8B/10B 编码的基本原理

8B/10B 编码是目前高速串行通信中经常用到的一种编码方式，其目的就是通过将一个字节宽度的数据经过映射机制转化为 10 位宽度的字符，来平衡位流中 0 与 1 的个数，也就是达到平衡直流的作用。由于直接将 8bit 数据编码成 10bit 来传输在实现上将占用芯片的大片物理面积， 并且严重影响了数据的传输速率， 所以目前大都采用将一串 8 位二进制数分为低 5 位和高 3 位，然后对低 5 位进行 5B/6B 编码，高 3 位进行 3B/4B 编码，最后再将 6 位和 4 位合在一起的编码方式。 这样做不进减少了芯片占用面积， 而且简化了编码， 提高了数据的传输速率。

通常用字符 HGFEDCBA 来表示编码前的 8 位二进制数，则低 5 位就是 EDCBA，高 3 位就是 HGF。5B/6B 编码后 6 位二进制数的表示方式为 abcdei，而 3B/4B 编码后 4 位二进制数的表示方式为 fghj，最后合成的 10 位二进制数为 abcdeifghj。人们喜欢把 8bit 数据表示成 Dx.y 的形式，而控制代码用 Kx.y 的形式，其 x=5LSB(least significant bit 最低有效位)，y=3MSB(most significant bit 最高有效位)。它们之间的对应关系如图 1 所示。

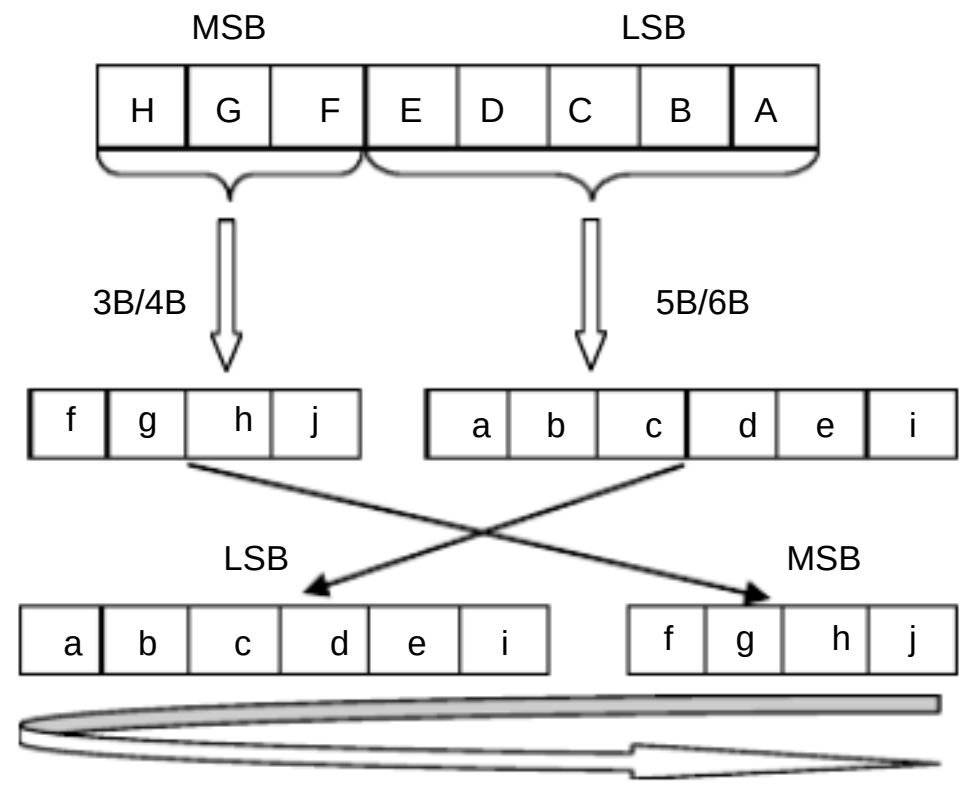

图 1 编码对应关系

8B/10B 编码中有两个重要的概念，不均等性（ disparity ）和极性偏差（ running disparity ， RD）。前者表示‘ 1’和‘ 0’个数的差值，其有如下规律：

（1）若 ABCDE 的 Disparity 值为 - 1，那么在‘ RD- ’项中生成的 abcde 与 ABCDE 有一一对应的关系 ，并且 i=‘ 1’，此时除 ABCDE= ‘ 00011’外 abcde i 是唯一的 ；

（2）若 ABCDE 的 Disparity 值为 + 1，那么在‘ RD- ’项中生成的 abcde 与 ABCDE 有一一对应的关系 ，并且 i=‘ 0’，此时除 ABCDE = ‘ 11100’外 abcdei 是唯一的 ；

（3）若 ABCDE 的 Disparity 值为 + 3，那么在‘ RD- ’项中生成的 abcde 与 ABCDE 有一一对应的关系 ，并且 i=‘ 0’，此时 abcdei 是互为反码的两个值 ；

（4）若 ABCDE 的 Disparity 为其他值 ，此时 abcde i 具有互为反码的两个值 ，对这些特殊的‘ RD －’项中的值可直接用查表法实现。

后者的取值分为以下三种：

（1）当码字中‘ 1’比‘ 0’多，或者 4B 码为 1100,或者 6B 码为 111000 时，该码字被定义为正极性码 ,这个时候 RD 取正；

（2）当码字中‘ 1’比‘ 0’少，或者 4B 码为 0011,或者 6B 码为 000111 时，该码字被定义为负极性码 ，这个时候 RD 取负；

（3）除了上述两种情况其他码字定义为中性码 , RD 取其前一码字的 RD 游程值。为了直观

了解这个规则 ，RD 取值图如下 ：

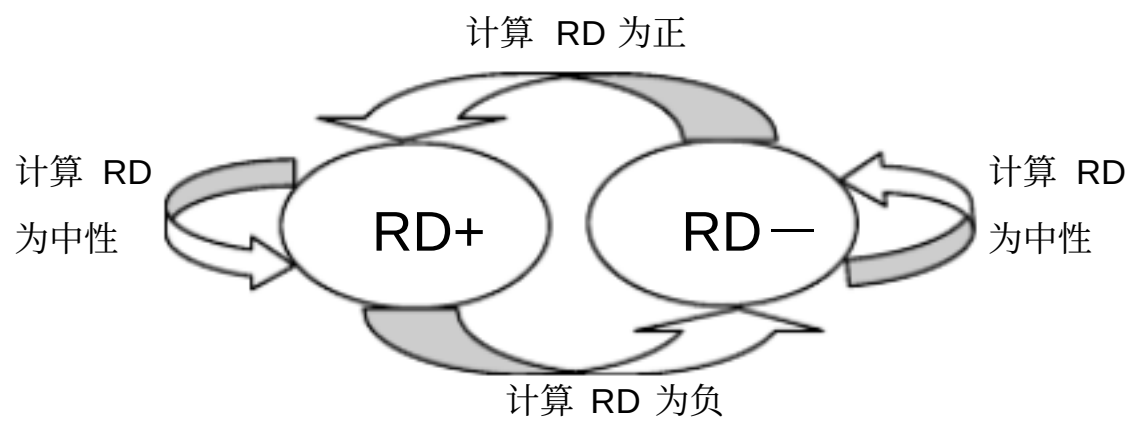

图 2 RD 值计算方式

3B/4B 和 5B/6B 是通过映射机制进行编码的， 这种映射机制已经标准化成相应的映射表， 如表 1（略），

3B/4B 和 5B/6B 编码还需要通过映射控制模块（ Disparity Control ）的控制才能最终完成编码。通过表 1 我们可以看出， 5 位映射到 6 位可能存在两种编码，这两种编码是互为反码的。 在 5B/6B 编码中， 先预设 RD 为负， 然后与实际的 RD 值进行比较， 如果实际值为负，则输出 RD– 栏中的编码，否则，将 RD– 栏中的编码取反后输出，即输出 RD+栏中的编码。5B/6B 编码的实际流程如图 3。3B/4B 编码由于数据量少，可以把映射数据直接存储，然后通过查表实现。

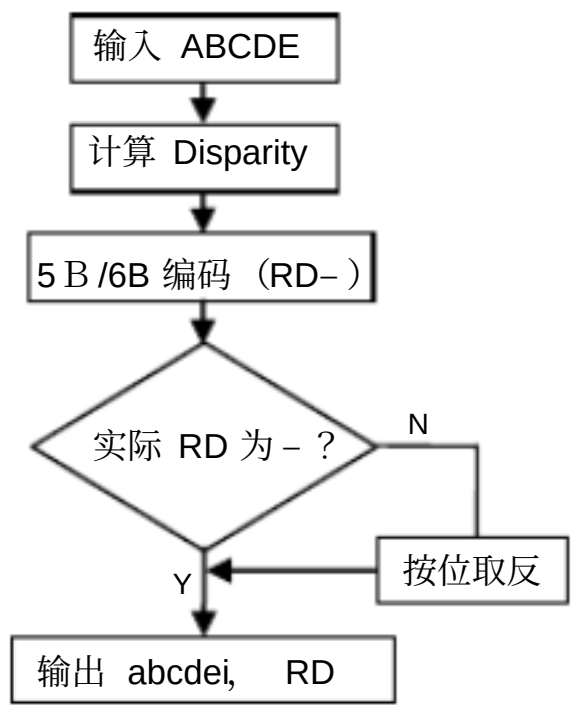

图 3 5B/6B 编码实际流程图

有了 3B/4B 和 5B/6B 编码，那么就可以实现 8B/10B 编码，其实际流程如图 4 所示。

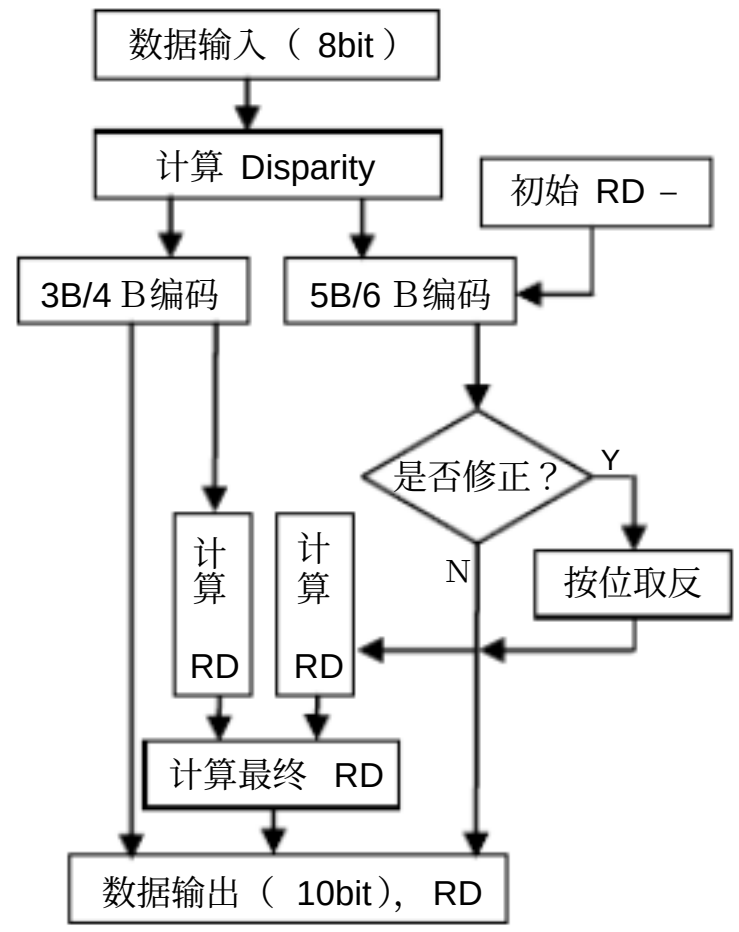

图 4 8B/10B 编码整体流程图

RD 运算时基于每个子模块的 D isparity 属性的，所以必须首先对输入字符 DX 的高三位 FGH 和低五位 ABCDE 分别进行 D isparity 计算。通过分析总结出低五位和高三位编码前后的 RD 值变化情况，其中 rdout 表示新生成的 RD 值而 rdin 表示当前的 RD 值:

(1) ABCDE 的 Dispar ity 计算为 + 1 时，若 ABC DE =‘ 00 111 ’且控制字符 K 无效，则 rdout= not rdin, 否则 rdout= rd in;

(2) ABCDE 的 Disparity 计算为 - 1 时，若 ABCDE = ‘ 00 011 ’则 rdout= no t rdin, 否则 rdou t= rd in;

(3) ABCDE 的 Disparity 计算为其他值时 , rdout= not rdin;

(4) FGH 的值为‘ 000 ’、‘ 001 ’或‘ 111 ’是 rdout= no t rdin, 否则 rdout= rd in 。

5B/6B 编码中，计算实际 RD 值的方法也是如此。

计算最终 RD 值的方法是，如果 3B/4B 编码后的 RD 和 5B/6B 编码后的 RD 的极性相同，那么最终要输出的 RD 和初始输入的 RD 一致，否则相反。具体如表 2 所示。

初始输入的 RD	3B/4B 编码后的 RD	5B/6B 编码后的 RD	最终要输出的 RD
RD-	RD-	RD-	RD-
RD-	RD-	RD+	RD+
RD-	RD+	RD-	RD+
RD-	RD+	RD+	RD-
RD+	RD-	RD-	RD+
RD+	RD-	RD+	RD-
RD+	RD+	RD-	RD-
RD+	RD+	RD+	RD+

表 2

8B/10B 标准中使用了 12 个特殊的控制代码，它们采用查表的方式进行编码。

Control symbols

input		RD = −1	RD = +1
	HGF EDCBA	abcdei fghj	abcdei fghj
K.28.0	000 11100	001111 0100	110000 1011
K.28.1 †	001 11100	001111 1001	110000 0110
K.28.2	010 11100	001111 0101	110000 1010
K.28.3	011 11100	001111 0011	110000 1100
K.28.4	100 11100	001111 0010	110000 1101
K.28.5 †	101 11100	001111 1010	110000 0101
K.28.6	110 11100	001111 0110	110000 1001
K.28.7 ‡	111 11100	001111 1000	110000 0111
K.23.7	111 10111	111010 1000	000101 0111
K.27.7	111 11011	110110 1000	001001 0111
K.29.7	111 11101	101110 1000	010001 0111
K.30.7	111 11110	011110 1000	100001 0111