8B10B 详解&综述

目　录

前言

在高速的串行数据传输中,传送的数据被编码成自同步的数据流,就是将数据和时钟组合成单一的信号进行传送,使得接收方能容易准确地将数据和时钟分离,而且要达到令人满意的误码率,其关键技术在于串行传输中数据的编码方法。8B10B 作为互连接口的一种编码技术，设计简单、性能出众，因此成为应用最广泛的技术。然而，它的系统开销高达 25%，问题突出。为了解决这个问题，设计者们一直在探寻改进的方法。本文就将介绍 8B10B 码的编码原理及实现方法，并介绍了一些低开销的编码技术，讨论它们的优势与存在的问题。

1、编码技术基础理论

目前，高速接口正在被广泛应用于包括 SATA、SAS、高速 PCI 等多种标准中。这些接口的速率甚至可以达到并超过每线 10Gbits/s。同时，所有主流 ASIC 和 FPGA 平台也都支持这些高速接口技术。从结构上看，这些高速接口主要包括三个组成部分：

1） 电路部分（串行/解串行）

2）物理部分（实现编码）

3）链路与协议部分（高层）

支持多速率、多协议的串行/解串行器已经实现。以 OIF（光互联论坛）为例，他们已经为两组速率制定了电路规范，分别为 5Gbits/s- 6.375Gbits/s 和 10Gbits/s-11Gbits/s。OIF 同样为两种应用距离制定了规范，分别为短距离（采用一个连接器，8 英寸）和长距离（采用两个连接器，40 英寸）。串行/解串行器还可以被设计用来满足更多的规范，包括不同的速率、距离、电路规格等等。

物理部分的主要任务是对数据进行编码，以保证串行/解串行器的正常运行。这些编码的目的包括：确保必须的变换（“1”到“0”和“0”到“1”的变换），保证稳定的直流均衡（“0”码与“1”码的个数相当），以及满足其它标准的要求（最大化信道带宽利用率，提高对误差的容忍能力等等）。

在光纤通信中，线路编码是必要的，因为电端机输出的数字信号是适合电缆传输的双极性码，而光源不能发射负脉冲，只能用光脉冲的“有”和“无”来表示二进制码中的“1”和“0"。该方法虽然简单，却存在三个问题[1]:

1)遇到数字序列中出现长连“0”或长连“1”时，将给光纤线路上再生中继器和终端光接收机的定时信息提取工作带来困难;

2)简单的单极性码中含有直流分量。由于线路上光脉冲中“1”和“0”是随机变化的，这将导致单极性码的直流成分也作随机性的变化。这种随机性变化的直流成分，可以通过光接收机的交流耦合电路引起数字信号的基线漂移，给数字信号的判决和再生带来困难;

3)不能实现不中断通信业务下的误码检测;

为解决以上问题，通常对于由电端机输出的信号码流，在未对 LED(或 LD)调制以前，一般要先进行码型变换使调制后的光脉冲码流由简单的单极性码,转换为适合于数字光纤传输系统传输的线路码。适合于光纤通信的线路码型有多种，但都要满足以下要求:

1)能保证比特序列独特性。

2)能提供足够的定时信息。

由于在光纤数字传输系统的传输中，只传送信码，而不传送时钟，因此在接收端，必须从收到的码流中提取出定时信息，以利于上述的定时提取。必须限制线路码流中同符号连续数不能过大，也就是说，应避免长连“0”及长连“1”的出现，提高电平跳变的密度，是定时提取较为简单。

3)减少功率密度中的高低频分量。

线路码的功率谱密度中的低频分量是由码流中的“0”、“1”分布状态来决定的，低频分量小，说明“0”、“1”分布比较均匀，直流电平比较恒定，也就是信号基线浮动小，有利于接收端判决电路的正常工作。高频分量是由线路码的速率决定的，这在带宽(色散)限制系统中特别值得注意，在这种系统中，中继距离主要由光纤线路的总带宽(总色散)决定，如果线路码速率提高的太多，会使中继距离大大缩短。

4)要有利于减少码流的基线漂移，即要求码流中的“1"、"0”码分布均匀，否则不利于接收端的的再生判决。

5)码率增加要少，光功率代价要低。

6)接收端将线路码还原后，误码增殖要小。

线路传输中发生的一个误码，往往使接收端的解码(反变换)发生多个错误，这就是误码倍增，也叫做误码扩展或误码增值。由于误码倍增，使光接收机要达到原要求的误码性能指标，必须付出光功率代价，即光接收机灵敏度劣化。因此误码倍增系数越小越好。

7)能提供适当的冗余度。

8）低的对称抖动。

传输的比特序列必须保持低的码型相关的抖动。

9)易于实现。

数字光纤通信系统中常用的线路码型有:加扰二进码、插入比特码和 mBnB 码。

2、8B10 码编码原理

8B/10B编码最初由IBM公司的Albert X.Widemer和Peter A.Franaszek发明,并应用于ESCON(200M互连系统)中[2]。它是mBnB编码中的一个特例。

8B/10B编码方法是把8bit代码组合编码成10bit代码,代码组合包含256个数据字符编码和12个控制字符编码,分别记为Dx. y和Kx.y。通过仔细选择编码方法可以获得不同的优化特性。这些特性包括满足串行/解串行器功能必须的变换；确保“0”码元与“1”码元个数的一致，又称为直流均衡；确保字节同步易于实现（在一个比特流中找到字节的起始位）；以及对误码率有足够的容忍能力和降低设计复杂度[3]。

8B/10B编码方案是把8bit数据分成2个子分组：3个最高有效位(y)和5个最低有效位(x)。代码字按顺序排列,从最高有效位到最低有效位分别记为H、G、F和E、D、C、B、A。3bit的子分组编码成4 bit,记为j、h、g、f;5 bit的子分组编码成6bit,记为i、e、d、c、b、a,其映射关系如图1所示, 4bit和6bit的子分组再组合成10bit的编码值。

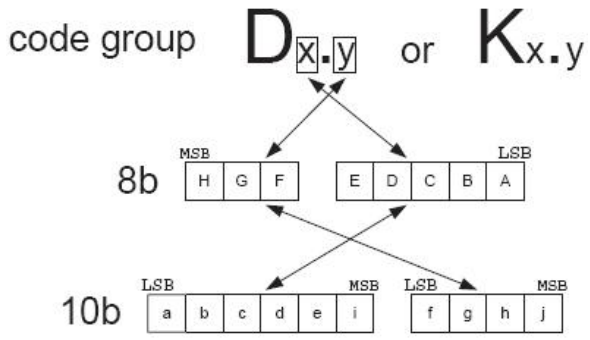

图1 8B/10B码编码原理图

将8bit数据分成3bit和5bit两组,分别对应10bit中的4bit和6bit,直流平衡代码的不平衡度就是通过“0”的个数减去“1”的个数来计算得到的。如果4bit和6bit的各分组中“0”

和“1”的个数相等,称为完美平衡代码,或称为完美的直流平衡代码,无需补偿,但是这种情况是不可能的。因为在4bit的子分组中, 16种编码中只有6 种是完美平衡的,这对于3bit的8种编码值是不够的。同时,在6bit的子分组中也只有20种编码是完美平衡的,对于5bit的32种编码值也是不够的。由于4 bit和6bit的两个子分组都是偶数个位数,而不平衡度不可能是“+1”或“-1”,因此,在8B/10B编码方案中还要使用不平衡度为“+2”和“-2”的值。在编码过程中,用一个极性偏差(running disparity, RD)参数表示不平衡度,在不平衡时用2个10 bit字符表示一个8位字符,其中一个称为RD- ,表示“1”的个数比“0”的个数多2个,另一个称为RD+ ,表示“0”的个数比“1”的个数多2个。

表1列出了3 bit编码成4 bit和5 bit编码成6bit的代码值,在3bit到4 bit的编码中,编码1、2、5、6使用了“1”和“0”相邻的完美平衡代码,编码采用一对一的关系;编码3使用了“1”和“0”有间隔的完美平衡代码,编码采用一对二的关系;编码0和4使用了“1”和“0”有间隔的不平衡代码,编码也采用一对二的关系;编码7使用了3个连续“1”或“0”的不平衡代码,为了防止更多连续“1”或“0”出现,提供了4种代码选择。在4bit到6 bit编码中,有19个编码RD-和RD +代码相同,编码7的RD-是111000,而RD+是000111。其余12个不平衡代码的RD-编码值包含4个“1”,而RD-编码值包含4个“0”。8B/10B编码规则将4 bit和6 bit组合,使其最坏的情况下10 bit代码值的不平衡度为“+2”或“-2”,例如:不平衡度为“+2”的4 bit编码值是不会和不平衡度为“+2”的6bit编码值组合在一起的,因为那样会产生一个不平衡度为“ + 4”的10 bit的编码值。

表1 8B/10B子分组编码表

3B Decimal	0	1	2	3	4	5	6	7
3B Binary(HGF)	000	001	010	011	100	101	110	111
4B Binary(fghi)	0100或 1011	1001	0101	0011或 1100	0010或 1101	1010	0110	0001或 1110或 1000或 0111

5B Decimal	5B Binary(EDCBA)	6B Binary(abcdei)	5B Decimal	5B Binary(EDCBA)	6B Binary(abcdei)
0	00000	100111或 011000	16	10000	011011或 100100
1	00001	011101或 100010	17	10001	100011
2	00010	101101或 010010	18	10010	010011
3	00011	110001	19	10011	110010
4	00100	110101或 001010	20	10100	001011
5	00101	101001	21	10101	101010
6	00110	011001	22	10110	011010
7	00111	000111或 111000	23	10111	111010或 000101
8	01000	111001或 000110	24	11000	110011或 001100
9	01001	100101	25	11001	100110
10	01010	010101	26	11010	010110
11	01011	110100	27	11011	110110或 001001
12	01100	001101	28	11100	001110
13	01101	101100	29	11101	101110或 010001
14	01110	011100	30	11110	011110或 100001
15	01111	010111或 101000	31	11111	101011或 010100

每一个代码组将转变为 2 种可能取值中的一种，见表 2 中的 RD-（极性偏差为负）列和RD+(极性偏差为正)列，RD-不平衡度或是“-2”或是“0”；RD+的不平衡度或是“+2”或是“0”。

表 2　8B/10B 码映射表

Code Group Name	Octet Value	Octet Bits HGF EDCBA	Current RD –	Current RD +
			abcdei fghj	abcdei fghj
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100
D8.0	08	000 01000	111001 0100	000110 1011
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001
D9.1	29	001 01001	100101 1001	100101 1001
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001

(continued)

Code Group Name	Octet Value	Octet Bits HGF EDCBA	Current RD –	Current RD +
			abcdei fghj	abcdei fghj
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101
D10.2	4A	010 01010	010101 0101	010101 0101
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011
D11.3	6B	011 01011	110100 1100	110100 0011
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100

(continued)

Code Group Name	Octet Value	Octet Bits HGF EDCBA	Current RD –	Current RD +
			abcdei fghj	abcdei fghj
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010
D12.4	8C	100 01100	001101 1101	001101 0010
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010
D13.5	AD	101 01101	101100 1010	101100 1010
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110
D14.6	CE	110 01110	011100 0110	011100 0110
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000
D15.7	EF	111 01111	010111 0001	101000 1110

(continued)

(continued)

Code Group Name	Octet Value	Octet Bits HGF EDCBA	Current RD –	Current RD +
			abcdei fghj	abcdei fghj
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

深入分析表2所表示的整个8B/10B编码的内在相关性可以知道：

1）当3B码有唯一4B编码时（001，010，101，110），若5B码为具有单值编码的码字，8B/10B编码表的RD-列和RD+列取唯一的10B编码，并且其RD值为5；若5B码具有双值编码，则在RD-列取RD值为4的6B编码，在RD+列取RD值为2的6B编码；若5B码为00111，则在RD-列取111000，在RD+列取000111；

2）当3B码有双值4B编码时（000，100），若5B码为18种具有单值编码的码字，则4B编码在RD-列和RD+列分别取RD值为3和1的编码，3B码为011的情况下则分别取1100 和0011；若5B码为13种有双值编码的码字时，则在RD-列和RD+列分别取6B RD值为4、4B RD值为1和6B RD值为2、4B RD值为3的编码；若5B 码为00111，3B 码为000，100，则在RD-列和RD+列分别取6B为111000、4B RD值为3和6B为000111、4B RD值为1的编码；若5B 码为00111，3B 码为011时，在RD-列和RD+列的编码分别111000、1100和000111、0011；

3）当3B码为111 时，根据8B/10B编码表可以看出，1000和0111这组编码分别只出现在5B码组为01011，01101，01110的RD+列和5B码组为10001，10010，10100 的RD-列，其它的编码规则同上面分析的3B码具有双值编码的情况一样来采用0001和1110这组编码。

另外，8B 码分为D 分组（数据字节编码）和K 分组（特殊控制符号编码），上面分析的D分组，对于K分组，共有12个8B 码组，其编码规则与D 分组不同，但是由于其码组少，编码比较简单，只需要在编码过程中根据输入判断这12个码组是否为K分组，再根据K分组编码规则进行编码。

表 3　8B/10B特殊字符编码真值表

S. C. Byte Name	S. C. Code Name		Bits		Current RD-		Current RD +	
			HGF	EDCBA	abcdei	fghj	abcdei	fghj
K28. 0	C0. 0	(C00)	000	00000	001111	0100	110000	1011
K28. 1	C1. 0	(C01)	000	00001	001111	1001	110000	0110
K28. 2	C2. 0	(C02)	000	00010	001111	0101	110000	1010
K28. 3	C3. 0	(C03)	000	00011	001111	0011	110000	1100
K28. 4	C4. 0	(C04)	000	00100	001111	0010	110000	1101
K28. 5	C5. 0	(C05)	000	00101	001111	1010	110000	0101
K28. 6	C6. 0	(C06)	000	00110	001111	0110	110000	1001
K28. 7	C7. 0	(C07)	000	00111	001111	1000	110000	0111
K23. 7	C8. 0	(C08)	000	01000	111010	1000	000101	0111
K27. 7	C9. 0	(C09)	000	01001	110110	1000	001001	0111
K29. 7	C10. 0	(C0A)	000	01010	101110	1000	010001	0111
K30. 7	C11. 0	(C0B)	000	01011	011110	1000	100001	0111

编码器将根据当前的极性偏差为正或为负，来为每个传送的字节选择表2的2种可能编码中的一种。表2中使用的10bit字符为有效字符，未使用的其余10bit字符为无效字符。

编码过程当中5B/6B编码在前而3B/4B编码在后，用于当前5B/6B编码的RD是前面一个字节编码所产生的RD(接收或发送第一个传输字符时为初始化的RD值)，而用于3B/4B编码是前面相邻的5B/6B编码所生成的RD，但是整个字节编码所生成的RD值是由3B/4B编码得到的。

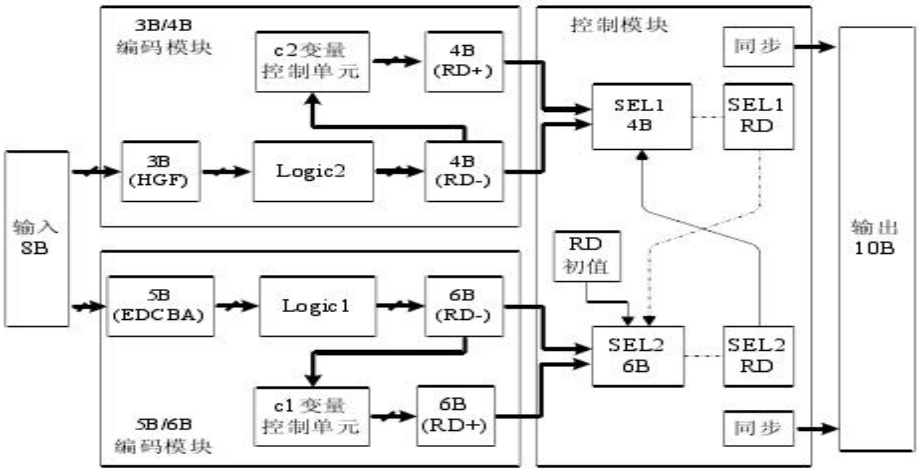

图 2 8B/10B码的实现流图

编码器初始状态的一般取负极性偏差值(RD-),如果当前极性偏差为RD-,则编码器会在RD-栏选择8bit数据的对应值输出;同时检测10bit的编码值是否为完美平衡代码,如果是完美平衡代码,那么极性偏差值(RD-)不改变,否则,改变极性偏差值RD-值为RD+值。同样地,如果当前的极性偏差为正(RD+) ,编码器选择RD+栏的对应值;如果被选择10bit编码是一个完美平衡代码,那么极性偏差的值将保持(RD+)不变,否则,极性偏差值改变成RD-。这样交替地使用RD-栏和RD+栏的值,使得差分信号的直流分量尽量小。图3描述了RD极性偏差状态转移的计算过程[4]。

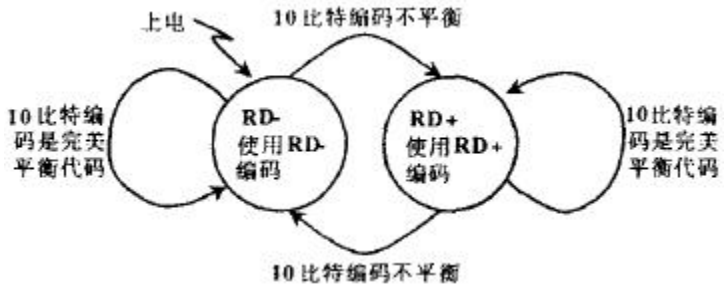

图 3 极性偏差值 RD 状态转移

8B10B借助总共268个字符及它们的反转码，还可以检测任何可能破坏不平衡的误码。但是，即使接收机接收到了正确的码字，有时它们也有可能因为不平衡性而导致误码，这种现象被称为误码复制。如果传输中发生了错误,在接收端会出现2种情况,一种是接收的10bit字符是一个无效字符,另一种是一个有效字符,但极性偏差RD是违规的。例如某一时刻发送端发送的字符串为RD - ,发送D21. 1字符,编码101010 1001;由于“1”和“0”个数相等,因此RD不变,仍为RD - ,接着发送D10. 2字符,编码010101 0101; “1”和“0”个数仍然相等,因此RD不变,继续为RD - ,接着再发送D23. 5字符,编码111010 1010,由于“1”比“0”个数多2个,因此改变RD值为RD + ;在传输过程中发生了一个比特错误,接收端收到的字符串为101010 1011、0101010101、111010 1010,译码后为RD-,字符为D21.0,下一个为RD+,字符为D10. 2,接着下一个出现RD值的违规,即在RD值为RD +时下一个字符串中“1”和“0”的个数

只能是相等或者“1”比“0”的个数少,否则就是违规的。

3、与其他码的比较

MB810是在10G以太网中提出的新的编码策略,不仅保持了二进制分组码本身所固有的许多优点,例如,在给定的相同情况下最大程度地接收SNR(信噪比)、不含直流成分、提供有用的检测能力等,最主要的是大大减少了二进制分组码的带宽困境。这一点对于10G以太网的顺利实现具有重要的现实意义[5][6]。

8B/10B用于光纤信道和所有千兆位以太网的编码方案。MB810和8B/10B码都属于10G以太广域网物理层可选的编码策略,两者均以二进制分组码为基础,都属于更为一般的*m*B *n*B码(*m*个二进制位被映射到*n*个二进制传输位)的例子,而且内建的冗余通过*n*>*m*可以提供希望的传输特性。

但在编码实现方式上两者具有很大的区别,8B/10B的实现结合了两种方案5B/6B和3B/4B ,这两种码的使用只是为了简化映射的方式和实现,使映射可以直接定义为8B/10B码。而MB810 的实现则是10个编码器平行工作,先用一个125 MHz的基本逻辑,然后再执行两倍10∶1运算达到最后结果.

另外,MB810区别在于MB810码只需要合成线性带宽的一半.例如,在10G以太网上采用8B10B码时,主瓣带宽为12.5GHz,若采用MB810码时,主瓣带宽只需6.25 GHz.这个编码的设计是对一个已建立和公布了的定理的创新使用,该定理提出了数字信令系统在理论的最小带宽(MB)下运行的可行性(MB通常被称为Nyquist带宽).根据定义,Nyquist带宽是信令频率的一半,而在类似于NRZ、8B10B 和PAM5 等非最小带宽码中,其带宽是等于信令频率的.

	MB810	8B/10B	备注
输入字节	8	8	
输出字节	10	10	
最大步长	7	5	
RAS	2.5	*	* 未限制
RDS	3	3	
ASV	5	*	* 未限制
DSV	6	6	
频谱带宽	1/(2 *T*)	1/ *T*	1/ *T* 是输出位率
应用方法	硬件	硬件(5B6B + 3B4B)	

表4 MB810与8B/10B详细比较

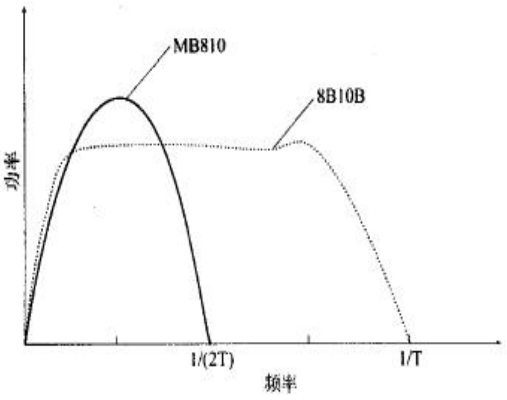

图4 MB810与8B10B的功率谱比较

表4中的RAS、RDS、ASV、DSV的定义如下:

假定线性码每*T* 秒钟输出一个信号,*Yn* 表示第*t* = *nT* 时间内的输出信号,则定义一个编码参数*RAS*(running alternate sum):

$$RAS = \sum_{n=I}^{J} (-1)^n y_n$$

其中, I, J均为整数。RAS为任意一个时间间隔t=IT到t=JT之间输出信号的总和。

ASV(alternating sum variation):在整个编码输出信号流中测量的*RAS*的峰值变化.

$$ASV = \max_{I.J.y} |RAS| = \max_{I.J.y} |\sum_{n=1}^{J} (-1)^n y_n|$$

通过式(1),(2)来正确评估*RAS*和*ASV*的前提是要假定*Yn*具有等距间隔。有两个类似于*RAS* 和*ASV* 的编码参数*RDS* running digital sum) 和*DSV*(digital sum variation)

$$RDS = \sum_{n=1}^{J} y_n$$

其计算的过程可以参见文献[]；
RDS 的峰值变化*DSV* (digital sum variation) 定义为

$$DSV = \max_{1.J.y} | RDS | = \max_{1.J.y} | \sum_{n=1}^{J} y_n |$$

4、优点

8B/10B编码技术编码之所以能得到广泛应用,主要在于它较好地解决了以下问题[7]。
(1) 转换密度:
保证数据流中有足够的信号转换。采用8B/10B编码方法,数据流中连续的“1”或连续的“0”不超过5个,使接收端锁相环(PLL)能正常工作,避免接收端时钟漂移或同步丢失而引起数据丢失。保证了1和0的相对平衡组合,而与数据值无关,简化了时钟恢复,降低了接收机成本。
(2)DC补偿:
在高速的数据传输线路中,一般采用差分信号,需要直流分量尽量小,而8B/10B有DC补偿功能,即链路中不会随着时间推移而出现DC偏移。
(3)检错:
8B/10B编码采用冗余方式,将8位的数据和一些特殊字符按照特定的规则编码成10位的数据,根据这些规则,能检测出传输过程中单个和多个比特误码。
(4)特殊字符:
8B/10B编码规定了一些特殊字符,可用作帧同步字符和其他的分隔符或控制字符，有助于比特流的码组定位和信息识别。许多独立标准都以这个公共字符集为基础,定义更高的协议层。

Table 1. Control code usage in major standards

		SAS	SATA	FC	GE	XAUI	PCIe	IB	RIO	HT
Comma	K28.5	√	√	√	√	√	√	√	√	√
	K28.1						√			
	K28.7									
	K28.0					√	√	√	√	
	K28.2						√	√		
	K28.3	√	√			√	√		√	
	K28.4					√				
	K28.6	√						√		
	K23.7				√		√	√		
	K27.7				√	√	√	√	√	√
	K29.7				√	√	√	√	√	
	K30.7				√	√	√	√		

表 5.　主要标准中的控制码字使用

(5)链路灵活性：
由于采用 8B/10B 编码，链路可以是交流(AC)耦合的，这样就给任一端的设备厂商提供了更大的灵活性。

5、性能分析方法

- 功率谱

线路码的连续功率谱是一个很重要的特性，根据它可以知道信号功率的分布，从而确定信道的带宽，并考虑信道带宽和传输网络的传输函数，对信道进行正确的设计以防止码间干扰。

根据有限状态机模型，采用Carilolaro算法即(状态转移求解法)来分析8B10B码的连续功率谱[8][9][10]。

要对线路码进行功率谱分析，必须先建立其模型。由于码属块码编码,所以我们采用典型的块码编码器模型。如图5所示

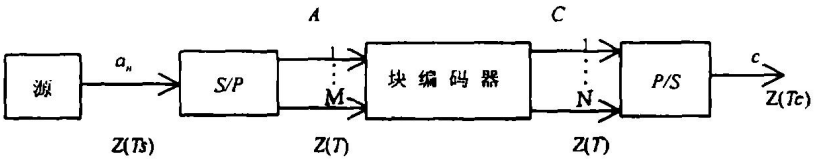

图5 块码编码器框图

由此可看出，S/P,P/S尸都是线形变换，计算时较易处理，而编码器都是非线形变换，直接求相关函数比较困难。因此为简化计算，采用一个有限状态机模型来描述它，并推导它的状态转移图。

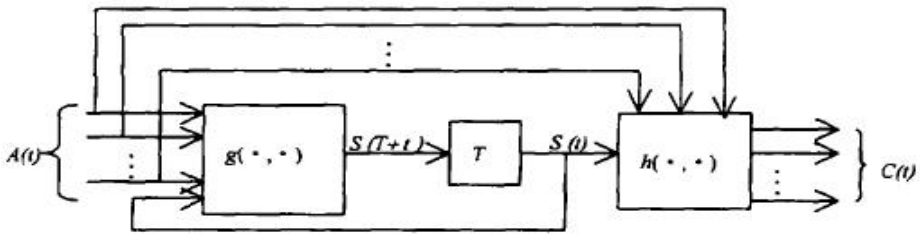

图6 有限状态机模型框图

对状态转移函数 $g(\cdot,\cdot)$，有 $S(t+T)=g\{S(t),A(t)\}$，对输出函数 $h(\cdot,\cdot)$，有 $C(t)=h\{S(t),A(t)\}$，对状态 σ_i 可写为行向量形式，定义转移状态 $\sigma_j \overset{\Delta}{=} g(\sigma_i,\sigma_u)$，含义为当输入 σ_u，当前状态为 σ_i 条件下，将状态转移到 σ_j。输出函数 $\gamma_{iu} \overset{\Delta}{=} h(\sigma_i,\sigma_u)$，其含义是在输入为 σ_u，当前状态为 σ_i 时的输出码字。对于同一 σ_u，在不同状态下有不同输出，因此还可以定义对应于输入 σ_u 的输出矩阵 $\Gamma_u(I\times N)=[\gamma_{1u},\gamma_{2u},\cdots,\gamma_{iu}]$。至此可以画出状态转移图，方法是对每一个 σ_u，均可用一个有方向的支路由 σ_i 画至 σ_j，并标为 σ_u/γ_{iu}。更简单的方法是列出状态转移表。为此定义状态转移矩阵 $E_u(I\times I)$：

$$E_u(i,j)\overset{\Delta}{=}\begin{cases}1, & g(\sigma_i,\sigma_u)=\sigma_j\\ 0, & others\end{cases}$$

一般输入集{0，1}，假设输入序列a(t)是一个平稳无记忆的源，且 $P = P\{a(t) = 1\}$ ，$q = 1 - p$ ，则可定义 $q_u = p^M \cdot q^{M-M_u}$ 为源码字的概率，其中 M_u 为源码字中1的个数。向量 $V'(f) = [1, \exp(-j2\pi fT_c), \cdots, \exp(-j2\pi f(N-1)T_c)]/N$ 的作用是计算并/串之后的序列连续功率谱。Paz等已证明，状态序列是一个无记忆的马尔可夫链，其转移概率矩阵为

$$\Pi = \sum_{u=1}^{K} q_u E_u$$

在此基础上。可以采用Cario laro提出的一种较简便而严格的状态转移矩阵算法计算8B10B码的连续功率谱。其计算公式如下：

$$W_s(f) = V'(f)[T[Y(Z) + Y^{-1}(Z^{-1})]]V(f)$$

$$Y(Z) = \frac{1}{2}(C_0 - m_c m_c') + C_1 B(Z) C_2$$

$$C_0 = \sum_{u=1}^{K} q_u \Gamma_u' D \Gamma_u$$

$$C_1 = \sum_{u=1}^{K} q_u \Gamma_u' D E_u (U - \Pi\infty)$$

$$C_2 = \sum_{u=1}^{K} q_u \Gamma_u$$

$$B(Z) \overset{\Delta}{=} Z^{-1} \sum_{K=-\infty}^{+\infty} (\Pi - \Pi\infty)^K Z^{-K} = \{ZU - (\Pi - \Pi\infty)\}^{-1}$$

$$m_c = \sum_{u=1}^{K} q_u \pi \Gamma_u$$

$$Z = \exp(j2\pi fT)$$

此算法的步骤是：根据 q_u 和状态转移矩阵 E_u 求出状态转移概率矩阵 Π，而状态序列是无记忆的马氏链，故可求出平稳分布 π，在此基础上结合输出矩阵并代入公式可求出编码后串行序列的连续谱。

就8B10B编码原理，可以得到当p=q=0.5时的 Π 值以及功率谱图如下：

$$\Pi = \sum_{u=1}^{256} q_u E_u = \begin{bmatrix} 0.5234375 & 0.4765625 \\ 0.4765625 & 0.5234375 \end{bmatrix}$$

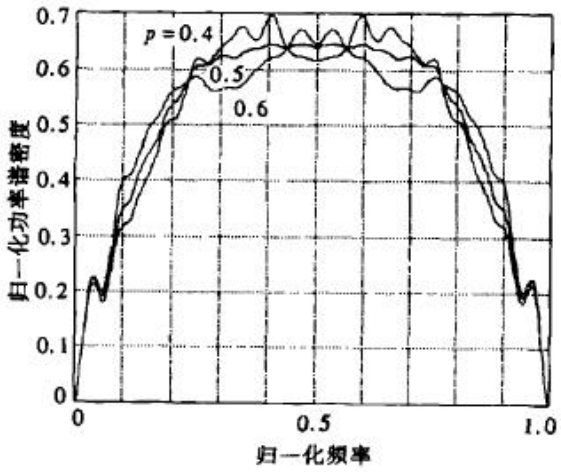

图7 8B10B的功率谱

由图7可见，8B10B码大部分能力集中在码速的二分之一处，对高频和低频分量有较好的抑制特性，适合于千兆以太网中的高速数据传输。

由于这种求功率谱的方法需要得到矩阵的逆，对于复杂的编码，马尔科夫链有很多状态，那么矩阵的逆是一个高阶矩阵，其计算复杂度很大。为此，有文献[11]提出了另一种求序列码功率谱的方法。它先求输入序列的自相关 $r(k)=E\{a_n,a_{n+k}\}$，然后求自相关的Z变换 $R(z)=\sum_{k=0}^{+\infty}r(k)z^{-k}$，再通过一些简单的代数运算就得了编码序列的功率谱 $\Phi(f)=-1+R(z)+R(z^{-1})$。

而Justesen[12]针对复杂编码也提出了一种近似方法。他提出了低频截止频率 f_0 与极性偏差(running disparity, RD)的方差 σ_z^2 之间的近似关系，截止频率的定义为

$$\frac{1}{T}H(2\pi f_0T)=\frac{1}{2}$$

式中 $H(2\pi f_0T)$ 为码字的功率谱密度函数，T为采样间隔。Justesen的近似公式如下：

$$2\pi f_0T=\frac{1}{2\sigma_z^2}$$

这个近似关系非常重要，因为序列的和方差比计算完全的频谱容易多，因此假如能够计算出和方差，就可以确定复杂编码重要的低频特性。其和方差的计算过程参见文献[13]

- 眼图

在评价MB810的性能时，除了看其功率谱之外，还提出了观察眼图的方法[5]。

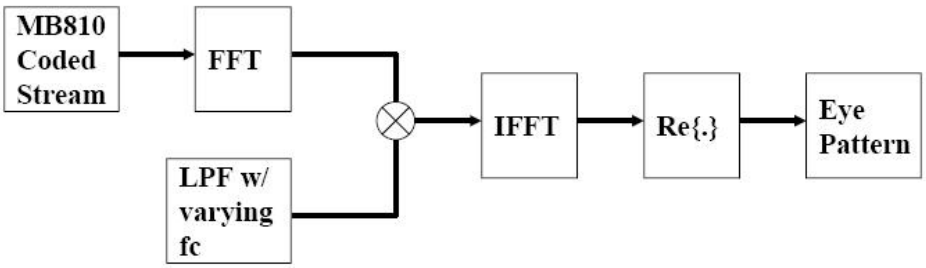

图8 MB810的眼图系统仿真框图

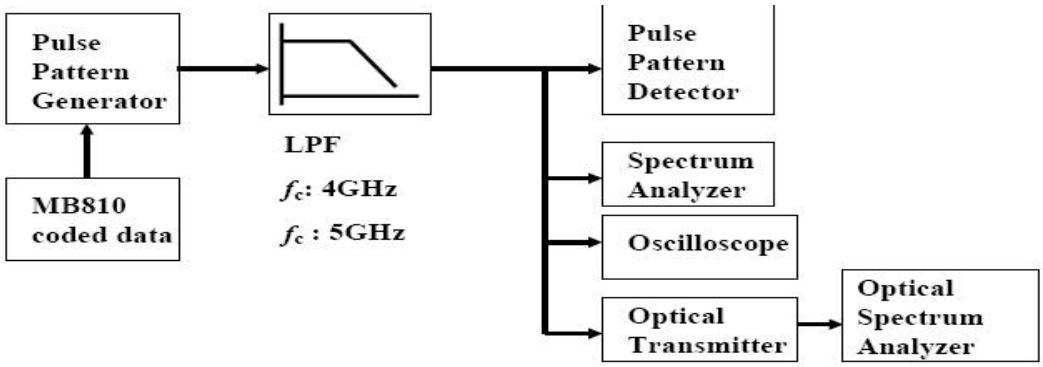

图9 MB810编码数据的测试组成框图

测量的MB810功率谱如图19所示,图10(a)中是使用方波脉冲整形后的功率谱,信道速率设置为10 Gbps ,输入数据速率设置为8 Gbps,从图中可以明显看出在5GHz(Nyquist 频率)处产生一个谱零点,同样在0Hz处也有一个谱零点,但在图中未显示出来.而在10 GHz处的一个尖峰则是因为不完美的方波脉冲整形所引起的,例如,非对称的负载周波.图10(b)中所示的是MB810码流通过一个截止频率为5 GHz的5阶切比雪夫低通滤波器时的结果.

检测这样一种限制带宽结果的比较好的方法是观测如图11所示的合成眼图.左边是传输器的输出眼图,而右边是经过过滤后的结果.可以看出过滤流的眼图相对来说要好些,同时也暗示了一个稳定和几乎无错误的接收.

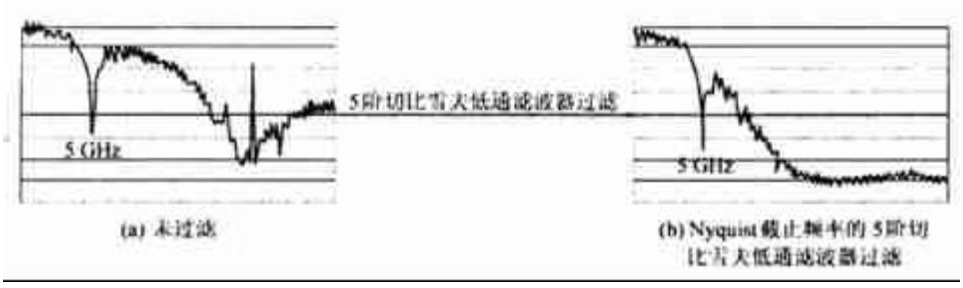

图10 测量的MB810功率谱

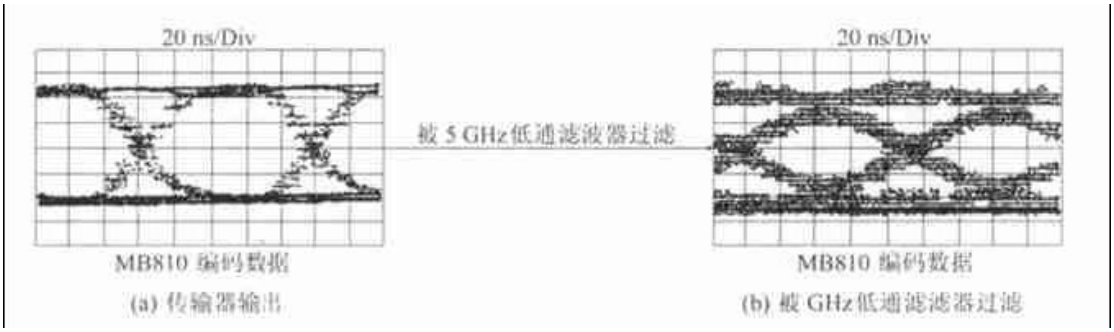

图11 测量的MB810眼图

- 其他

为衡量码字的性能，还可以放到一个系统中，通过观察系统的性能，如时钟抖动、信噪比与误码率的关系图以及接收机的灵敏度等等，还可以比较码字的游程长度、极性差的范围、以及第三章的RDS、RAS等。相关的可以参见文献[14]。

6、实现方式

进行编解码设计时通大体下面几种方法。

第一种是用查找表直接将8位信号映射成10位信号,该方法用存储器存储所有可能出现的码组,再将输入码组转换为存储地址,找出对应的编解码。方法逻辑简单,开发时间很短,但是编解码电路的工作速度受到FPGA内部存储器读取时间的限制,同时不可避免地增加了芯片的面积和功耗。

第二种是通过逻辑运算直接完成编解码功能对，该方法的优点是可以明显减小内部使用面积，难点在于逻辑关系复杂。如果采用卡诺图直接化简则会产生大扇入逻辑表达式，大大限制电路的最高工作速度，同时对逻辑电路的驱动也将加大电路功耗[15][16]。

第三种是，8B/10B编码模块化实现,较好地反映了8B/10B编码的特点,实现流程清楚。实现步骤：①判断是特殊字符还是数据;②若是特殊字符,根据RD极性直接取值；③若是数据,把一节8位字节拆成3bit和5bit,然后在RD控制器的控制下以并列的方式编/译码。RD控制器的原则是：系统设定的RD默认初始值为RD-，RD的初值作为选择信号用以决定5B/6B编码模块中6B码的选取，同时由所选取的6B码计算出新的RD值作用于3B/4B编码模块。4B编码所得到的RD值又作为下一组编码的RD输入值，由此完成了全部的8B/10B编码[7]。

这种方法的组合逻辑实现可以简化码表、减小电路板的面积、有效提高编码工作速度。同时由于电路板的面积减小，功耗也显著降低。

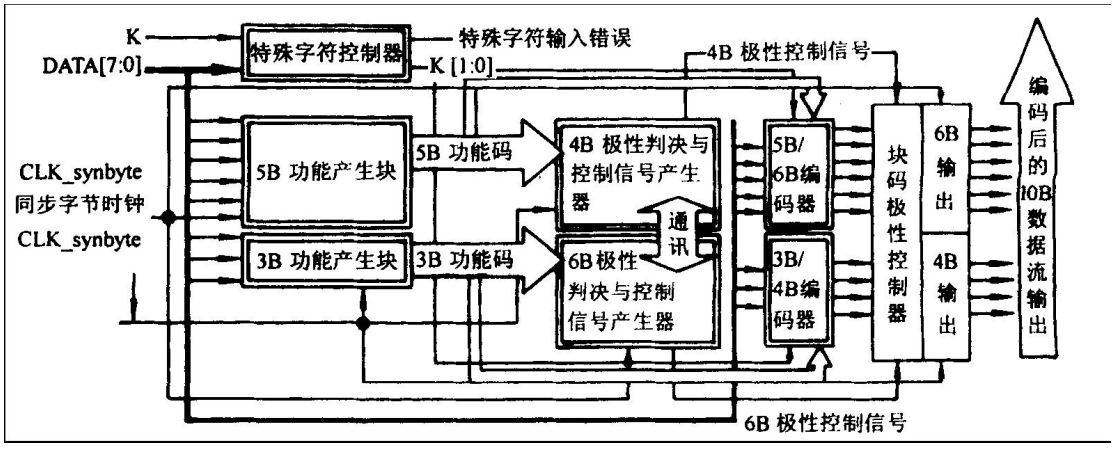

图12　8B/10B编码结构图

7、运用

目前大多数高速串行标准都采用8B/10B编码方案,例如串行连接SCSI、串行ATA、光纤链路、吉比特以太网、XAUI（1吉比特接口）、PCI Express总线、InfiniBand、Serial RapidIO、HyperTransport总线、DVB-ASI以及IEEE1394b接口（火线）技术中。

8、改进

8B/10B 最主要的缺点只有一个，就是高达 25%的系统开销。

改进传统 8B10B 编码技术的一种方法是，在编码之前增加扰码过程。有证据证明，特殊的模式 3 或差分群时延可能会导致重复模式产生不可预测的误码。解决这个问题最直接的办法就是在编码之前对数据进行扰码。

最近，人们提出了几种降低 8B10B 编码技术系统开销的改进方法。这些改进主要基于以下两个基础：一是随着链路速率与数量的增加，25%系统开销的问题显得越来越突出；二是集成技术的进步，使得硬件复杂度不再像过去那样重要[17]。

在这些低开销的改进技术中，有四个非常相似的技术脱颖而出，受到人们的广泛关注：

1）64B66B 编码技术，应用于 10G 以太网 4；

2）OIF CEIP5；

3）10GBase-KR6，应用于 10GbE 背板连接；

4）Interlaken7。

这些技术的共同点是，都以提高硬件设计复杂度（门数目）为代价，换取了较低的系统开销。

- 64B66B 编码技术。

这种技术应用于 10G 以太网（10GBase-R），是一种编码与扰码相结合的技术。首先，数据被分成 8 个字节一组（总共 64 比特）。然后，这些字节采用自同步扰码实现随机化，其特征多项式为 $x^{58}+x^{39}+1$。最后，如果这些 8 字节组是数据字符，那么会加上“01”标识；如果有一个或多个字节是控制字符，那么就会加上“10”标识。

在此编码技术中，将 8 个字节的字符（由 8B10B 编码定义，可能是数据或控制字符）编码为 64 比特长字符的过程通常被称为转换代码。GFP-T8 为组合 8B10B 的 8 比特为 64 比特字符提供了标准方法。而 10GbE 为 10G 以太网和 10Gbit/s 光纤链路提供了相关子集的映射表。用于同步定位的 “01”和“10”比特不参与扰码过程。这是因为其它比特在扰码后可能取任何值，只有同步比特“01”和“10”在经历长途传输后基本保持不变。同步比特 还可以保证每隔 66 比特至少会发生一次转换。

- CEI-P 编码技术。

这种技术由 OIF 定义。它的系统开销与 64B66B 编码相等，大约为 3%。当然，它还有很多不同之处。

CEI-P 采用边扰码，特征多项式为 $x^{17}+x^{14}+1$。这样做的优点是可以有效防止误码复制的产生（扰码的状态不会受之前产生的误码影响），缺点是需要在发送与接收之间同步扰码状态。

当边扰码与线路误码无关时，如果发送数据与扰码值相同（或恰好相反），那么扰码器会输出非常长的“1”或“0”序列。而这种非常长的扰码输出序列比短的序列对误码的容忍能力更强。

CEI-P 编码采用帧同步取代定位同步。64B66B 采用同步比特实现定位，而 CEI-P 将 24 个 64 比特码字看作一帧，这样用 1 个比特就可以实现数据或控制字符的判定。附加的 24 比特用于误码校验与信令。其中误码校验使用 20 比特，采用 fire-code 技术，可以纠正长达 7 比特的突发误码串。

- 10GBase-KR 编码技术。

这种技术的系统开销与 CEI-P 一致，都是 3%，主要的不同在于帧的长度是 32 个码字，而不是 24 个码字。这样一来纠错码长度就是 32 比特，可以纠正比 CEI-P 更长的突发误码串。

而它采用的扰码规则与 10GbE 相似。只不过它的扰码多项式与 10GBase-R 一致，寄存器采用的初始序列为“010101……”，而且每一帧重置一次。

- Interlaken PHY 编码技术。

Interlaken 编码技术的系统开销为 4.5%（64/67）。它的码字基于 64 比特。与其它低开销编码技术相比，Interlaken 的主要不同之处在于：

同步信号为 3 个比特，其中两个比特用于区分数据与控制，一个比特用于标识数据是否转化。数据转化比特的目的与 8B10B 类似，都是为了保证直流平衡；

Interlaken 采用与 10GbE 相同的边扰码特征多项式实现扰码。这样可以避免因采用自同步扰码器所导致的误码复制问题。一个同步码字用于传递扰码器状态，而且它非常长的扰码长度保证不会出现很长的连“0”或者连“1”；

表 6 对比了以上讨论的几种编码技术的异同，主要包括以下几点：

实际采用情况；

系统开销；

转换密度与直流平衡；

同步定位；

误码（保护、校验、复制）；

硬件复杂度（门数目）。

Table 2. Coding method comparison

Encoding method	Deployed	Overhead (%)	Transition density DC balance	Alignment	Error	Complexity
64B66B	Yes	3.1	Scrambling self sync (58b) CID max 66; RD not bound	Sync = “01,” “10”	Protection none Replication (×3)	Low
CEI-P 64B65B	No	3.1	Scrambling—side (17b) CID not bound; RD not bound	Sync using training	Protection FEC (up to 7 bits); Replication none	High
.3ap (10G backplane)	Not yet	3.1	Scrambling—side (58b) CID not bound; RD not bound	Sync using framing N/M	Protection FEC (up to 11 bits); Replication none	Low–Med
Interlaken (64/67)	Not yet	4.6	Scrambling—side (58b) CID max 67; RD bound to ±66	Sync with 3 bits	Error detection (CRC) Replication none	Low–Med
8B10B	Yes	25.0	Coding CID max 5; RD bound ±2	Sync using comma	Coding protection Replication yes (small)	Very low

表 6.　编码技术的对比

虽然 8B10B 编码行之有效，而且被广泛采用，但是它高达 25%系统开销的缺点在未来数百吉比特链路与系统应用中显得越来越突出。现在已经有若干种低开销的编码技术，它们具备不同的优势与缺点。但是到目前为止，还没有哪种低开销技术能够脱颖而出，成为继 8B10B 之后被广泛采用的首选技术。不过，本文所提到的几种技术都以提高硬件复杂度为代价，以满足低开销的要求。