported during the next byte cycle with no adverse impact, i.e. pipe-lining limited to these circuits is possible.

Updating of Running Disparity at Tail End

If either one or both of the vectors are disparity dependent, either PDUBY or PNDUBY are asserted to establish a positive or negative running disparity, respectively, at the end of the byte:

$$PDUBY = PDUR4 + PDUR6 \bullet NDUR4'$$

$$NDUBY = NDUR4 + NDUR6 \bullet PDUR4'$$

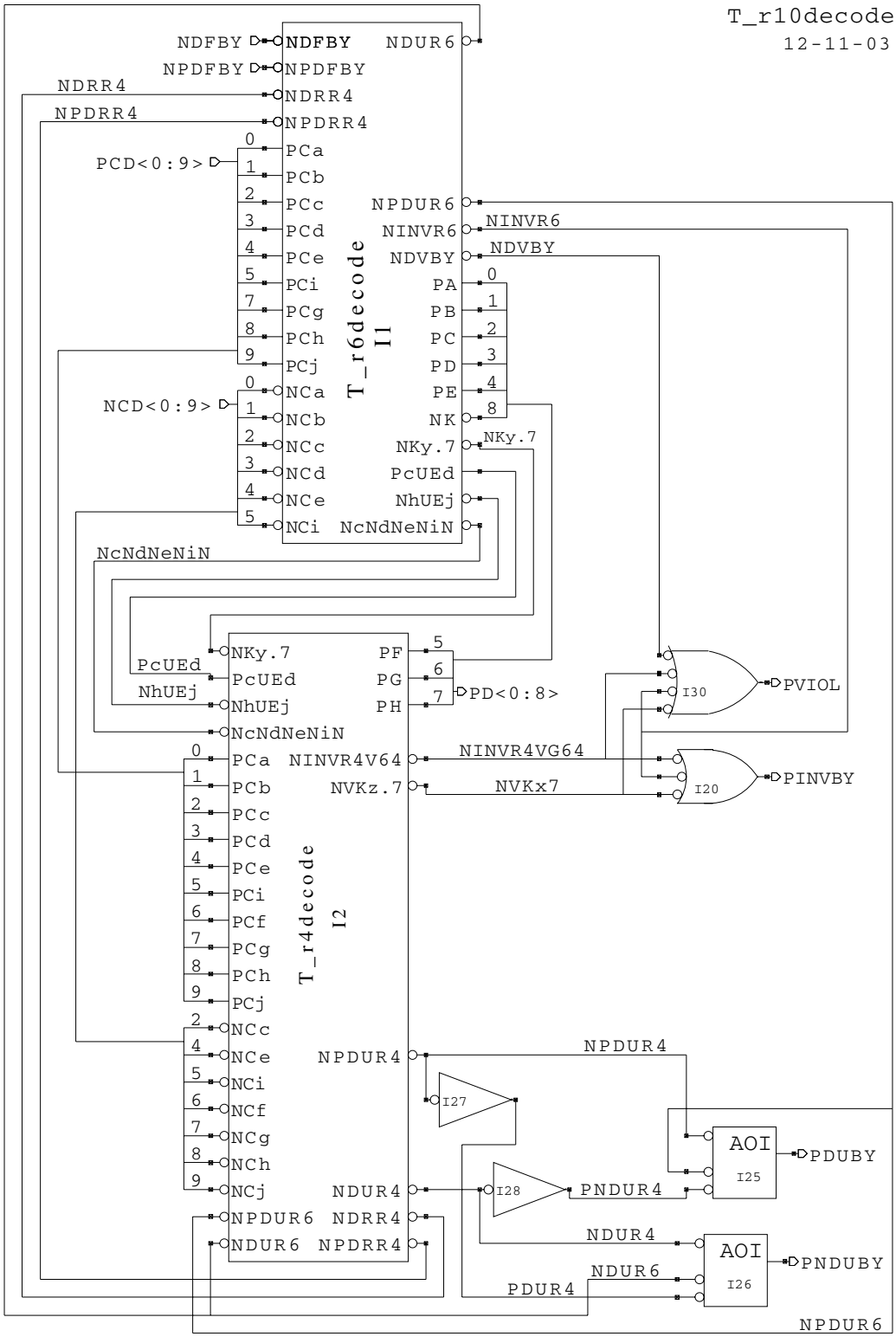


FIG. 12. 10B/8B-T DECODER, ERROR CHECKS

Disparity Operations

Similar to the encoding side, examples of circuits for the updating of the running disparity are shown in a faster pipe-lined version and a simpler slower version for a single byte and for four parallel bytes originating from a single serial link.

Single Byte Disparity, Fast Version, FIG. 13

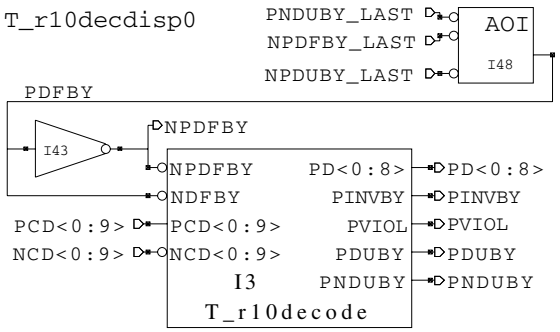


FIG. 13. 10B/8B-T BYTE DISPARITY FAST Version 07-18-02

The circuit shown in FIG. 13 uses limited pipe-lining to establish the running disparity at the front of the byte.

Notation: The signal names PDFBY and NDFBY refer to a positive and negative running disparity, respectively, in front of the byte. The signal names PDUBY and NDUBY refer to the assumed positive or negative exit disparity, respectively, of the byte, regardless of the starting disparity at the front end. If neither the 6B vector nor the 4B vector of the byte is disparity dependent, none of the two outputs is asserted.

The values for the outputs NPDFBY, PDUBY, and PNDUBY are stored in three latches (not shown in the diagrams) which are clocked concurrently with the decoded data output. The outputs of the latches are labelled NPDFBY_LAST, NPDUBY_LAST (from pin L2N, the inverted output of the slave latch L2), and PNDUBY_LAST, respectively, and are used for the computation of the starting disparity PDFBY for the next byte.

Logic Equations for the Determination of the Disparity at the Start of the Byte

$$PDFBY = PDUBY\_LAST + PDFBY\_LAST \bullet NDUBY\_LAST'$$

Note that NDFBY and PDFBY are complementary: NDFBY = PDFBY' and the values of PDUBY and NDUBY are exclusive, none or one alone can be true.

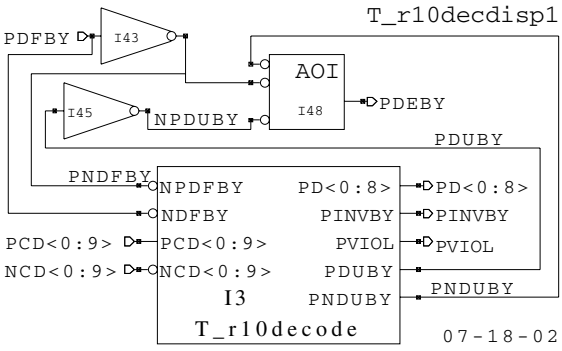


FIG. 14. 10B/8B-T BYTE DISPARITY, Slower Version

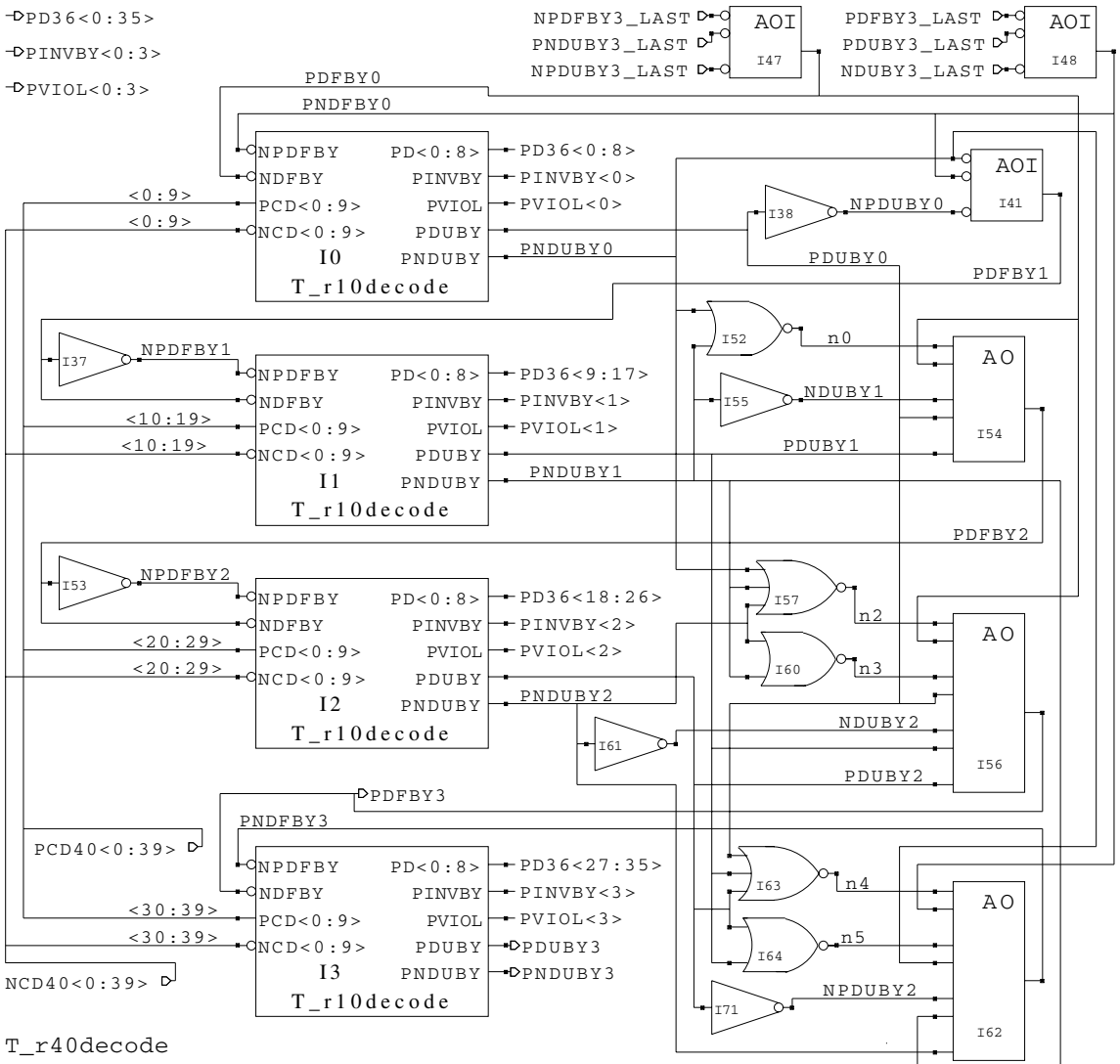
Single Byte Disparity, Slower Version, FIG. 14

The incentive to use the slower version of FIG. 14 is the saving of two latches. If timing is not critical, the ending disparity PDEBY is generated in the same cycle as the decoding and the error checks. So only this single parameter must be passed on to next byte in the traditional manner. The output of this latch is the signal PDFBY, the disparity in front of the new byte. If the longest delay

path is to the PDEBY output, we have then added to critical path delays associated with one inverter plus one OAI21 gate.

Four-Byte Word Decoder, FIG. 15

The circuit of FIG. 15 comprises four parallel encoder circuits T_r10decode and the circuits required to establish a value for the running disparity at the input of each of the four decoders.



12-11-03 FIG. 15. 10B/8B-T FOUR-BYTE DECODER

Notation

The signal names PDFBY0 and NDFBY0 refer to a positive and negative disparity, respectively, in front of byte #0.

The signal names PDUBY0 and NDUBY0 refer the assumed positive or negative exit disparity, respectively, of byte #0. If neither the 6B vector nor the 4B vector of the byte is disparity dependent, none of the two outputs is asserted.

The values for the outputs PDFBY3, PDUBY3, and PNDUBY3 are stored in latches with complementary outputs PDFBY3_LAST/NPDFBY3_LAST, PDUBY3_LAST/ NPDUBY3_LAST, and PNDUBY3_LAST/NPNDUBY3_LAST, respectively, for the computation of the starting disparities PDFBY0 and PNDFBY0 for the next word cycle at the top of the diagram.

Logic Equations for the Determination of the Disparity at the Start of the Bytes

$$PDFBY0 = PDUBY3\_LAST + PDFBY3\_LAST \bullet NDUBY3\_LAST'$$

$$NDFBY0 = NDUBY3\_LAST + NDFBY3\_LAST \bullet PDUBY3\_LAST'$$

The values of PDFBY0 and NDFBY0 are complementary, but the values of PDUBY3 and NDUBY3 are exclusive, none or one alone can be true.

To minimize the circuit delays, the disparity values for the front of byte #1, #2, and #3 are determined not sequentially from byte to byte, but based on the disparity in front of byte #0 and the changes in disparity contributed by the byte(s) in between.

$$PDFBY1 = PDUBY0 + PDFB0 \bullet NDUB0'$$

$$NDFBY1 = NDUBY0 + NDFB0 \bullet PDUBY0'$$

$$PDFBY2 = PDUBY1 + PDUB0 \bullet NDUB1' + PDFBY0 \bullet NDUBY0' \bullet NDUBY1'$$

$$n0 = NDUBY0 + NDUBY1$$

$$PDFBY2 = PDUBY1 + PDUB0 \bullet NDUB1' + PDFBY0 \bullet n0'$$

$$NDFBY2 = NDUBY1 + NDUBY0 \bullet PDUBY1' + NDFBY0 \bullet PDUBY0' \bullet PDUBY1'$$

$$n1 = PDUBY0 + PDUBY1$$

$$NDFBY2 = NDUBY1 + NDUBY0 \bullet PDUBY1' + NDFBY0 \bullet n1'$$

$$PDFBY3 = PDUBY2 + PDUBY1 \bullet NDUBY2' + PDUBY0 \bullet NDUBY1' \bullet NDUBY2' + PDFBY0 \bullet NDUBY0' \bullet NDUBY1' \bullet NDUBY2'$$

$$n2 = NDUBY0 + NDUBY1 + NDUBY2$$

$$n3 = NDUBY1 + NDUBY2$$

$$PDFBY3 = PDUBY2 + PDUBY1 \bullet NDUBY2' + PDUBY0 \bullet n3' + PDFBY0 \bullet n2'$$

$$NDFBY3 = NDUBY2 + NDUBY1 \bullet PDUBY2' + NDUBY0 \bullet PDUBY1' \bullet PDUBY2' + NDFBY0 \bullet PDUBY0' \bullet PDUBY1' \bullet PDUBY2'$$

$$n4 = PDUBY0 + PDUBY1 + PDUBY2$$

$$n5 = PDUBY1 + PDUBY2$$

$$NDFBY3 = NDUBY2 + NDUBY1 \bullet PDUBY2' + NDUBY0 \bullet n5' + NDFBY0 \bullet n4'$$

In the circuit implementation of FIG.15, we take advantage of the relationships:

PNDFBY1 = PDFBY1' *PNDFBY2 = PDFBY2'*

These signals are not in the critical path and the added inversion does not decrease the maximum rate. For applications which have sufficient timing margin, the same simplifications can also be used for the signal PNDFBY3 and perhaps PNDFBY0 at a penalty of one inversion for each of those two signals.

5B/6B-T Code and Partitioned 10B/12B-T Code

Some applications are compatible with modulo 5-bit data units. Plain data can be transmitted in modulo five format by just concatenating 6B vectors as defined above. However, in the 6-bit domain there is no comma character defined. Synchronization must be accomplished by other means. To gain access to comma characters, the code must be viewed as a 10B/12B code, but actual operation can easily shift from one mode to the other. A compatible partitioned 10B/12B-T transmission code can be constructed from concatenated pairs of 6B vectors with identical definitions as defined in Table 1. The only difference between a 5B/6B-T and the 10B/12B-T code is the availability of a comma and a larger set of control characters in the 12B-T format. For purposes of special, non-data characters, for example a comma, the 5B/6B-T code is expanded to a 10B/12B-T code using a pair of contiguous 6B vectors (2 x 5B/6B = 10B/12B). For an example, refer to FIG. 16, where ($a_0b_0c_0d_0e_0i_0$) and ($a_1b_1c_1d_1e_1i_1$) are needed to define these characters. The *comma-character* is a special character of interest. It can be used to mark and recover quickly, and to monitor the 6B, 12 B, and packet boundary alignments for each individual link in a parallel bus configuration regardless of the skew of the several transmission lanes.

Keeping in mind that for data neither the first four bits nor the last four bits of a 6B vector can be identical, the 5B/6B-T and the 10B/12B-T codes can easily be evaluated with the help of the trellis diagram of FIG. 6 and its characteristics can be summarized as follows:

1. The maximum run length RL is five with at most two contiguous runs of five in data mode, e.g, +0111**00+000111+11**0001+. Three contiguous runs of five are generated if the first two 6B vectors of the above example are followed by the comma character or certain other control characters: +0111**00+000111+110000–0**10011–

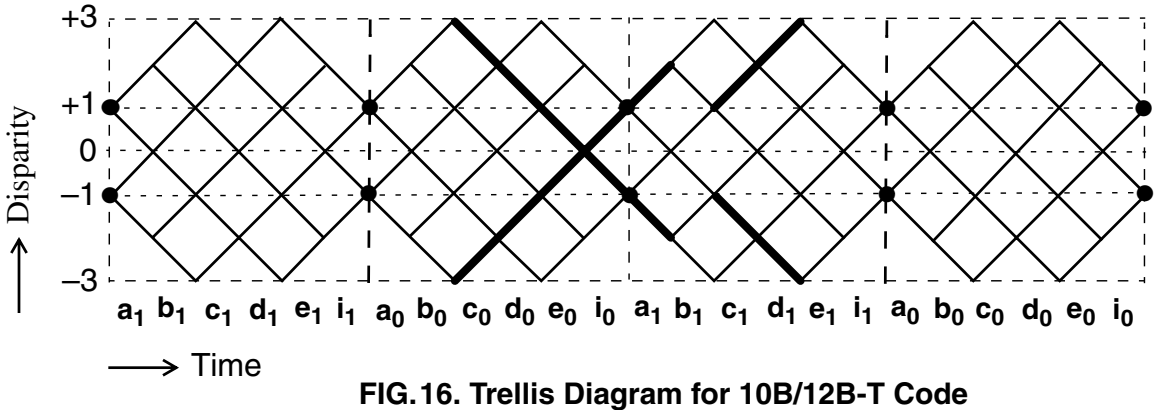


FIG. 16. Trellis Diagram for 10B/12B-T Code

2. For start-up purposes, the coding circuitry can generate a steady stream of runs of six, i.e. a symmetrical waveform at one twelfth the baud rate. The waveform "+110000'001111+110000'001111+..." is obtained by holding the encoder input at a steady K3 value (00111 K=1). The waveform "-1111100'000011-111100'000011-..." is obtained by holding the encoder input at a steady K15 (11110 K=1) value (Refer to Table 6). These waveforms can be used to adjust the receiver circuits on parallel lanes to deal with skew and/or to find the 6B boundaries.
3. The minimum sustainable average transition density is three per 12-bit coded data interval, for example, the bit pattern '+100001−111000−011110+000111+' can be repeated indefinitely. Any single 12-bit interval may have as few as two transitions.
4. The code is dc-balanced with a maximum digital sum variation or running disparity variation of ±3. In the steady-state condition (ignoring irrelevant start-up abnormalities), all 6B vectors start and end with a running disparity of ±1.
5. The normalized maximum dc-offset is (13/6) = 2.17 versus 1.9 for the 8B/10B-T code. The normalized maximum dc-offset is derived from the trellis diagram and is defined as the average area (bit interval x disparity) per bit-interval enclosed between the zero-disparity line and the outermost contour any valid code vector can traverse, which is +110100+ or −001011− for the 5B/6B-T code. It is a better indicator for the low frequency behavior than the traditional DSV parameter. By simulations, it can be shown that at a 2 Gbaud rate, a 3 dB low frequency cutoff at 3.9 MHz (0.195% of Baud rate) produces an eye amplitude closure of 0.5 dB. A cutoff at 7.9 MHz generates a closure penalty of 1 dB for a worst case pattern. It is generally desirable to operate with a high low frequency cut-off in order to filter out low frequency noise from several sources and to permit small reactance values for ac-coupling. As an example, optical receiver designs usually have at least one high-pass filter on chip and so it is very important to reduce the size of the required capacitance. For equivalent results, the low frequency cut-off for the 5B/6B-T code must be set about 5.5% lower than for the 8B/10B or the 8B/10B-T code.
6. The maximum error spread caused by an error in the coded data is five contiguous decoded bits.

Coding Constraints and 12B Control Characters

A total of fifty-seven 12B control characters can be defined if needed without violating the existing run length and disparity constraints. A first set of 28 control characters is identified by K3 (110000 or 001111) in the first half segment, followed by the set of 15 balanced vectors which do not generate a run of six across the center division line. All these balanced 6B vectors are made disparity dependent as shown in Table 6. Thirteen unbalanced 6B vectors also meet the constraints. Beyond that, one can define a new special 6B vector K15 with a leading run of four as listed in Table 6. This vector is restricted to the second half of the 12B characters. So K3 followed by K15 yields another 12B control character which can be recognized as such by the presence of a special vector in either half-segment. A second set of 28 control characters is identified by the presence of K15 in the second half segment and the first half segment restricted to vectors which do not generate a run of six across the center boundary. Essentially, this second set is the first

set of 12B characters with reversed bit order in the coded domain. Interestingly, there is also a second comma character with the same alignment with respect to the 12B boundaries, but the comma sequence is in reverse order (‘11**00**1**0000**011’ or ‘00**1101111**100’). The definitions for the second set of 28 control characters are not included in Table 6.

Table 6. 5B/6B-T Encoding for 10B/12B-T Control Characters

Name	ABCDE K	Inverted Bits	Primary a b c d e i	Alternate a b c d e i	DR Class	DR	DB Class	DB
K0	00000 1	A D I	100101	011010	NDRS6	–	(A•B)’•C’•D’•E’	0
K1	10000 1	C I	101001	010110	NDRS6	–	(A•B)’•C’•D’•E’	0
K2	01000 1	E I	010011	101100	NDRS6	–	(A•B)’•C’•D’•E’	0
K3	11000 1		110000	001111	PDRS6	+		–
K4	00100 1	B I	011001	100110	NDRS6	–	ZB6	0
D/K5	10100 x		101000	010111	PDRS6	+		–
D/K6	01100 x		011000	100111	PDRS6	+		–
D/K7	11100 x		111000	000111	NDRS6	–	ZB6	0
K8	00010 1	B I	010101	101010	NDRS6	–	XB6	0
D/K9	10010 x		100100	011011	PDRS6	+		–
D/K10	01010 x		010100	101011	PDRS6	+		–
K11	11010 1		110100	001011	NDRS6	–	XB6	0
D/K12	00110 x		001100	110011	PDRS6	+		–
K13	10110 1		101100	010011	NDRS6	–	XB6	0
K14	01110 1		011100	100011	NDRS6	–	XB6	0
K15	11110 1		111100	000011	NDRS6	–		+
K16	00001 1	A I	100011	011100	NDRS6	–	XB6	0
D/K17	10001 x		100010	011101	PDRS6	+		–
D/K18	01001 x		010010	101101	PDRS6	+		–
K19	11001 1		110010	001101	NDRS6	–	XB6	0
D/K20	00101 x		001010	110101	PDRS6	+		–
K21	10101 1		101010	010101	NDRS6	–	XB6	0
K22	01101 1		011010	100101	NDRS6	–	XB6	0
D/K23	11101 x		111010	000101	NDRS6	–		+
D/K24	00011 x		000110	111001	PDRS6	+		–
K25	10011 1		100110	011001	NDRS6	–	YB6	0
K26	01011 1		010110	101001	NDRS6	–	YB6	0
D/K27	11011 x		110110	001001	NDRS6	–		+
D/K29	10111 x		101110	010001	NDRS6	–		+
D/K30	01111 x		011110	100001	NDRS6	–		+

Comma Sequence for the 10B/12B-T Code

The preferred comma patterns is '11111011' or its complement '00000100'. The complete 12-bit *comma character* is "+11**0000 0**1**00**11–" or "–00**1111 1**0**11**00+" and it is generated by K3.K2 with some coder circuit modification to complement K2 in special characters when the starting disparity is positive. In the absence of errors, it is sufficient to monitor the seven bold digits for comma detection. An alternate comma pattern is the above bit sequence in reverse order "+11**00**1**0 0000**11–" or "–00**11**0**1 111**00+".

Once vector alignment at the receiver has been established, both the 5B/6B-T and the 10B/12B-T code operate identically for data. The only differences are related to control characters and the comma sequence. Of course, alignment can be established by means other than a comma, e.g, a check for coding violations and step by step changes in alignment, or with the sequence of runs of six as described in list item 2 just above.

Implementation

A 10B/12B-T encoding circuit T_s12encode is illustrated in FIGS.17A and 17B. The corresponding decoding circuit T_r12decode is shown in FIGS. 18A and 18B. As an illustration, these circuits are based on an early design and do not include redundancy for circuit sharing. Slightly faster performance may result, but otherwise there is no reason not to use modified versions of the circuits shown in FIGS. 5A and 5B.

Table 7. 6B-T Trailers of K3 to form 12B-T Control Characters

Name	ABCDE K	Inverted Bits	Primary a b c d e i	Alternate a b c d e i	DR Class	DR	DB Class	DB
K2	0 1 0 0 0 1	E I	0 1 0 0 **1 1**	1 0 1 1 0 0	NDRS6	–	A'•C'•E'•K	0
K3	1 1 0 0 0 1		1 1 0 0 0 0	0 0 1 1 1 1	PDRS6	+		–
D/K5	1 0 1 0 0 x		1 0 1 0 0 0	0 1 0 1 1 1	PDRS6	+		–
D/K6	0 1 1 0 0 x		0 1 1 0 0 0	1 0 0 1 1 1	PDRS6	+		–
D/K7	1 1 1 0 0 x		1 1 1 0 0 0	0 0 0 1 1 1	NDRS6	–	ZB6	0
D/K9	1 0 0 1 0 x		1 0 0 1 0 0	0 1 1 0 1 1	PDRS6	+		–
D/K12	0 0 1 1 0 x		0 0 1 1 0 0	1 1 0 0 1 1	PDRS6	+		–
D/K17	1 0 0 0 1 x		1 0 0 0 1 0	0 1 1 1 0 1	PDRS6	+		–
D/K18	0 1 0 0 1 x		0 1 0 0 1 0	1 0 1 1 0 1	PDRS6	+		–
D/K20	0 0 1 0 1 x		0 0 1 0 1 0	1 1 0 1 0 1	PDRS6	+		–
D/K23	1 1 1 0 1 x		1 1 1 0 1 0	0 0 0 1 0 1	NDRS6	–		+
D/K24	0 0 0 1 1 x		0 0 0 1 1 0	1 1 1 0 0 1	PDRS6	+		–
D/K27	1 1 0 1 1 x		1 1 0 1 1 0	0 0 1 0 0 1	NDRS6	–		+
D/K29	1 0 1 1 1 x		1 0 1 1 1 0	0 1 0 0 0 1	NDRS6	–		+
D/K30	0 1 1 1 1 x		0 1 1 1 1 0	1 0 0 0 0 1	NDRS6	–		+

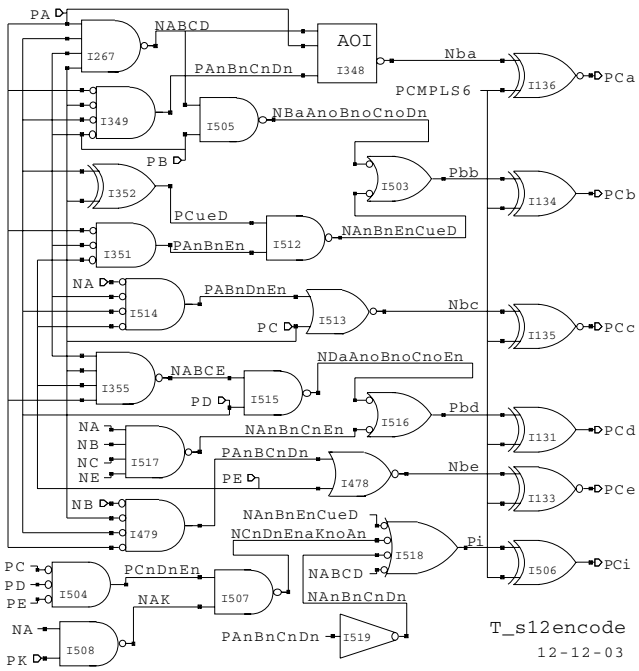


FIG. 17A. 10B/12B-T BIT ENCODING

In the circuit example described here, a limited set of fourteen 12B vectors is implemented to keep the circuits simple. Table 7 lists the 6B vectors which may follow contiguously the special character K3 to form a 12B control character. With this limited set, only K2 and K3 are handled differently from their normal data mode D2 and D3, respectively, for encoding and decoding. K2 is a balanced vector made disparity dependent to form a comma sequence together with the preceding K3 character regardless of the running disparity at the start of K3. K3 has already been described as part of the 8B/10B-T code.

5B/6B Encoding for 10B/12B-T Format, FIG. 17A

The 6-bit encoding portion of a 10B/12B encoder circuit T_s12encode is shown in FIG. 17A. The bit encoding for K2 is unchanged from data mode. However, the equation for the encoding for bit "i" is slightly changed because the value of K attached to the uncoded bit value "01000' may now be a one which would prevent the i-bit from being forced to one. The added term "+A' is underlined and in bold characters in the revised equation below:

$$i = A'\bullet B'\bullet E'\bullet(C\neq D) + C'\bullet D'\bullet E'\bullet(K'\underline{\mathbf{+ A'}}) + A'\bullet B'\bullet C'\bullet D' + A\bullet B\bullet C\bullet D$$

This equation is equivalent to:

$$i = A'\bullet B'\bullet E'\bullet(C\neq D) + C'\bullet D'\bullet E'\bullet(K\bullet A)' + A'\bullet B'\bullet C'\bullet D' + A\bullet B\bullet C\bullet D$$

Disparity Control for 10B/12B-T Format, FIG. 17B

The disparity control for encoding in the 10B/12B-T format is shown in FIG. 17B. Because of the expanded set of 6B control vectors, the disparity controls have to be modified and some circuit simplifications are not applicable. In the equation for PDRS6, the last term E'•K must be expanded to K•A•B•C'•D'•E' to identify the K2 vector.

$$PDRS6 = E'\bullet(A\neq B)\bullet(C\neq D) + A'\bullet(B\neq C)\bullet(D\neq E) + B'\bullet C'\bullet E\bullet(A\neq D) + K\bullet A\bullet B\bullet C'\bullet D'\bullet E'$$

The vector K2 is an addition to the set of vectors which require a *negative entry disparity.* Within the limited set of Table 7, it can be identified by A'•C'•E'•K and so the equation for the required negative entry disparity becomes:

$$NDRS6 = A\bullet B\bullet C\bullet D' + C\bullet D\bullet E\bullet(A\neq B) + A\bullet B\bullet C'\bullet D\bullet E + \underline{\mathbf{A'\bullet C'\bullet E'\bullet K}}$$

The vector K2 is also an addition to the set of *balanced S6 vectors*:

$$BALS6 = (D\neq E)\bullet(A\oplus B\oplus C)' + D\bullet E\bullet(A\oplus B\oplus C) + A'\bullet B'\bullet D'\bullet E' + C'\bullet D'\bullet E'\bullet K' + A\bullet B\bullet C\bullet E' + \underline{\mathbf{A'\bullet C'\bullet E'\bullet K}}$$

In the circuit of FIG. 17B, the following abbreviations are used for net names:

n1 = NDFS6•E'•A≠B•C≠D n2 = NDFS6•A'•B≠C•D≠E

n3 = NDFS6•B'•C'•E•A≠D n4 = NDFS6•K•A•B•C'•D'•E'

n5 = PDFS6•A•B•C'•D•E n6 = PDFS6•C•D•E•A≠B

n7 = PDFS6•A•B•C•D' n8 = PNDFS6•K•A'•C'•E'

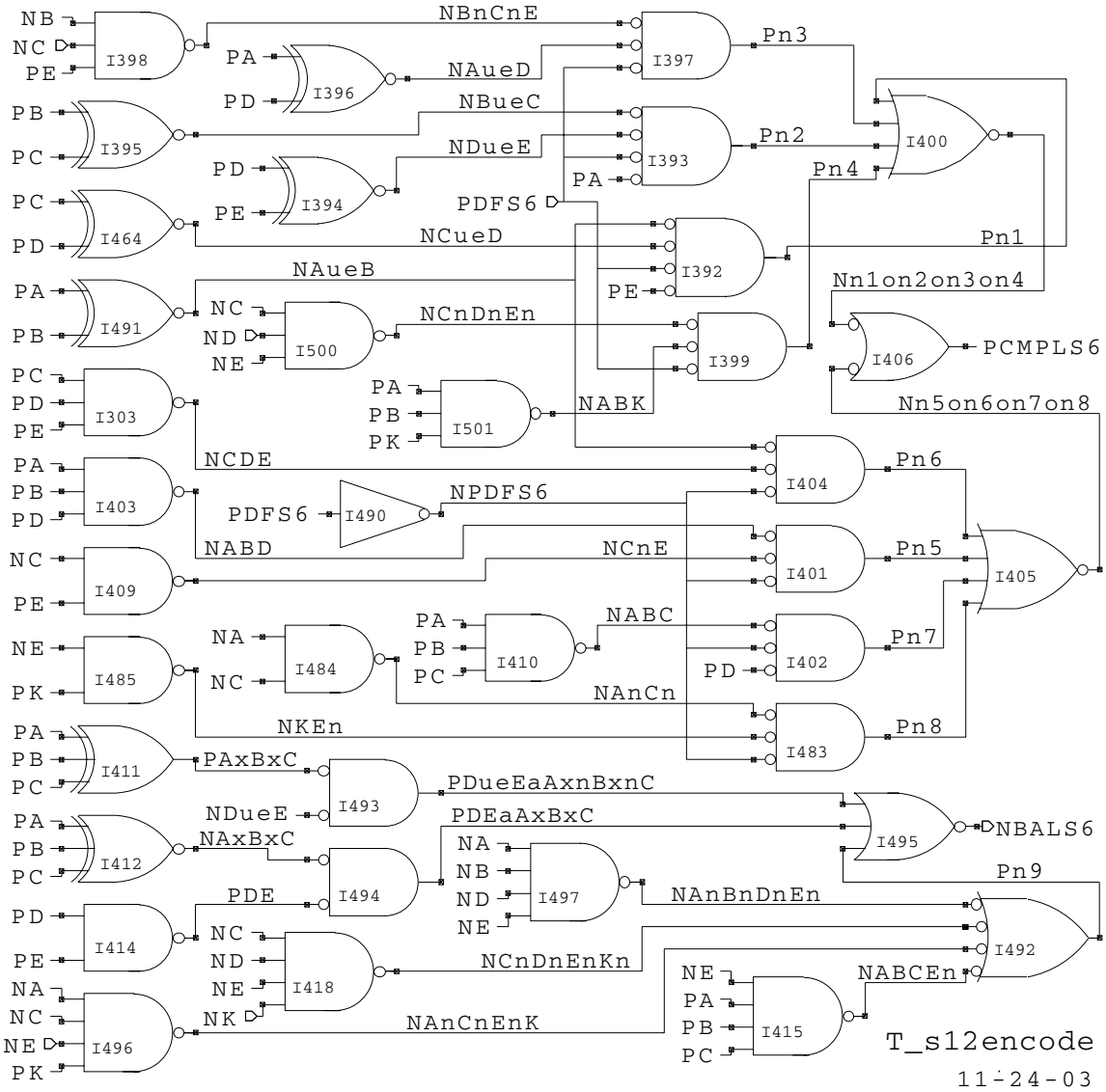


FIG. 17B. 10B/12B-T DISPARITY CONTROL

6B/5B-T Decoding of 12 B Format

The bit decoding circuit T_r12decode is illustrated in FIG. 18A. The alternate vector for K2 "101100" is normally decoded to D13=10110 in the B6 format, but if the preceding 6B vector is K3, then it is decoded to D2 = **<u>0100</u>**0, i.e. the first four bits are complemented. Since the alternate 6B vector for D/K18 '101101' is decoded to D18 = **<u>01001</u>**, i.e. the first four bits are also complemented, it follows that the first four bits are complemented if

(K3LAST•a•b'•c•d•e') is true. The "i" bit can be ignored. However, because the full logic term including i' is required for the function PDRR6, the implementation shown does not include simplifications available for applications which require disparity checks. Therefore:

CMPLa = CMPL5 + b'•i•(a≠c)•(d≠e) ***+ K3LAST•a•b'•c•d•e'•i'***

CMPLb = CMPL5 + a'•b•e'•i•(c≠d) + a'•b'•c•i•(d≠e) ***+ K3LAST•a•b'•c•d•e'•i'***

CMPLc = CMPL5 + a•b'•c•d'•e'•i ***+ K3LAST•a•b'•c•d•e'•i'***

CMPLd = CMPL5 + b'•i•(a≠c)•(c≠d)•(d≠e) ***+ K3LAST•a•b'•c•d•e'•i'***

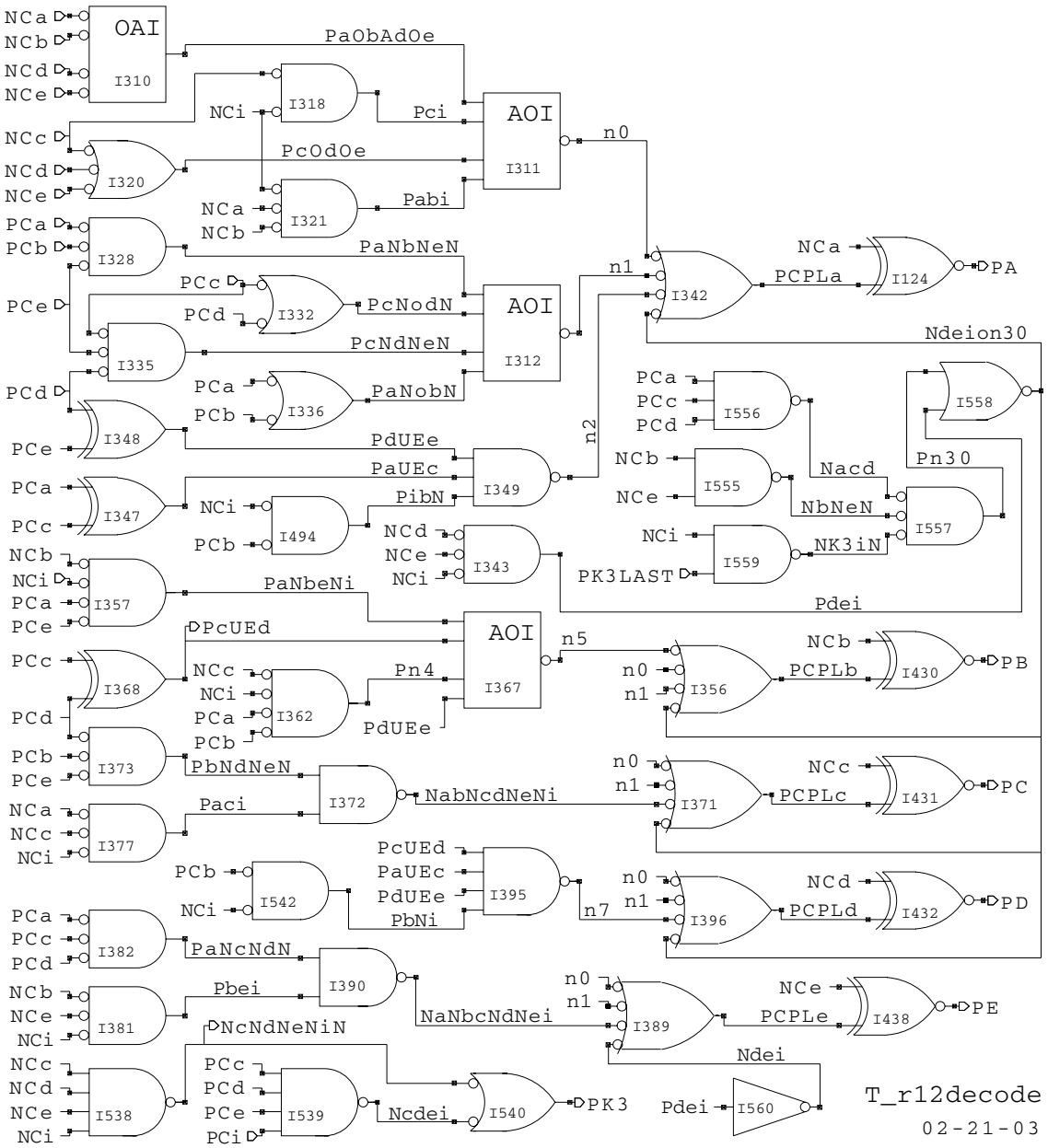


FIG. 18A. 6B/5B-T FOR 12B/10B-T DECODER

CMPL5 = n01 + n02 + n1 has been previously defined under 10B/8B-T Decoder. In the circuit diagram of FIG.19, the net name n30 stands for K3LAST•a•b'•c•d•e'•i', and n0 = a•b•i•(c+d+e) + c•i•(a+b)•(d+e).

Disparity and invalid Vector Checks

The circuit shown in FIG. 18B checks for invalid vectors and classifies all vectors according to their disparity attributes. For the limited set of control characters implemented, the primary vector K2P = 010011 and the alternate vector K2A = 101100 require a negative and positive entry disparity, respectively, when following K3. So the positive required front end disparity PDRR6 is augmented by the term: K3LAST•a•b'•c•d•e'•i' = n30

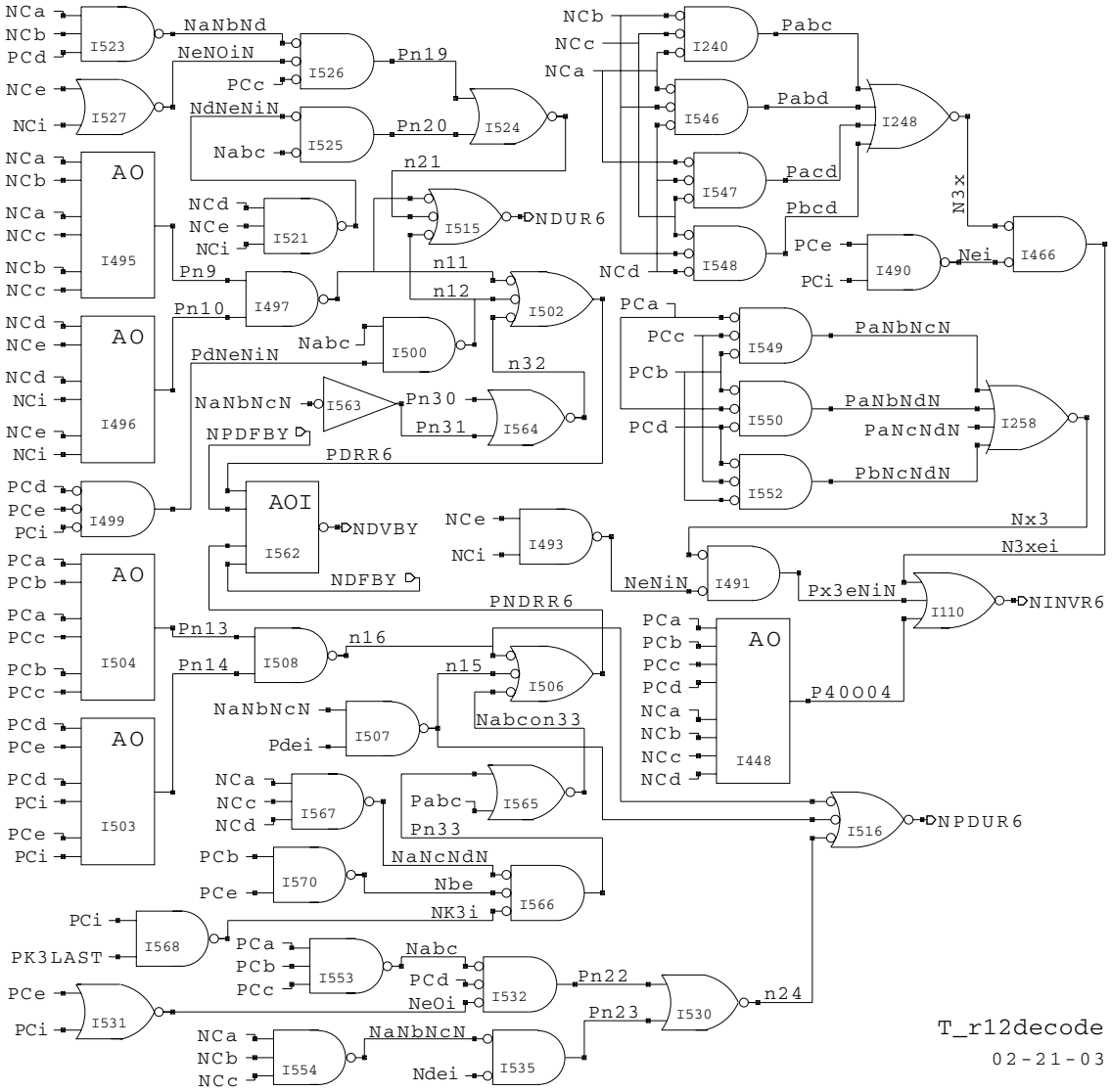


FIG. 18B. 12B/10B-T Disparity + Error Checks

The equation for the positive required front end disparity PDRR6 must include n30:

PDRR6 = a'•b'•c'+d'•e'•i'•(a•b•c)'+(a'•b'+a'•c'+b'•c') • (d'•e'+d'•i'+e'•i')+n30

The negative required front end disparity NDRR6 is augmented by the term:
K3LAST•a'•b•c'•d'•e•i = n33

The equation for the negative required front end disparity NDRR6 must include n33:

NDRR6 = a•b•c + d•e•i•(a+b+c) + (a•b+a•c+b•c) • (d•e+d•i+e•i) + n33

More validity checks could be implemented in the 12B domain, especially with regard to invalid control character combinations. It is expected, that most applications will not need those checks since most applications are expected to be focused on 5B/6B-T code and will use the 12B format only for start-up and special conditions. In other applications with multiple parallel lanes with error correction, the disparity checks at the receiver can be dispensed of because such errors are detected by other means.

Staggered and Pipe-lined 5B/6B-T and 3B/4B-T Encoding and Decoding

Staggered timing of the 5B/6B-T and 3B/4B-T encoding and decoding circuits can be used to reduce the latency of transceivers which can be an issue for short connections. It also helps to reduce the performance requirements of the circuits.

The signal PCMPLS4 in FIG. 5B is in the critical path of the encoder circuit. Before consideration is given to pipe-lining the entire 8B/10B-T encoder (or decoder) circuit, the possibility of delaying just some stages of this signal and the final stage of 4B vector encoding with reference of the 6B vectors must be evaluated. The encoder output is typically fed to a serializer. If the serializer is a multiplexer of the commutator type, it is necessary to double latch some of the trailing bits anyway so the updating of the 10-bit register does not interfere with the serialization, i. e. the first 6 bits 'abcdei' are updated while the last four bits 'fghj' are serialized and vice-versa. With this solution, five latches timed six serial baud intervals after the input update capture the following signals: Pbf, Nbg, PH, Nbj, and PCMPLS4. The high speed serializer circuitry then has to handle only the four exclusive OR gates which generate the coded bits PCf, PCg, PCh, and PCj. Still better timing margins can be obtained if the 2-way gate I467 in the PCMPLS4 path is also moved into the serializer domain at the cost of an extra latch.

Staggering the timing for the 6B and 4B parts with a few additional circuits to relieve timing and latency problems has been described above. If that simple solution is not enough, the circuit functions may have to be executed in two (or more) full clock cycles which adds latency and requires more extra latches.

For some applications such as very high speed parallel busses, a pipe-lined structure is preferable over several parallel CoDecs multiplexed into a single serial line to overcome circuit delay problems. At very high speeds, the serializer and deserializer circuits may be implemented in another technology altogether using different design tools. The designers on both sides of the interface may prefer a simple clock structure, a minimum number of circuits in the high power, high speed area, and an interface where all ten bits are updated simultaneously. In this situation, relaxation of timing constraints can be provided by

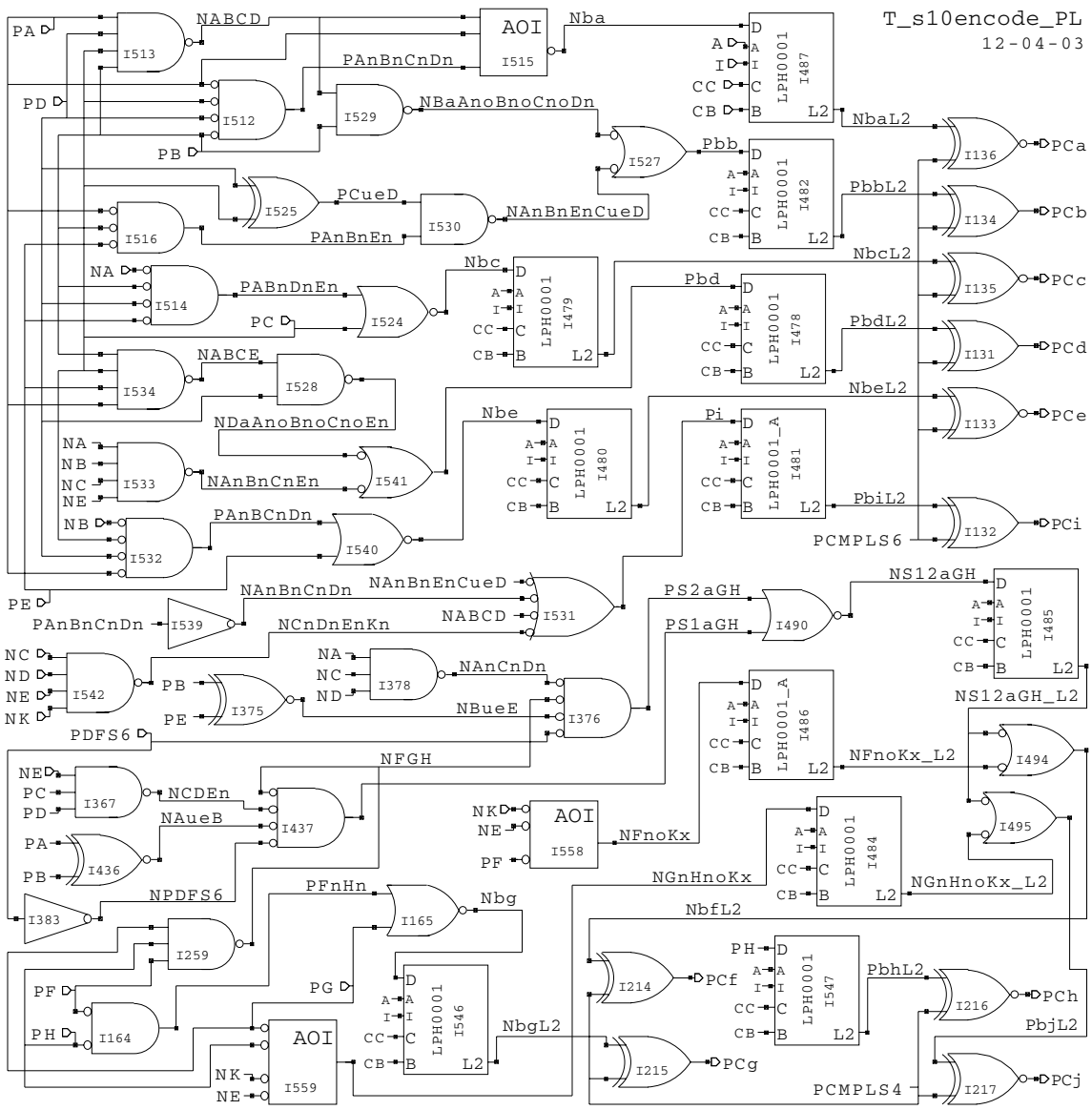


FIG. 19A. PIPE-LINED 8B/10B-T BIT ENCODING

inserting latches at suitable nodes and taking two basic clock intervals to obtain the coded or decoded results. An example is shown in the circuits of FIGS. 19A and 19B. In the first cycle, primary vector bit encoding and partial disparity attributes are established. In the second cycle, the generation of the complementation control signals is completed and the final bit inversion, if any, is performed. The symbols LPH represent standard low power Level Sensitive Scan Design D-Latches. The clock CC drives the L1 latches and the clock CB drives the L2 latches.

To best exploit this technique, all circuits must be reevaluated. In the example cited, the signal path to ‘Pi’ is slightly longer than the encoding circuits for the other five bits because of the of the 4-way OR function. This function could be split into two parallel 2-

way OR functions which are then combined by a 2-way NOR2 gate in the next cycle at the cost of an extra latch. Similarly, the two NOR4 gates I204 and I439 at the lower right side of FIG. 5A, are split up into two NOR2 gates for pipe-lining as shown in FIG. 19A.

For another example of circuit changes in the pipe-lined version refer to the upper right quadrant in the circuit of FIG. 5B. The signals ‘Pna’, ‘Pn2’, and ‘Pn3’ are all gated by a common input PDFS6 to eliminate a gating level. In the pipe-lined version with less critical timing, this gating function might be applied to the signal ‘Pn1on2on3’ instead which then provides a larger margin for the signal Nn1on2on3on4 as shown in FIG. 19B. Similar changes have been made in the ending portions of the signal path for PBALBY and PCMPLS4. Because of the larger margin available in the pipe-lined version, the signal NUBLALS6 is generated by simply inverting the signal PBALS6.

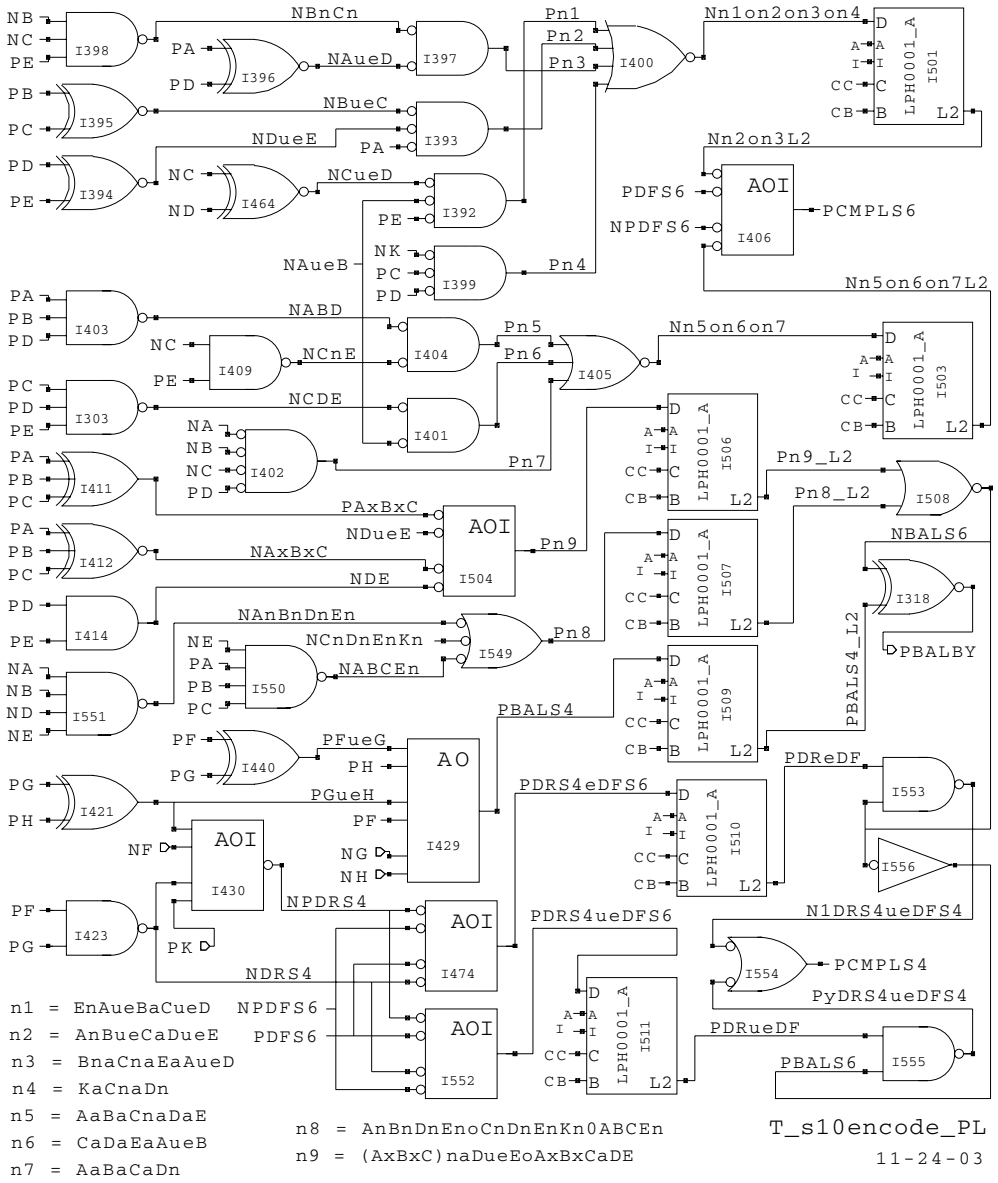


FIG. 19B. PIPE-LINED 8B/10B-T DISPARITY CONTROL

The pipe-lined version has no more than three levels of gates in the first step and generally less delay in the second step. Performance can further be enhanced by adding more latches and transferring some more logic functions of the critical paths from the first step to the second step.

Analogous staggering or pipe-lining of the 6B/5B-T and the 4B/3B-T decoding circuits can be done with similar results.

Acknowledgment

The contribution of Chuck Haymes is gratefully acknowledged. He flushed out errors and verified most of the equations and circuits using logic design tools and then built a circuit macro of a single Coder/Decoder and integrated it on an interface chip with extensive simulations and functional verification.

References

1. IBM US Patent 4,486,739, Franaszek and Widmer “Byte Oriented DC Balanced (0,4) 8B/10B Partitioned Block Transmission Code”, Dec. 4, 1984.
2. IBM Research Report RC18855, Widmer, “The ANSI Fibre Channel Transmission Code”, 04-23-1993.
3. IBM US Patent Application No. 60/289556US1, PCT No. US02/13798, “8B/10B Encoding and Decoding for High Speed Applications”.
4. IBM US Patent 6,614,369, Widmer, “DC Balanced 7B/8B, 9B/10B, and Partitioned DC Balanced 12B/14B, 17B/20B, and 16B/18B Transmission Codes”.
5. IBM US Patent 6,496,540, Widmer, “Transformation of Parallel Interface into Coded Format with Preservation of Baud-Rate”.

doc/Bus/Code8B10B-T