不使用查找表的高速统一8B/10B编码器

蒋毅飞 李俊 范好好
上海高性能集成电路设计中心 上海 201204

摘要：
8B/10B编码在USB 3.0和众多串行总线中得到了广泛应用，其主要由数据码和控制码两部分构成。通过深入分析8B/10B编码、尤其是控制码的特点，提出了控制与数据融合的编码方法，将控制码编码统一于数据码编码过程之中，避免了对控制码的单独编码，简化了编码过程。在此基础上实现了不使用查找表的统一8B/10B编码器。实验结果表明，在某主流工艺下该编码器运行频率可达3.0GHz以上，其电路面积开销较使用查找表的编码器降低了80%左右。

关键词：串行通信，USB3.0，8B/10B编码

1. 引言

USB 3.0为了提高传输速率，放弃了USB 2.0所采用的NRZI（Non Return to Zero Invert，NRZI）编码，采用了应用更为广泛的8B/10B编码。除USB 3.0外，包括PCI Express 2.0、HyperTransport、InfiniBand、IEEE 1394b、SATA等在内的众多串行总线都采用了8B/10B编码。8B/10B编码将输入的8比特码字（Code）转换为10比特的符号（Symbol），并定义了12个控制码。8B/10B编码表包括数据编码和控制编码两部分。本文深入分析了8B/10B控制编码与数据编码的异同，提出了控制与数据融合的编码方法，将控制编码统一于数据编码过程中，从而避免了常规实现方法中对控制码的单独编码[1]，简化了8B/10B编码过程。在此基础上，实现了不使用查找表的统一8B/10B编码器，其运行频率和电路面积等指标较使用查找表的编码器均有显著提升。

2. 8B/10B编码技术

8B/10B编码由IBM公司在1983年提出[2]，并于后来成为专利[3]，目前已经广泛应用于各种高速串行总线。8B/10B编码技术将输入的8比特码字经过某种映射机制转换为10比特的符号（下文统一将编码输入称为码字，将编码输出称为符号）。8B/10B编码定义了256个数据码，用D表示；还定义了12个控制码，用K表示。对同样一个输入码字，用D/K区分是数据码还是控制码。例如D00011100表示一个数据码，而K00011100则表示一个控制码。

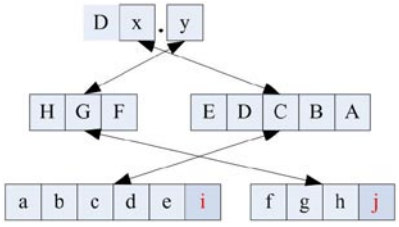

图1 8B/10B编码的结构

为了表述方便，8B/10B编码将输入的8比特码字（HGFEDCBA）表示成Dx.y（或者Kx.y），其中x对应于输入码字的低5位（EDCBA），y对应于输入码字的高3位（HGF）。编码时对低5位码字（x）和高3位码字（y）分别进行编码。x（EDCBA）经过5B/6B编码输出符号abcdei，y（HGF）经过3B/4B编码输出符号fghj。8B/10B的编码结构如图1所示。

1.1 偏差和运行时偏差

8B/10B编码具有较好的直流平衡（DC Balance）特性，可使发送序列中的“0”、“1”数量保持一致，且连续的“0”或者“1”基本上不超过5位。为了实现直流平衡，8B/10B编码对3B/4B编码和5B/6B编

本文曾发表于2014年计算机工程与工艺年会论文集；
基金项目：2013年核高基课题“超级计算机处理器研发”（2013ZX01028-001-001-001）

码的偏差（Disparity）进行了严格的控制。

所谓偏差，是指一个序列中“1”的个数与“0”的个数之间的差异。如果“1”的个数比“0”的个数多，用“+”表示；如果“1”的个数比“0”的个数少，用“-”表示；如果两者数量相等，则用“0”表示。如果一个序列的偏差为0，则表示该序列是平衡的。8B/10B编码要求3B/4B编码输出和5B/6B编码输出的偏差只能是0、+2或者-2。此外编码规则还要求两个连续编码序列的偏差极性必须翻转。譬如，如果编码符号abcdei的偏差为+2，则要求fghj的偏差为-2。如此，前一个符号中多出的2个“1”（或“0”）可被后一个符号中多出的2个“0”（或者“1”）补偿，以保证编码的总体平衡。

5B/6B和3B/4B的编码输出，分别有C_6^3和C_4^2种平衡编码（如000111、0101等）。由于5比特输入可表示32个不同码字、3比特输入可表示8个不同码字，所以无论是5B/6B编码还是3B/4B编码，有些码字必然分配不到平衡编码。8B/10B编码为无法分配到平衡编码的码字分配了一对互补的输出符号，即1个偏差为+2的符号和1个偏差为-2的符号。

运行时偏差（Running Disparity，RD）又称极性偏差，有“+”、“-”两种取值。在对一个码字进行编码前，需根据当前的运行时偏差在一对互补的编码符号中选择一个合适的符号。

1.2 8B/10B编码表

如图1所示，8B/10B编码分为5B/6B编码和3B/4B编码两个编码模块，它们的编码表分别如表1和表2所示。

表1 5B/6B编码

名称	EDCBA	K	D-1	abcdei	D0	备选abcdei
D.0	00000	0	+	011000	-	100111
D.1	00001	0	+	100010	-	011101
D.2	00010	0	+	010010	-	101101
D.3	00011	0	×	110001	0	
D.4	00100	0	+	001010	-	110101
D.5	00101	0	×	101001	0	
D.6	00110	0	×	011001	0	
D.7	00111	0	-	111000	0	000111
D.8	01000	0	+	000110	-	111001
D.9	01001	0	×	100101	0	
D.10	01010	0	×	010101	0	
D.11	01011	0	×	110100	0	
D.12	01100	0	×	001101	0	
D.13	01101	0	×	101100	0	
D.14	01110	0	×	011100	0	
D.15	01111	0	+	101000	-	010111
D.16	10000	0	-	011011	+	100100
D.17	10001	0	×	100011	0	
D.18	10010	0	×	010011	0	
D.19	10011	0	×	110010	0	
D.20	10100	0	×	001011	0	
D.21	10101	0	×	101010	0	
D.22	10110	0	×	011010	0	
D/K.23	10111	×	-	111010	+	000101
D.24	11000	0	+	001100	-	110011
D.25	11001	0	×	100110	0	
D.26	11010	0	×	010110	0	
D/K.27	11011	×	-	110110	+	001001
D.28	11100	0	×	001110	0	
D/K.29	11101	×	-	101110	+	010001
D/K.30	11110	×	-	011110	+	100001
D.31	11111	0	-	101011	+	010100
K.28	11100	1	-	001111	+	110000

表2 3B/4B编码

名称	HGF	K	D-1	fghj	D0	备选fghj
D/K.0	000	×	+	0100	-	1011
D.1	001	0	×	1001	0	
D.2	010	0	×	0101	0	
D/K.3	011	×	-	1100	0	0011
D/K.4	100	×	+	0010	-	1101
D.5	101	0	×	1010	0	
D.6	110	0	×	0110	0	
D.P7	111	0	-	1110	+	0001
D.A7*/K.7	111	×	-	0111	+	1000
K28.1	001	1	+	1001	0	0110
K28.2	010	1	+	0101	0	1010
K28.5	101	1	+	1010	0	0101
K28.6	110	1	+	0110	0	1001

*如果是数据编码，在RD>0且(e=i=0)或者RD<0且(e=i=1)时，用A7替换P7。

表1和表2中，D-1表示运行时偏差RD。如D-1为“×”，则表明运行时偏差可以是“+”，也可以是“-”。abcdei/fghj表示当运行时偏差等于D-1指定的值时编码输出的符号，D0表示编码输出符号abcdei/fghj的偏差。如果运行时偏差与D-1指定的值相反，则输出备选符号，此时符号的偏差与表1、表2中D0指示的偏差相反。

表3列出了12个控制码在RD为“-”、“+”两种情况下编码输出的符号abcdei_fghj以及符号对应的偏差D0。

表3 控制编码

名称	HGF_EDCBA	K	D-1	abcdei_fghj	D0	D-1	备选 abcdei_fghj	D0
K28.0	000_11100	1	-	001111_0100	0	+	110000_1011	0
K28.1	001_11100	1	-	001111_1001	+	+	110000_0110	-
K28.2	010_11100	1	-	001111_0101	+	+	110000_1010	-
K28.3	011_11100	1	-	001111_0011	+	+	110000_1100	-
K28.4	100_11100	1	-	001111_0010	0	+	110000_1101	0
K28.5	101_11100	1	-	001111_1010	+	+	110000_0101	-
K28.6	110_11100	1	-	001111_0110	+	+	110000_1001	-
K28.7	111_11100	1	-	001111_1000	0	+	110000_0111	0
K23.7	111_10111	1	-	111010_1000	0	+	000101_0111	0
K27.7	111_11011	1	-	110110_1000	0	+	001001_0111	0
K29.7	111_11101	1	-	101110_1000	0	+	010001_0111	0
K30.7	111_11110	1	-	011110_1000	0	+	100001_0111	0

3. 控制与数据融合的8B/10B编码方法

8B/10B编码定义了12个控制码，如表3所示。仔细考察8B/10B编码表可发现，在表2和表1中分别为控制码的高低两段定义了3B/4B编码和5B/6B编码。这表明，在形式上存在控制编码与数据编码融合的可能性，即控制编码也采用图1所示的分块编码方式。

然而，以控制码K28.0为例，如果采用分块编码方式，其编码过程如下（假定当前RD为-）：

1 在表1中查找K28编码，当前RD为-，选择abcdei=001111；

2 编码符号中“1”比“0”多2个，故符号偏差为+2（表1中用D0=+表示）；

3 因为此时符号偏差为+2，所以当前运行时偏差RD为+；

4 在表2中查找K0编码，当前RD为+，选择fghj=0100；

5 编码符号中“1”比“0”少2个，故符号偏差为-2（表2中用D0=-表示）。

完成上述步骤后，K28.0编码输出符号abcdei_fghj=001111_0100，符号偏差D0为-。然而如果查找表3，编码输出符号abcdei_fghj=001111_0100，符号偏差D0=0。事实上，K28.0的编码是平衡编码，符号偏差应该为0。显然，按照分块编码方式对K28.0编码得到的符号偏差D0是错误的。这或许是目前几乎所有实现方案都对控制码进行独立编码的原因[1, 4, 6-7]。

3.1 统一运行时偏差控制

运行时偏差，顾名思义即运行到当前时刻的偏差，是一个累计偏差。如果运行时偏差RD为+，则编码时应选择偏差为“-2”的符号；如果运行时偏差RD为-，则编码时应选择偏差为“+2”的符号。运行时偏差实际上决定了当前符号偏差的极性，因此运行时偏差又称为极性偏差，仅有“+”、“-”两种取值。

运行时偏差与偏差既有联系，又有区别。再以K28.0编码为例，假定当前RD为-，采用分块编码方式：

1 在表1中查找K28编码，当前RD为-，选择abcdei=001111；

2 编码符号中“1”比“0”多2个，故符号偏差为+2（表1中用D0=+表示）；

3 因为此时符号偏差为+2，所以当前运行时偏差RD为+；

4 在表2中查找K0编码，当前RD为+，选择fghj=0100；

5 编码符号中“1”比“0”少2个，故符号偏差为-2（表2中用D0=-表示）；

6 因为此时符号偏差为-2，所以当前运行时偏差RD为-。

上述分析表明，采用分块编码方式对K28.0编码时，其最后输出的D0=-实际上是高三位码字K.0编码后的符号偏差，同时也是K28.0编码后的运行时偏差。而直接通过查找表3对K28.0编码，输出的D0=0仅仅是编码后10位符号的偏差，并不反应编码后的运行时偏差。由于平衡编码需要传递之前的运行时偏差，所以通过查找表3对K28.0编码后的运行时偏差应当和对K28.0编码前的运行时偏差保持一致，即RD=-。由此看出，对K28.0采用分块编码和采用查找表3直接编码所得到的编码符号以及编码后运行时偏差是完全一致的。采用类似过程，同样可一一证明对每个控制码，采用分块编码和采用查找表3直接编码，所得到的编码符号以及编码后运行时偏差彼此相同。

由于编码过程中关注的是编码符号和运行时偏

差是否正确，而不关注单个符号的偏差，所以控制编码具备和数据编码融合的可行性。控制编码和数据编码融合的前提是统一的运行时偏差控制。

表4 统一运行时偏差控制

编码前RD	符号偏差	编码后RD
-	0	-
-	±2	+
+	0	+
+	±2	-

从上述分析可知，运行时偏差取决于编码前运行时偏差和编码符号的偏差。如果符号偏差不为0，则运行时偏差取反；如果符号偏差为0，则运行时偏差保持不变。统一运行时偏差控制规则具体如表4所示。

3.2 控制与数据融合的编码表

在统一运行时偏差控制的基础上，依据8B/10B原x始编码表，本文重新构造了包含运行时偏差控制的融合编码表，对输入码字的低5位和高3位分别编码，如表5和表6所示。

表5和表6分别列出了在RD为“-”、“+”两种情况下编码输出的符号abcdei/fghj以及编码后的运行时偏差RD’。 RD’=RD表示编码后运行时偏差保持不变；RD’=-RD表示编码后运行时偏差发生翻转。

表5 融合的5B/6B编码

输入		abcdei输出		RD’	输入		abcdei输出		RD’
D/K.x	EDCBA	RD-	RD+		D/K.x	EDCBA	RD-	RD+	
D.0	00000	100111	011000	-RD	D.16	10000	011011	100100	-RD
D.1	00001	011101	100010		D.17	10001	100011		RD
D.2	00010	101101	010010		D.18	10010	010011		
D.3	00011	110001		RD	D.19	10011	110010		
D.4	00100	110101	001010	-RD	D.20	10100	001011		
D.5	00101	101001		RD	D.21	10101	101010		
D.6	00110	011001			D.22	10110	011010		
D.7	00111	111000	000111		D/K.23	10111	111010	000101	-RD
D.8	01000	111001	000110	-RD	D.24	11000	110011	001100	
D.9	01001	100101		RD	D.25	11001	100110		RD
D.10	01010	010101			D.26	11010	010110		
D.11	01011	110100			D/K.27	11011	110110	001001	-RD
D.12	01100	001101			D.28	11100	001110		RD
D.13	01101	101100			D/K.29	11101	101110	010001	-RD
D.14	01110	011100			D/K.30	11110	011110	100001	
D.15	01111	010111	101000	-RD	D.31	11111	101011	010100	
					K.28	11100	001111	110000	-RD

表6 融合的3B/4B编码

输入		fghj输出		RD’
D/K.x	HGF	RD-	RD+	
D/K.0	000	1011	0100	-RD
D.1	001	1001		RD
K.1		0110	1001	
D.2	010	0101		
K.2		1010	0101	
D/K.3	011	1100	0011	
D/K.4	100	1101	0010	-RD
D.5	101	1010		RD
K.5		0101	1010	
D.6	110	0110		
K.6		1001	0110	
D.P7	111	1110	0001	-RD
D.A7/K.7*		0111	1000	

*如果是控制编码，或者(RD>0且(e=i=0) || RD<0且(e=i=1))，选择该行编码。

4. 统一8B/10B编码器的实现

前文分析表明，控制编码和数据编码可以融合于图1所示的同一编码模型。在此思想指导下，本文依据控制与数据融合的编码表，实现了统一8B/10B编码器，其结构如图2所示。

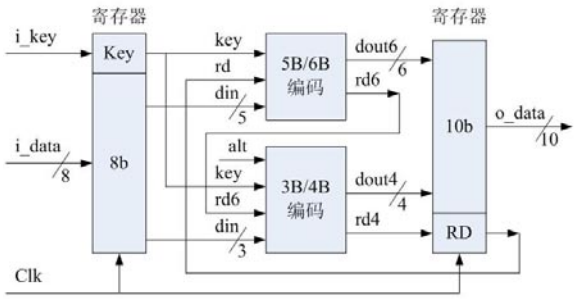

图2 统一8B/10B编码器结构图

图2中，输入信号（包括输入码字i_data和控制标志i_key）首先进入寄存器，编码模块输出的符号

（dout6和dout4）经下一级寄存器后输出。图2中除输入、输出寄存器外，所有逻辑均采用组合电路实现，整个编码过程在1个时钟周期内完成。

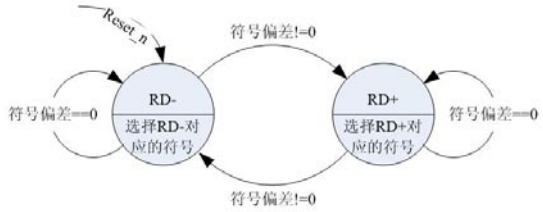

图3 编码过程及运行时偏差转换图

编码时，5B/6B编码模块在key和当前运行时偏差rd的控制下首先对低5位码字编码，输出符号dout6和运行时偏差rd6（从表6可看出，对输入码字7，有主要符号P7和备用符号A7。图2中alt信号表示是否需要用备用符号代替主要符号，其生成逻辑见表6说明）。3B/4B编码模块在key和当前运行时偏差rd6的控制下对高3位码字编码，输出符号dout4和运行时偏差rd4。3B/4B编码模块输出的运行时偏差rd4经寄存器寄存后，作为下一次编码的运行时偏差。运行时偏差的转换如图3所示。

由于8B/10B编码映射关系固定，比较适合用查找表实现编码过程，因此许多编码方案都采用查找表方式实现[5-6]。为了便于对比，本文也实现了基于查找表的8B/10B编码。查找表方案为表5定义了14×32的存储单元，分别存储32个码字在RD=+和RD=-时的符号，以及K=0和K=1时运行时偏差RD是否翻转的标志。此外，还为K28定义了一个14位宽的伴随条目。当输入码字为28时，伴随条目同时输出，依据控制标志K在两个输出条目之间进行选择。查找表方案为表6中的数据码字和控制码字各定义了9×8的存储单元，分别存储8个码字在RD=+和RD=-时的符号以及运行时偏差RD是否翻转的标志。此外，还为D7定义了一个9位宽的伴随条目。

5. 实验结果与分析

5.1 电路仿真结果

本文用Verilog语言实现了统一8B/10B编码器的RTL代码，用Cadence NC-Sim对设计代码进行仿真，以验证设计功能的正确性。仿真时依次输入8B/10B编码表所定义的256个数据码字和12个控制码字，仿真波形如图4所示。

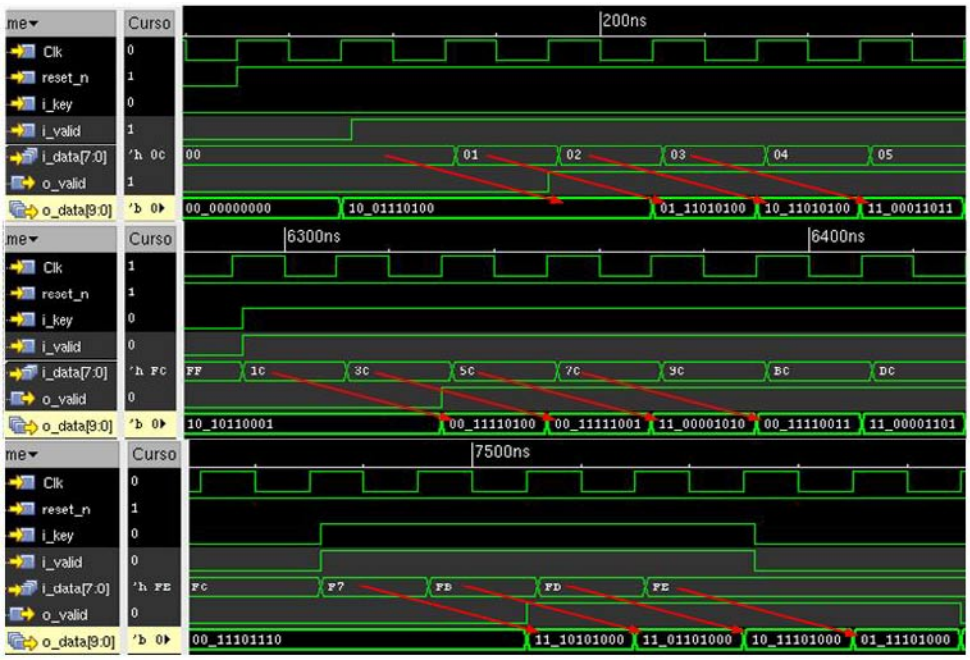

图4 编码波形

图4顶部为数据码字D0.0、D1.0、D2.0和D3.0的编码波形。从图中可看出，输入码字经2个时钟周期后输出编码符号。上述4个数据码字的输出符号依次为2’b100111_0100、2’b011101_0100、2’b101101_0100、2’b110001_1011。

图4中部为控制码字K28.0～K28.7的部分编码波形。从图中可看出，对K28.0～K28.4编码得到的符号依次为2’b001111_0100、2’b001111_1001、2’b110000_1010、2’b001111_0011、2’b110000_1101。从输出结果可看出，K28.0是平衡编码，传递其编码前运行时偏差RD-给K28.1；K28.1是非平衡编码，编码后运行时偏差极性翻转，从而选择K28.2的

符号为2’b110000_1010（注意K28.1和K28.2符号的高6位互为补码）。由于控制码也采用了分块编码方式，该波形同时证明了统一运行时偏差控制的正确性。

图4底部为控制码字K23.7、K27.7、K29.7和K30.7的编码波形，编码得到的符号依次为2’b111010_1000、2’b110110_1000、2’b101110_1000、2’b011110_1000。由于这四个控制字均为平衡编码，所以运行时偏差保持为RD-未发生改变。

USB 3.0协议附录A中列出了8B/10B编码的256个数据码和12个控制码的编码表。本文设计的编码器，对全部268个码字的编码结果与8B/10B编码表保持一致。

2.2 电路综合结果

本文分别用Synopsys Design Compiler和IC Compiler在某主流工艺下对设计代码进行综合。作为对比，本文也在相同工艺下对使用查找表的8B/10B编码器进行了综合，其查找表采用带复位的触发器实现。综合结果分别如表7和表8所示。

表7列出了两种设计方案的Design Compiler综合结果。从表7可看出，当时钟周期设置为0.4ns时，两种设计方案均能满足时序要求；而当时钟周期设置为0.33ns时，使用查找表的编码器出现了违反时序要求的路径。在两种时钟周期设置下，不使用查找表编码器的电路单元面积分别为使用查找表编码器的12.28%和15.04%。

表7 DC综合结果

对比方案	时钟周期/ns	逻辑层数	关键路径		总计负Slack	违反路径数目	单元面积/um²		
			长度	Slack			总单元面积=组合逻辑+非组合逻辑		
查找表方案	0.40	17	0.31	0.00	0.00	0	2756.43	1334.47	1421.96
本文方案	0.33	10	0.31	0.00	0.00	0	338.51	233.02	105.49
查找表方案		12	0.28	-0.05	-0.38	9	3592.74	2096.34	1496.40
本文方案		8	0.23	0.00	0.00	0	540.31	428.12	112.19

表8 ICC综合结果

对比方案	时钟周期/ns	逻辑层数	关键路径/ps			后端可实现时钟周期	单元面积/um²		
			长度	考虑建立时间	考虑时钟偏斜		总单元面积=组合逻辑+非组合逻辑		
查找表方案	0.25	17	302	302+23=325	357.5	——	4291.22	2532.40	1758.82
本文方案	0.30	12	244	244+21=265	291.5	——	908.81	741.11	167.70
查找表方案		15	300	300+12=312	343.2	350	4252.65	2512.17	1740.48
本文方案		12	239	239+11=250	275.0	280	833.08	692.19	140.88

IC Compiler的综合结果如表8所示。从表8可看出，当时钟周期设置为0.25ns时，两种设计方案因关键路径延时超过时钟周期而不具可实现性。当时钟周期设置为0.30ns时，不使用查找表的编码器关键路径长度为239ps，考虑建立时间以及10%的时钟偏斜，合理的时钟周期设置在280ps左右。在0.30ns的时钟周期设置下，不使用查找表编码器的电路单元面积仅为使用查找表编码器的19.59%。

6. 结束语

本文总结了8B/10B编码的原理与过程，深入分析了运行时偏差与偏差的联系与区别，在此基础上提出了统一运行时偏差控制方案以及控制与数据融合的编码方法，避免了对控制码的单独编码，从而简化了编码过程。本文实现的统一8B/10B编码器，摒弃了传统8B/10B编码器中使用查找表编码的方法，用组合电路在单周期内完成8B/10B编码，其运行频率在3.0GHz以上，而其电路面积开销仅为使用查找表编码器的20%左右。鉴于8B/10B编码在当前高速串行接口中的广泛应用，本文的工作将对现有的硬件电路产生积极影响。

参考文献:

[1] WANG Qi, HUA Si-liang, WANG Dong-hui. A 1.1GHz 8B/10B encoder and decoder design[C] //Proc of the Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia 2010), 2010: 138-141.

[2] Widmer A.X, Franaszek P.A. A DC - balanced, partitioned - block, 8B/10B transmission code[J]. IBM Journal of Research and Development, 1983, 27(5): 440 - 451.
[3] Franaszek P.A, Widmer A.X. Byte oriented DC balanced (0,4) 8B/10B partitioned block transmission code: U.S, 4486739[P]. 1984 - 12 - 4.
[4] Lattice Semiconductor. 8b/10b encoder/decoder[EB/OL].[2012 - 2 - 13].
www.latticesemi.com/~/media/Documents/ReferenceDesigns/1D/8B/10BEncoderDecoder - Documentation.PDF
[5] Abiri S, Sahebdel S. A method for implementation of the DC - balanced 8B/10B coding used in SuperSpeed USB[C] //Proc of the 1st International Conference on Integrated Intelligent Computing (ICIIC 2010), 2010: 68 - 72.
[6] 申长伟，赵士坤. 光纤通道中8B/10B编解码模块的实现[C]//全国抗恶劣环境计算机第20届学术年会论文集， 2010: 289 - 293.
[7] 李宥谋. 8B/10B编码器的设计及实现[J]. 电讯技术, 2005, 45(6): 26 - 32.

要闻集锦

美能源部投资研究下一代超级计算机算法

据www.hpcwire.com网站2016年9月13日消息报道，美国能源部近日投资了一项240万美元的项目，旨在开发用于解决线性和非线性方程式的新型计算机算法，该项目由美国佐治亚理工学院领导，圣地亚国家实验室、天普大学和田纳西大学都参与其中。该技术最终有望为新一代超级计算机的开发铺平道路。

该研究将在千万亿次迈向百亿亿次级计算的过渡期采用更先进的算法，目前，同步操作已阻碍了数学工具的发展，由于处理器必须按顺序执行计算，因此这种操作已成为计算的瓶颈。新提出的方法提供了一种“异步”技术，允许每个处理器独立操作，这样就可以对最近可用数据进行处理，而不是等待其他处理器的同步数据。

研究人员认为，同构建更强大的超级计算机相比，设计出适合在这些超级计算机上运行的算法和应用更为重要。目前佐治亚理工学院的专家正致力于开发、测试适合的异步技术。

这项为期三年的计划是美国在2023年前构建百亿亿次超级计算机计划的一部分。该研究将对一系列应用带来重大影响，如大规模材料科学、气候研究、燃烧模拟等。另外，它还将改变目前对并行计算的理解。

（来源：《高性能计算技术》）