XAPP645 (v2.2) 2006 年 8 月 9 日

单纠错和双检错

作者：Simon Tam

提要

本应用指南介绍了 “纠错控制”(Error Correction Control, ECC) 模块在 Virtex™–II、Virtex–II Pro、Virtex–4 或 Virtex–5 器件中的实现。该设计可检测和纠正全部单位元错误 (single bit error) (在由 64 位数据和 8 个校验位或由 32 位数据和 7 个校验位组成的代码字内)，并可以检测数据中的双位元错误 (double bit error)。设计采用的是汉明码 (Hamming code)，这是用于 ECC 操作的一种简单而高效的代码。因此，该设计的性能卓越，并能提供非常高的资源利用率。

简介

对可靠度和性能要求较高的应用领域通常会使用检错和纠错功能。例如，企业的数据存储系统就是利用存储器缓存来提高系统可靠性的。缓存通常置于控制器内部主机接口与磁盘阵列之间。功能强大的缓冲存储器设计一般都包括 ECC 功能，避免客户数据中单位的缺失。ECC 作为一种重要功能已广泛应用于通信领域，如卫星接收机；与重新传输数据相比，纠错能大大降低成本，并提高效能。

本应用指南中介绍的参考设计 (XAPP645.zip) 在速度等级为 6 的 Virtex–II Pro 器件中执行检错与纠错时，无流水线情况下的数据读 / 写速率可达 144 MHz，有流水线时可达 313 MHz。它可以对代码字中任意位置的双位元错误进行检测，还可纠正单位元错误。该参考设计专门针对 72 位双数据速率 (DDR) DIMM 存储器。32 位和流水线版现均已推出。只需稍做调整，即可将此设计应用于较窄的数据宽度。

汉明码

本应用指南中介绍的 ECC 功能由汉明码完成，这是一种相对简单而功能强大的 ECC 代码。它通过多校验位 （ 奇偶校验位 ）来传输数据，并在接受数据进行检错时对关联的校验位进行解码。

校验位即平行奇偶校验位，根据原始数据字中的某些 XORing 位生成。如果代码字中产生了位元错误，在对检索到的代码字进行解码后，就会有数个校验位显示奇偶校验错误。这些校验位错误的组合即可显示错误的实质。另外，任何单位元错误的位置都可通过校验位加以识别。

汉明码字由原始数据和校验位 （ 奇偶校验位 ）组合而成，并通过一个有序集合 (d + p,d) 进行描述，其中 d 表示数据宽度，p 表示奇偶校验位宽度。奇偶校验矩阵 [P] 可表示为：

$$[P] = [D] \bullet [G]$$

其中，[D] 代表数据矩阵，[G] 代表生成矩阵。[G] 矩阵由单位矩阵 [I] 和创建矩阵 (creation matrix) [C] 组成。

$$[G] = [I:C]$$

例如，(7,4) 汉明码如下：

$$[G] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

纠正单位元错误所需的最少校验位数目按以下方程式计算：

$$D + P + 1 \leq 2^P$$

本参考设计使用的是 (72,64) 汉明码。也就是说，汉明码的字宽为 72 位，包括 64 个数据位和 8 个校验位。在 64 位字中纠正单位元错误，所需的最少校验位数为七。多出的那个校验位是为了扩展功能，来检测双位元错误。

检错时，代码字向量将与生成矩阵的转置相乘得出 8 位向量 [S]，即校正子向量。

$$[S] = [D,P] \bullet [G']$$

如果校正子向量各项均为零，说明无错误报告。其他所有非零的结果均代表位元错误的类型，并提供各个单位元错误的位置。这些信息然后用于纠正原始输入数据。

为了形象了解汉明码，请看以下表格。所有数据位的位置及校验位都在图 1 校正子表中作了标示。表中各单元格的具体位置由其所在行和列确定。例如，数据位 60 位于 100 列 1000 行，即所在位置为 1000100。通过计算各个位置的位元的奇偶性（或奇或偶），即可得出七个校验位。校验位方程式由 XOR 操作符构成，表示为 ⊕ 。例如，校验位 1 (CB1) 的逻辑方程式如下：

CB1 = D0 ⊕ D1 ⊕ D3 ⊕ D4 ⊕ D6 ⊕ D8 ⊕ D10 ⊕ D11 ⊕ D13 ⊕ D15 ⊕ D17 ⊕ D19 ⊕ D21 ⊕ D23 ⊕ D25 ⊕ D26 ⊕ D28 ⊕ D30 ⊕ D32 ⊕ D34 ⊕ D36 ⊕ D38 ⊕ D40 ⊕ D42 ⊕ D44 ⊕ D46 ⊕ D48 ⊕ D50 ⊕ D52 ⊕ D54 ⊕ D56 ⊕ D57 ⊕ D59 ⊕ D61 ⊕ D63

基本上，所有表格单元位置 1（最低位）的数据位被挑选出来，组成 CB1（参见图 1）。CB2 包含次低位到最低位，依次类推。

111	110	101	100	011	010	001	000	
D63	D62	D61	D60	D59	D58	D57	*CB7*	1000
D56	D55	D54	D53	D52	D51	D50	D49	0111
D48	D47	D46	D45	D44	D43	D42	D41	0110
D40	D39	D38	D37	D36	D35	D34	D33	0101
D32	D31	D30	D29	D28	D27	D26	*CB6*	0100
D25	D24	D23	D22	D21	D20	D19	D18	0011
D17	D16	D15	D14	D13	D12	D11	*CB5*	0010
D10	D9	D8	D7	D6	D5	D4	*CB4*	0001
D3	D2	D1	*CB3*	D0	*CB2*	*CB1*	No Error	0000

x645_01_022103

图 1: **校正子表**

如无位元错误，如图 2 所示，校验位会与计算出的相应数据的校验位相符。此时，所有校正子位均为零，指向无错误位置。

111	110	101	100	011	010	001	**000**	
D63	D62	D61	D60	D59	D58	D57	*CB7*	1000
D56	D55	D54	D53	D52	D51	D50	D49	0111
D48	D47	D46	D45	D44	D43	D42	D41	0110
D40	D39	D38	D37	D36	D35	D34	D33	0101
D32	D31	D30	D29	D28	D27	D26	*CB6*	0100
D25	D24	D23	D22	D21	D20	D19	D18	0011
D17	D16	D15	D14	D13	D12	D11	*CB5*	0010
D10	D9	D8	D7	D6	D5	D4	*CB4*	0001
D3	D2	D1	*CB3*	D0	*CB2*	*CB1*	No Error ←	**0000**

x645_01_022003

图 2: **未检测出位元错误**

如果存在单位元错误，如图 3 所示，数个校正子就会为奇校验（产生逻辑 1），可确定相应的列和行。如果 D28 有错，那么 CB1、CB2 和 CB6 就会发生奇偶性错误。这样，D28 就会在校正子表中被认定为错误位。

111	110	101	100	**011**	010	001	000	
D63	D62	D61	D60	D59	D58	D57	*CB7*	1000
D56	D55	D54	D53	D52	D51	D50	D49	0111
D48	D47	D46	D45	D44	D43	D42	D41	0110
D40	D39	D38	D37	D36	D35	D34	D33	0101
D32	D31	D30	D29	**D28** ←	~~D27~~	~~D26~~	~~*CB6*~~	**0100**
D25	D24	D23	D22	D21	D20	D19	D18	0011
D17	D16	D15	D14	D13	D12	D11	*CB5*	0010
D10	D9	D8	D7	D6	D5	D4	*CB4*	0001
D3	D2	D1	*CB3*	D0	*CB2*	*CB1*	No Error	0000

x645_02_022003

图 3: **检测出单位元错误**

如果存在双位元错误，如图 4 所示，错误位的位置不是未被指出，就是未被正确指出。例如，如果一个双位元错误发生在 D28 和 D22 上，产生的校正子会指向 111 列 0111 行。不过，通过添加的那个涵盖每个数据位的校验位 (CB8)，仍有可能检测出双位元错误。如果从 CB1 到 CB7 均产生非零值，而 CB8 返回零，则说明存在双位元错误。

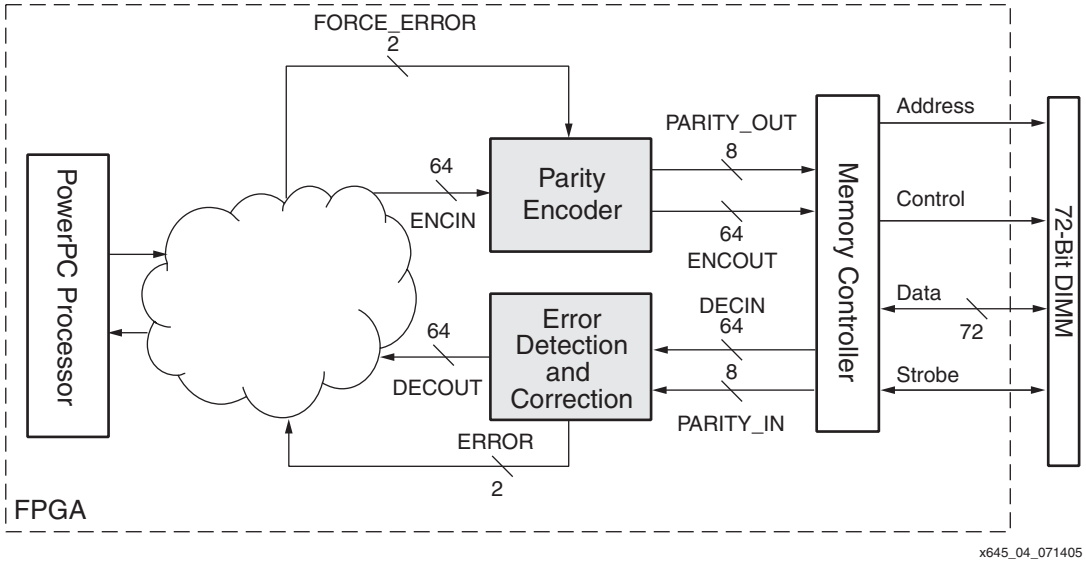
111	110	101	100	011	010	001	000	
D63	D62	D61	D60	D59	D58	D57	*CB7*	1000
~~D56~~	~~D55~~	~~D54~~	~~D53~~	~~D52~~	~~D51~~	~~D50~~	~~D49~~	**0111**
D48	D47	D46	D45	D44	D43	D42	D41	0110
D40	D39	D38	D37	D36	D35	D34	D33	0101
D32	D31	D30	D29	**D28**	D27	D26	*CB6*	0100
D25	D24	D23	**D22**	D21	D20	D19	D18	0011
D17	D16	D15	D14	D13	D12	D11	*CB5*	0010
D10	D9	D8	D7	D6	D5	D4	*CB4*	0001
D3	D2	D1	*CB3*	D0	*CB2*	*CB1*	No Error	0000

x645_03_022003

图 4: **检测出双位元错误**

设计简介

图 5 中的结构图描述了一个包含 ECC 功能的 DDR 存储器控制器的使用情况。此示例中的 DDR DIMM 为 Micron MT18VDDT6472G，一种 ECC 配置模块。此参考设计包含一个奇偶校验编码器及 (parity encoder) 奇偶校验解码器单元。编码器执行生成矩阵的功能，而解码器则负责检错和纠错 (error detection and correction)。另外，该设计还支持诊断功能。以下章节对这些功能进行了详细说明。



x645_04_071405

图 5: **存储器系统中的 ECC**

奇偶校验编码器

编码器由 XOR 和一个在查找表 (LUT) 中实现的位元错误生成器组成。若添加一个可选的流水线级数，可进一步改进性能。图 6 为奇偶校验编码器的结构图。

校验位与关联的 64 位数据一起被写入存储器。存储器读取数据时，会同时读取该数据及校验位。任何在 FPGA 和存储器之间进行读写的过程中产生的错误都会被检测出来。

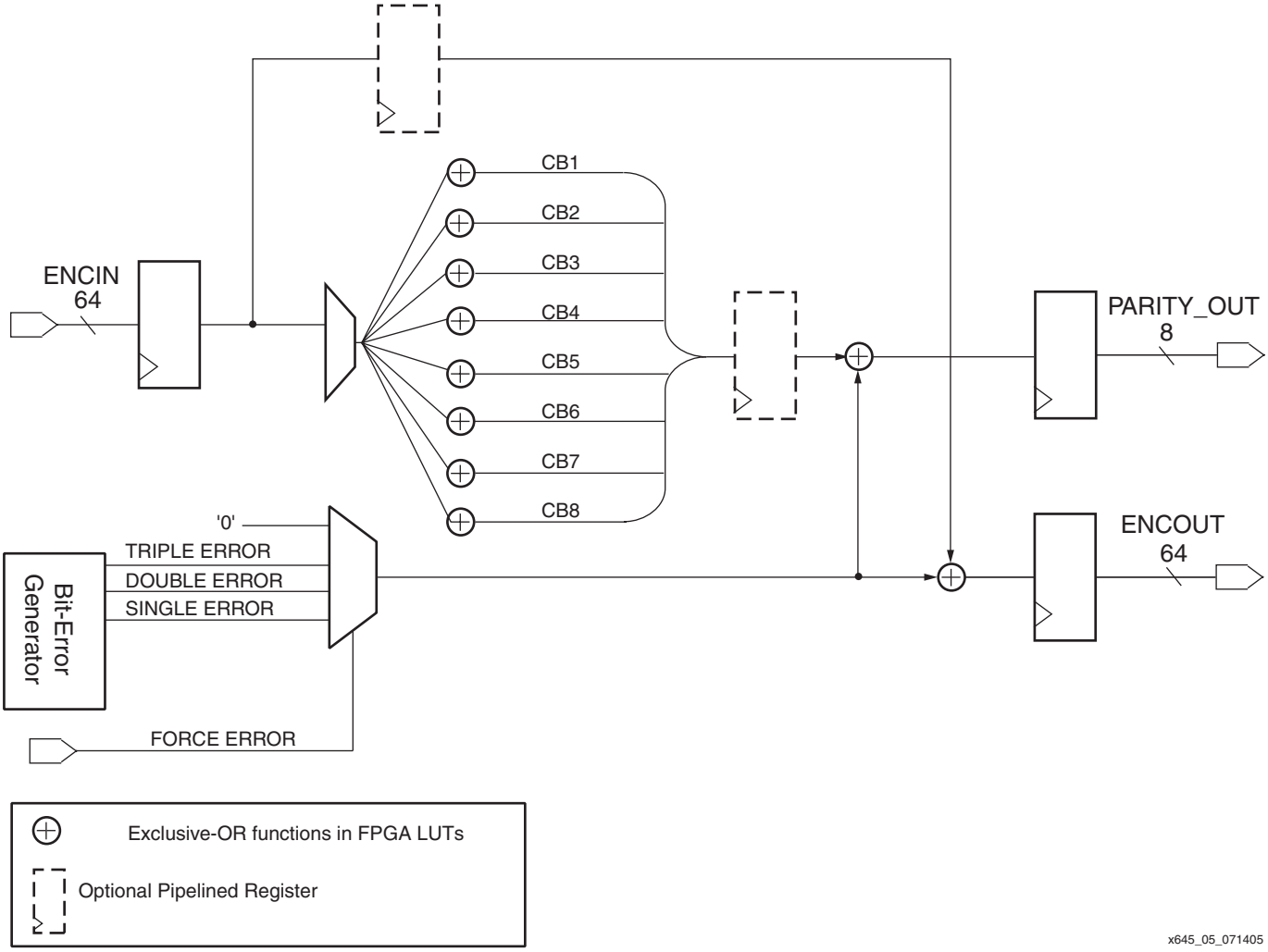


图 6: **奇偶校验编码器结构图**

奇偶校验位根据未经修改的汉明码生成。表 1 表明生成 (72,64) 代码字的参与位。表 2 表明生成 (39,32) 代码字的参与位。

表 1: **64 位汉明码**

参加校验的数据位	生成的校验位							
	CB1	CB2	CB3	CB4	CB5	CB6	CB7	CB8
0	√	√						√
1	√		√					√
2		√	√					√
3	√	√	√					√
4	√			√				√
5		√		√				√
6	√	√		√				√
7			√	√				√
8	√		√	√				√
9		√	√	√				√
10	√	√	√	√				√
11	√				√			√
12		√			√			√
13	√	√			√			√
14			√		√			√
15	√		√		√			√
16		√	√		√			√
17	√	√	√		√			√
18				√	√			√
19	√			√	√			√
20		√		√	√			√
21	√	√		√	√			√
22			√	√	√			√
23	√		√	√	√			√
24		√	√	√	√			√
25	√	√	√	√	√			√
26	√					√		√
27		√				√		√
28	√	√				√		√
29			√			√		√
30	√		√			√		√
31		√	√			√		√
32	√	√	√			√		√
33				√		√		√
34	√			√		√		√

表 1: 64 位汉明码 *（续表）*

参加校验的数据位	生成的校验位							
	CB1	CB2	CB3	CB4	CB5	CB6	CB7	CB8
35		√		√		√		√
36	√	√		√		√		√
37			√	√		√		√
38	√		√	√		√		√
39		√	√	√		√		√
40	√	√	√	√		√		√
41					√	√		√
42	√				√	√		√
43		√			√	√		√
44	√	√			√	√		√
45			√		√	√		√
46	√		√		√	√		√
47		√	√		√	√		√
48	√	√	√		√	√		√
49				√	√	√		√
50	√			√	√	√		√
51		√		√	√	√		√
52	√	√		√	√	√		√
53			√	√	√	√		√
54	√		√	√	√	√		√
55		√	√	√	√	√		√
56	√	√	√	√	√	√		√
57	√						√	√
58		√					√	√
59	√	√					√	√
60			√				√	√
61	√		√				√	√
62		√	√				√	√
63	√	√	√				√	√

表 2: **32 位汉明码**

参加校验的数据位	生成的校验位						
	CB0	CB1	CB2	CB3	CB4	CB5	CB6
0	√	√					√
1	√		√				√
2		√	√				√
3	√	√	√				√
4	√			√			√
5		√		√			√
6	√	√		√			√
7			√	√			√
8	√		√	√			√
9		√	√	√			√
10	√	√	√	√			√
11	√				√		√
12		√			√		√
13	√	√			√		√
14			√		√		√
15	√		√		√		√
16		√	√		√		√
17	√	√	√		√		√
18				√	√		√
19	√			√	√		√
20		√		√	√		√
21	√	√		√	√		√
22			√	√	√		√
23	√		√	√	√		√
24		√	√	√	√		√
25	√	√	√	√	√		√
26	√					√	√
27		√				√	√
28	√	√				√	√
29			√			√	√
30	√		√			√	√
31		√	√			√	√

奇偶校验解码器

图 7 所示的解码器单元由三个模块组成：

- 校正子 (Syndrome) 生成
- 校正子 LUT 和掩码 (Mask) 生成
- 数据校正 (Data correction)

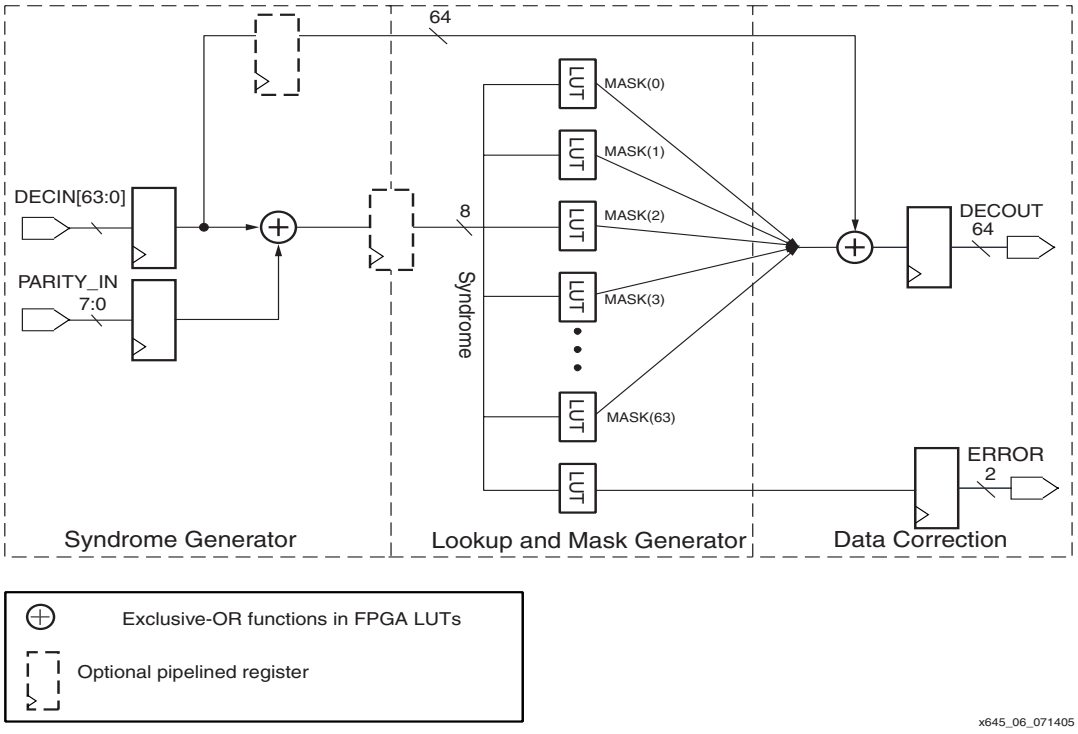


图 7: **ECC 功能结构图**

校正子生成

输入的 64 位数据与 8 位奇偶校验位一起经过 XOR 运算，生成 8 位校正子 （S1 至 S8）。这与校验位生成非常类似，例如：

S1 = DECIN0 ⊕ DECIN1 ⊕ DECIN3 ⊕ DECIN4 ⊕ DECIN6 ⊕ DECIN8 ⊕ DECIN10 ⊕ DECIN11 ⊕ DECIN13 ⊕ DECIN15 ⊕ DECIN17 ⊕ DECIN19 ⊕ DECIN21 ⊕ DECIN23 ⊕ DECIN25 ⊕ DECIN26 ⊕ DECIN28 ⊕ DECIN30 ⊕ DECIN32 ⊕ DECIN34 ⊕ DECIN36 ⊕ DECIN38 ⊕ DECIN40 ⊕ DECIN42 ⊕ DECIN44 ⊕ DECIN46 ⊕ DECIN48 ⊕ DECIN50 ⊕ DECIN52 ⊕ DECIN54 ⊕ DECIN56 ⊕ DECIN57 ⊕ DECIN59 ⊕ DECIN61 ⊕ DECIN63 ⊕ PARITY_IN(1)

随后，下一阶段使用此校正子查找错误类型和错误位置。若在此处添加一个可选的流水线级数，可进一步改进性能。

校正子 LUT 和掩码生成

为校正单位元错误，系统会创建 64 位校正掩码。此掩码的每一位都是基于上一阶段的校正子结果生成的。如果未检测到错误，此掩码的所有位都会变为零。如果检测到单位元错误，相应的掩码会屏蔽除错误位之外的所有位。下一阶段，使用原始数据对此掩码进行 XOR 运算。最终，错误位被反转 （或校正）至正确状态。如果检测到双位元错误，所有掩码位都会变为零。在同一时钟周期内，系统会创建错误类型和相应的校正掩码。

数据校正

在数据校正阶段，如果需要，掩码会与原始输入数据一起进行 XOR 运算，以将错误位改为正确状态。不再有单位元错误或双位元错误后，所有掩码位将为零。最终，输入数据通过 ECC 单元，而原始数据不变。

错误诊断

除了显示错误类型外，本参考设计还支持诊断模式。单位元、多位元和三位元错误可能发生在输出代码字中。

ERROR 端口为 00 表明未检测到单位元、双位元或多位元错误。换言之，受检的数据没有奇偶位错误。ERROR 端口为 01 表明 72 位代码字中出现了单位元错误，并且此错误已校正，数据不再包含错误。ERROR 端口为 10 表明代码字中出现双位元错误。这种情况下不可能进行纠错。如果错误端口为 11，代码字中则可能出现了超出检测能力范围的错误，且无法进行纠错。这是无效的错误类型。

在编码器的输出端，位元错误可能会被故意加入代码字中，以对系统进行测试。Force_error 提供数种错误模式。

Force_error = 00

这是正常操作模式。编码器输出中未加入位元错误。

Force_error = 01

单位元错误模式。在每个时钟上升沿生成的代码字中，一个位元反转（0 变成 1 或 1 变成 0）。单位元错误按顺序从代码字的 0 位移至 72 位。只要单位元错误模式处于活动状态，该顺序就会重复出现。

Force_error = 10

有界双位元错误模式。在每个时钟上升沿生成的代码字中，两个连续位元反转（0 变成 1 或 1 变成 0）。双位元错误按顺序从代码字的 (0,1) 位移至 (71, 72) 位。只要双位元错误模式处于活动状态，该顺序就会重复出现。

Force_error = 11

有界三位元错误模式。在每个时钟上升沿生成的代码字中，三个位元反转（0 变成 1 或 1 变成 0）。三位元错误按顺序从代码字的 (0, 1, 2) 位一起移至 (70, 71, 72) 位。只要三位元错误模式处于活动状态，该顺序就会重复出现。

资源利用和性能

本参考设计使用最少的资源，而具有出众的性能。表 3 提供了性能和资源利用的总结。此设计使用 Xilinx 综合工具 (XST) 综合而成。其性能总结是基于 ISE 8.2i 速度指标，只反映 64 位版 ECC 参考设计。

表 3: **性能和资源利用总结**

器件	利用率 (1)	性能	
		非流水线	1 级流水线
XC2VP4 –6	16%	144 MHz	313 MHz
XC2VP7 –6	10%	136 MHz	298 MHz
XC2VP20 –6 或 XC2VPX20 –6	5%	132 MHz	232 MHz
XC2VP50 –6	2%	127 MHz	176 MHz
XC4VLX15 –11	9%	197 MHz	295 MHz
XC4VFX20 –11	7%	204 MHz	256 MHz
XC4VFX60 –11	3%	158 MHz	253 MHz
XC4VSX35 –11	4%	172 MHz	293 MHz
XC5VLX30 –2	4%	251 MHz	302 MHz
XC5VLX110 –2	1%	239 MHz	301 MHz

延迟

尽管没有要求，但模块的 I/O 已被寄存。对于编码器，从输入数据出现在 ENCIN 中到编码后的数据可用于 ENCOUT 的延迟时间为两个时钟周期（非流水线）或三个时钟周期（流水线）。

对于解码器，从输入的数据出现在 DECIN 中到经过处理的数据可用于 DECOUT 的延迟时间为两个时钟周期（非流水线）或三个时钟周期（流水线）。状态信号 ERROR 与 DECOUT 同步。图 8 说明了时序延迟。

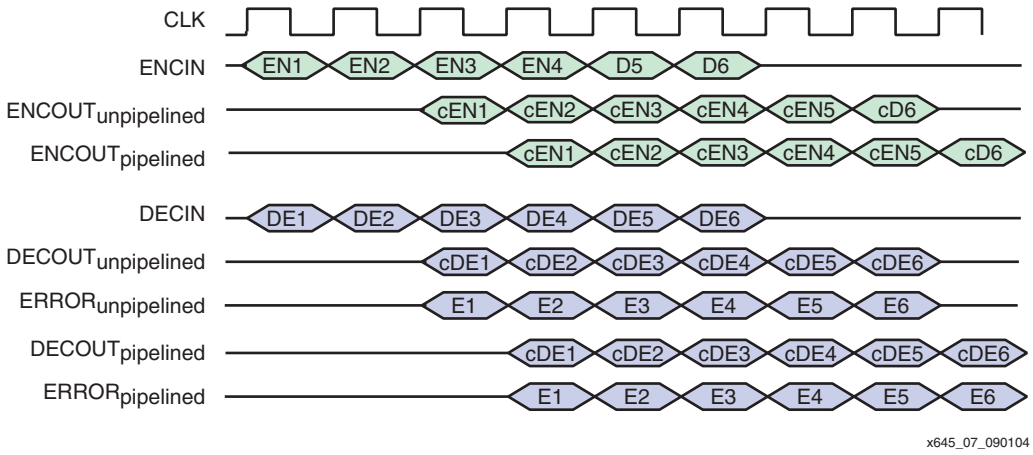


图 8: **时序图**

延迟说明

ENx = 编码前的写入数据。

cENx = 编码后的写入数据。校验位可用于写入操作。

DEx = 通过 ECC 单元前的读取数据。

cDE = 通过 ECC 单元后的被纠正的读取数据。

Ex = ECC 单元生成的错误状态。

引脚说明

表 4 列出 ECC 模块上升时钟边沿的所有用户界面引脚。

表 4: **ECC 模块引脚说明**

引脚名称	输入 / 输出	宽度（64 位）	宽度（32 位）	说明
CLK	输入			时钟输入。
RESET	输入			低电平有效复位。
ENCIN	输入	63:0	31:0	输入到编码器的原始数据。
ENCOUT	输出	63:0	31:0	通过编码器的被寄存原始数据。
PARITY_OUT	输出	7:0	6:0	基于在相同时钟边沿被寄存的数据 (encin)，从编码器生成的奇偶校验位。
DECIN	输入	63:0	31:0	输入解码器的数据。
DECOUT	输出	63:0	31:0	通过 DECIN 纠正的数据。

表 4: **ECC 模块引脚说明 *(续表)***

引脚名称	输入 / 输出	宽度 (64 位)	宽度 (32 位)	说明
PARITY_IN	输入	7:0	6:0	与在相同上升时钟边沿被寄存的输入数据 (DECIN) 相关的奇偶校验位。
FORCE_ERROR	输入	1:0	1:0	在编码的数据字中为测试而引入位元错误。 00 – 正常操作 01 – 加入单位元错误 10 – 加入双位元错误 11 – 加入三位元错误
ERROR	输出	1:0	1:0	错误状态 00 – 无错误 01 – 检测到单位元错误并已纠正 10 – 检测到双位元错误，未纠正 11 – 检测到无效位元错误

参考设计文件

VHDL 和 Verilog 参考设计文件发布在 Xilinx 网站上，网址：

http://www.xilinx.com/cn/bvdocs/appnotes/xapp645.zip

结论

本应用指南说明了在 Virtex–II、Virtex–II Pro 和 Virtex–4 器件中编写和查找汉明码的一种简单方法。

修订历史

下表说明此技术文档的修订历史。

日期	版本	修订
2003 年 3 月 3 日	1.0	Xilinx 最初版本。
2003 年 9 月 17 日	1.1	更新关于 32 位数据的检错和纠错 (EDC) 功能。目前的性能反映了 1.81 版速度文件。
2004 年 2 月 3 日	1.2	扩展文件以包括流水线应用程序。
2004 年 9 月 1 日	2.0	更新以包括 Virtex–4 FPGA。
2006 年 7 月 20 日	2.1	更新性能和资源利用总结表 (表 3)。
2006 年 8 月 9 日	2.2	更新**资源利用和性能**部分，以及性能和资源利用总结表 (表 3)。