(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 1750411 B
(45)授权公告日 2012.02.22

(21)申请号 200510069363.1

(22)申请日 2005.05.13

(30)优先权数据
10/939,880 2004.09.13 US

(73)专利权人 安华高科技杰纳勒尔 IP（新加坡）私人有限公司
地址 新加坡新加坡市

(72)发明人 维森特·V·卡万纳
杰弗里·R·墨菲 迪伦·杰克逊

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
代理人 王怡

(51)Int.Cl.
H03M 13/01(2006.01)
H03M 13/09(2006.01)
H03M 13/00(2006.01)

(56)对比文件
US 20020069232 A1,2002.06.06, 全文.
CN 1523830 A,2004.08.25, 全文.
US 20040059984 A1,2004.03.25, 全文.
CN 1319801 A,2001.10.31, 全文.
US 5619516 A,1997.04.08, 全文.

审查员 赵小宁

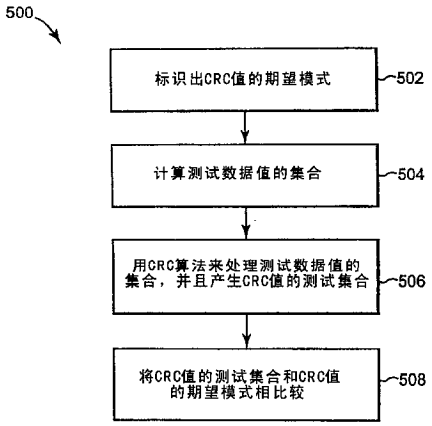
权利要求书 2 页 说明书 7 页 附图 3 页

(54)发明名称

用于测试循环冗余码算法的方法和系统

(57)摘要

一种产生用于测试CRC算法的测试数据模式的方法，该CRC算法被配置成基于生成多项式产生CRC值，该方法包括标识出中间CRC值的期望模式。该方法包括基于中间CRC值的期望模式和生成多项式产生测试数据模式，其中测试数据模式被配置成引发CRC算法来产生中间CRC值的期望模式。



CN 1750411 B

1. 一种测试循环冗余码算法的方法，所述循环冗余码算法被配置成处理数据消息并且基于生成多项式产生对应的中间循环冗余码值，所述方法包括：

产生中间循环冗余码值的期望集合；

基于中间循环冗余码值的所述期望集合和所述生成多项式产生测试数据值的集合；

利用所述循环冗余码算法处理测试数据值的所述集合，由此产生中间循环冗余码值的测试集合；以及

将中间循环冗余码值的所述测试集合和中间循环冗余码值的所述期望集合相比较。

2. 如权利要求 1 所述的方法，其中中间循环冗余码值的所述期望集合是递增模式和递减模式之一，其中所述中间循环冗余码值的连续值相差固定值。

3. 如权利要求 2 所述的方法，其中所述固定值和所述循环冗余码算法每个周期所处理的字节数目相同。

4. 如权利要求 1 所述的方法，其中所述期望集合中的所述中间循环冗余码值具有相同值。

5. 如权利要求 1 所述的方法，其中所述期望集合中的所述中间循环冗余码值具有移位模式，其中所述期望集合中的所述中间循环冗余码值的连续值中的位移动至少一位的位置。

6. 如权利要求 1 所述的方法，其中所述期望集合中的所述中间循环冗余码值在两个值之间切换。

7. 如权利要求 1 所述的方法，其中产生测试数据值的集合的步骤包括：

(i) 计算代表所述生成多项式的矩阵的逆矩阵；以及

(ii) 将所述矩阵的逆矩阵乘以第一向量和第二向量的异或结果，所述第一向量代表所述中间循环冗余码值的当前值，所述第二向量代表所述中间循环冗余码值的下一后继值，其中所述相乘的结果提供所述测试数据值中的一个测试数据值。

8. 如权利要求 7 所述的方法，还包括：

对于所述测试数据值的集合中的每个测试数据值来重复步骤 (ii)。

9. 如权利要求 1 所述的方法，其中每个测试数据值对应于所述中间循环冗余码值之一，并且其中每个测试数据值是基于所述中间循环冗余码值的当前值以及所述中间循环冗余码值的下一后继值而被产生的。

10. 如权利要求 1 所述的方法，其中所述循环冗余码算法是以循环冗余码硬件实现的，并且其中产生期望集合并且比较所述测试集合的步骤是利用所述循环冗余码硬件中的测试硬件执行的。

11. 一种测试循环冗余码算法的系统，所述循环冗余码算法被配置成处理数据消息并且基于生成多项式产生对应的循环冗余码值，所述系统包括：

用于产生中间循环冗余码值的期望集合的装置；

用于基于中间循环冗余码值的所述期望集合和所述生成多项式产生测试数据值的集合的装置；

用于利用所述循环冗余码算法处理测试数据值的所述集合，由此产生中间循环冗余码值的测试集合的装置；以及

用于将中间循环冗余码值的所述测试集合和中间循环冗余码值的所述期望集合相比

较的装置。

12. 如权利要求 11 所述的系统，其中中间循环冗余码值的所述期望集合是递增模式和递减模式之一，其中所述中间循环冗余码值的连续值相差固定值。

13. 如权利要求 12 所述的系统，其中所述固定值和所述循环冗余码算法每个周期所处理的字节数目相同。

14. 如权利要求 11 所述的系统，其中所述期望集合中的所述中间循环冗余码值具有相同值。

15. 如权利要求 11 所述的系统，其中所述期望集合中的所述中间循环冗余码值具有移位模式，其中所述期望集合中的所述中间循环冗余码值的连续值中的位移动至少一位的位置。

16. 如权利要求 11 所述的系统，其中所述期望集合中的所述中间循环冗余码值在两个值之间切换。

17. 如权利要求 11 所述的系统，其中用于产生测试数据值的集合的装置执行以下步骤：

(i) 计算代表所述生成多项式的矩阵的逆矩阵；以及

(ii) 将所述矩阵的逆矩阵乘以第一向量和第二向量的异或结果，所述第一向量代表所述中间循环冗余码值的当前值，所述第二向量代表所述中间循环冗余码值的下一后继值，其中所述相乘的结果提供所述测试数据值中的一个测试数据值。

18. 如权利要求 17 所述的系统，其中用于产生测试数据值的集合的装置还执行以下步骤：

对于所述测试数据值的集合中的每个测试数据值来重复步骤 (ii)。

19. 如权利要求 11 所述的系统，其中每个测试数据值对应于所述中间循环冗余码值之一，并且其中每个测试数据值是基于所述中间循环冗余码值的当前值以及所述中间循环冗余码值的下一后继值而被产生的。

20. 如权利要求 11 所述的系统，其中所述循环冗余码算法是以循环冗余码硬件实现的，并且其中产生期望集合并且比较所述测试集合的步骤是利用所述循环冗余码硬件中的测试硬件执行的。

用于测试循环冗余码算法的方法和系统

技术领域

[0001] 本发明一般涉及循环冗余码 (CRC) 错误检测技术，更具体而言，涉及用于测试 CRC 算法的测试数据模式。

背景技术

[0002] 错误检测技术（例如基于循环冗余码 (CRC) 的技术）的目的是，使得通过噪声信道传送的消息的接收器能够确定消息是否已损坏。为实现此目的，发送器产生作为消息的函数的值（称为帧校验序列或 FCS），并且通常将 FCS 附加于消息上。然后接收器可以使用用来产生所接收消息的 FCS 的相同函数来察看该消息是否被正确接收。

[0003] 利用 CRC 算法，消息位被视为 n 位多项式的二进制系数。通常将消息多项式乘以 x^m，其中 m 是 CRC 多项式（即，“生成多项式”）的幂次。相乘的结果除以 CRC 多项式。大多数实现采用同时执行乘以 x^m 和除以 CRC 多项式的方法，而不是顺序进行这些操作。这些操作的结果是 FCS，FCS 通常被求补并附加到消息上。在某些情况下，FCS 不被求补，并且偶尔 FCS 被置于别的位置，例如被置于头部字段中。

[0004] 接收器将具有附加 FCS 的所接收消息除以 CRC 多项式。假定 FCS 在被附加到消息上之前已被求补，并且在传送过程中未发生错误，则在接收器处相除的结果将是固定值，该值等于 2m 幂次多项式（对于幂次较高的 m 项，系数为 1，对于幂次较低的 m 项，系数为 0）除以 CRC 多项式的结果。该固定值有时被称为“魔数 (magic number)”，并且取决于多项式。如果相除的结果不等于魔数，则这指示发生了错误。

[0005] 规定了大多数 CRC 算法的方法涉及描述 CRC 多项式、CRC 计算的初始和终了状态、输入（数据流）在处理之前是否被求补、以及结果 (CRC) 在被附加到由 CRC 保护的数据流之前是否被求补。此外，算法可以指示字节的交换以适应 Little Endian（低字节优先）和 Big Endian（高字节优先）处理器。通常提供样本测试模式 (pattern) 以验证 CRC 算法实现是否正确。

[0006] 软件或硬件 CRC 算法的设计者通常面临下述问题，即，当 CRC 算法产生不正确的结果时，用以确定算法在过程的哪个步骤出现故障的信息很少。样本测试模式通常帮助不大，这是因为该模式仅告诉算法实现者当整个过程完成时预期的结果，并且该模式没有提供有关 CRC 计算的中间状态的任何信息。

发明内容

[0007] 本发明的一种形式提供了一种产生用于测试 CRC 算法的测试数据模式的方法。该 CRC 算法被配置成基于生成多项式产生 CRC 值。该方法包括标识出中间 CRC 值的期望模式。该方法包括基于中间 CRC 值的期望模式和生成多项式产生测试数据模式，其中测试数据模式被配置成使 CRC 算法产生中间 CRC 值的期望模式。

[0008] 本发明的另一种形式提供了一种测试循环冗余码算法的方法，所述循环冗余码算法被配置成处理数据消息并且基于生成多项式产生对应的中间循环冗余码值，所述方法包

括：产生中间循环冗余码值的期望集合；基于中间循环冗余码值的所述期望集合和所述生成多项式产生测试数据值的集合；利用所述循环冗余码算法处理测试数据值的所述集合，由此产生中间循环冗余码值的测试集合；以及将中间循环冗余码值的所述测试集合和中间循环冗余码值的所述期望集合相比较。

[0009] 本发明的另一种形式提供了一种测试循环冗余码算法的系统，所述循环冗余码算法被配置成处理数据消息并且基于生成多项式产生对应的循环冗余码值，所述系统包括：用于产生中间循环冗余码值的期望集合的装置；用于基于中间循环冗余码值的所述期望集合和所述生成多项式产生测试数据值的集合的装置；用于利用所述循环冗余码算法处理测试数据值的所述集合，由此产生中间循环冗余码值的测试集合的装置；以及用于将中间循环冗余码值的所述测试集合和中间循环冗余码值的所述期望集合相比较的装置。

附图说明

[0010] 图 1 是被分割成子消息的消息以及对应于子消息之一的复合子消息的图形表示。

[0011] 图 2 是图示了根据本发明一个实施例用于产生测试数据模式和用于测试 CRC 算法的系统的框图。

[0012] 图 3 是图示了根据本发明一个实施例的图 2 中所示的 CRC 算法操作的框图。

[0013] 图 4 是图示了根据本发明一个实施例的图 2 中所示的测试数据生成器操作的框图。

[0014] 图 5 是图示了根据本发明一个实施例用于测试 CRC 算法的方法的流程图。

具体实施方式

[0015] 在下面的详细说明中，参考了附图，附图构成了说明的一部分，并且通过图示示出了可以实施本发明的具体实施例。还应该理解，可以利用其它实施例，并且可以进行结构或者逻辑改变而不背离本发明的范围。因而，下面的详细描述不应理解为限制的意义，并且本发明的范围由权利要求定义。

[0016] 图 1 图示了多位二进制数据消息 10。在诸如某些因特网协议的一些通信协议中，诸如数据消息 10 的数据消息可以被分组或者分割成子消息进行传送。例如，iSCSI 数据消息可以用其 CRC FCS 来保护并且经由多个 IP 分组（可表示为子消息）来传送。消息 10 可以任意分割成表示为子消息 12、14 和 16 的位块。消息 10 可以表达为子消息的模 2(MOD-2) 加和 (summation)。对于此加和，用零代替来自其它子消息的数据而修改每个子消息。例如，考虑子消息 14。为了用于 MOD-2 加和以形成消息 10，子消息 14 被建模为复合子消息 18。如图可见，复合子消息 18 具有用于代替来自子消息 12 和 16 的子消息数据的多个零 20。当照这样建模时，每个复合子消息将具有和初始消息 10 相同的长度（位数）。

[0017] 因为 CRC 运算是线性转换，所以消息 10 的 CRC 是复合子消息的 CRC 之和。如本文所采用的，子消息或者复合子消息的 CRC 将表示为递增 CRC 或者中间 CRC 值。

[0018] 如本领域普通技术人员将理解的那样，CRC 运算是利用无进位的二进制算术 MOD-2 进行的。利用这样的算术，加法和减法都等同于逻辑异或 (XOR) 操作，乘法等同于逻辑 AND 操作。从而，符号“+”和“-”指示逻辑 XOR。

[0019] 图 2 的框图图示了根据本发明一个实施例用于产生测试数据模式和用于测试 CRC

算法的系统 200。在本发明的一种形式中，系统 200 是计算机或类似设备。系统 200 包含处理器 202、输入 / 输出 (I/O) 接口 226 和存储器 208，所有这些部件都经由通信链路 228 可通信地耦合到一起。系统 200 经由 I/O 接口 226 传送和接收经 CRC 保护的数据消息。

[0020] 存储器 208 可以既包含易失性存储器也包含非易失性存储器组件。易失性组件是一旦掉电就不保持数据值的组件。非易失性组件是掉电时保持数据的组件。这样，存储器 208 可以包含例如随机存取存储器 (RAM)、只读存储器 (ROM)、硬盘驱动、经由关联的软盘驱动器存取的软盘、经由光盘驱动器存取的光盘、经由适当的磁带驱动器存取的磁带、和 / 或其它存储器组件、或者这些存储器组件任意两个或者多个的组合。此外，RAM 可以包含例如静态随机存取存储器 (SRAM)、动态随机存取存储器 (DRAM) 或者磁随机存取存储器 (MRAM) 以及其它这样的设备。ROM 可以包括例如可编程只读存储器 (PROM)、可擦可编程只读存储器 (EPROM)、电可擦可编程只读存储器 (EPROM) 或者其它类似的存储器设备。

[0021] CRC 算法 210、测试数据消息 214、测试数据生成器 220 和输入 CRC 模式 222 存储在存储器 208 中。测试数据消息 214 包含多个 8 位数据字节 216，其中每两个连续数据字节 216 的集合构成数据字或者子消息 218。在一个实施例中，测试数据消息 214 包含 1024 个 16 位字 218。处理器 202 包含存储当前 CRC 值 206 的处理器寄存器 (CRC 寄存器)204。在一个实施例中，CRC 寄存器 204 是存储 16 位 CRC 值 206 的 16 位寄存器。处理器 202 执行 CRC 算法 210 以处理诸如测试数据消息 214 的数据消息，并且基于 CRC 算法 210 的 CRC 生成多项式 212 产生对应的 CRC。在一个实施例中，数据消息由 CRC 算法 210 以每次 N 个字节的方式处理，其中 N 是整数，并且每 N 字节周期后更新存储在寄存器 204 中的中间 CRC 值 206。在一个实施例中，N 等于 2。

[0022] 在一个实施例中，测试数据消息 214 是与诸如 CRC 算法 210 的具体 CRC 算法相对应的“魔 (magic)”测试数据模式。在一个实施例中，测试数据消息 214 用于测试 CRC 算法 210，这可以用硬件或者软件来实现。测试数据消息 214 的目的是方便验证算法 210 是否如设计者想要的那样工作。在本发明的一种形式中，测试数据消息 214 允许 CRC 算法 210 的实现者精确地确定过程是在哪里 (例如，哪个时钟周期) 出现故障的，这使得调试算法 210 的过程更容易。在本发明的一种形式中，处理器 202 执行测试数据生成器 220，以基于期望的输入 CRC 模式 222 和 CRC 生成多项式 212 产生测试数据消息 214。在一个实施例中，输入 CRC 模式 222 包含在常规模式下的多个中间 CRC 值 206。

[0023] 在本发明的一种形式中，当测试数据消息 214 被 CRC 算法 210 处理时，测试数据消息 214 引发下述中间 CRC 值的产生，对于被处理的数据消息 214 的每个字节，该中间 CRC 值增加 1(在加载初始值并且数据流 214 的处理开始之后)。在此实施例中，中间 CRC 值可以被视为已由 CRC 算法 210 处理的数据字节数目的计数。在另一实施例中，测试数据消息 214 引发下述中间 CRC 值的产生，对于被 CRC 算法 210 处理的数据消息 214 的每个字节，该中间 CRC 值减少 1。在另一实施例中，测试数据消息 214 引发下述中间 CRC 值的产生，对于被 CRC 算法 210 处理的数据消息 214 的每个字节，该中间 CRC 值保持恒定不变。

[0024] 在另一实施例中，测试数据消息 214 引发下述中间 CRC 值的产生，对于被 CRC 算法 210 处理的数据消息 214 的每个字节，中间 CRC 值移动一个位 (或者多个位)。例如，在一个实施例中，测试数据消息 214 引发产生第一中间 CRC 值“00000001”、第二中间 CRC 值“00000010”、第三中间 CRC 值“00000100”等等。由此示例可见，每次数据消息 214 的新字

节被 CRC 算法 210 处理时，中间 CRC 值的“1”就向左移动一位。中间 CRC 值的这种模式被称为“步行 (walking)1”模式或者移位模式。

[0025] 在另一实施例中，测试数据消息 214 引发下述中间 CRC 值的产生，对于被 CRC 算法 210 处理的数据消息 214 的每个字节，中间 CRC 值在两个值之间切换。例如，在一个实施例中，测试数据消息 214 引发产生第一中间 CRC 值“0xAAAA”、第二中间 CRC 值“0x5555”、第三中间 CRC 值“0xAAAA”等。由此示例可见，对于被 CRC 算法 210 处理的数据消息 214 的每个字节，中间 CRC 值在“0xAAAA”和“0x5555”之间切换。中间 CRC 值的这种模式被称为切换值模式 (toggling value pattern)。

[0026] 在本发明的另一种形式中，CRC 算法 210 以每次两个字节的方式处理数据消息，并且测试数据消息 214 引发下述中间 CRC 值的产生，对于被 CRC 算法 210 处理的数据消息 214 的每两个 8 位字节 216，中间 CRC 值增加 (或者减少)2。在一个实施例中，测试数据生成器 220 被配置成产生将生成中间 CRC 值的任何期望模式的测试数据消息 214。下面参考图 3 和图 4 进一步详细描述 CRC 算法 210 和测试数据生成器 220。

[0027] 本领域普通技术人员应该了解，本发明的组件 (包括系统 200 的组件) 可以驻留于一个或多个计算机可读介质上的软件中。本文使用的术语“计算机可读介质”被定义为包含任何易失性或非易失性存储器类型，例如软盘、硬盘、CD-ROM、闪存、只读存储器 (ROM) 以及随机存取存储器。

[0028] 图 3 的框图图示了根据本发明一个实施例的图 2 中所示的 CRC 算法 210 的操作。如图 3 所示，CRC 算法 210 接收当前中间 CRC 值 (crc[n])302 和当前数据值 (data[n])304，并且基于值 302 和 304，以及 CRC 生成多项式 212(图 2) 产生下一中间 CRC 值 (crc[n+1])306，其中“n”是整数，其被用作标识 CRC 值和数据值的序号。在一个实施例中，CRC 值 302 和 306 是 16 位中间 CRC 值，这些中间 CRC 值在由 CRC 算法 210 运算出之后存储在 CRC 寄存器 204(图 2) 中。在一个实施例中，当前数据值 304 是来自诸如测试数据消息 214(图 2) 的正被处理的数据消息的 16 位数据值。在进行计算之后，由 CRC 算法 210 产生的下一 CRC 值 306 被存储在 CRC 寄存器 204 中，并且对于 CRC 运算过程的下一迭代或者周期 (cycle) 而言其成为当前 CRC 值 302。这样，在 CRC 运算过程的每次迭代期间，用新 (更新) 的 CRC 值 306 更新 CRC 寄存器 204 中的 CRC 值。

[0029] 由 CRC 算法 210 生成的中间 CRC 值 306 的模式将基于 CRC 算法 210 的 CRC 多项式 212 以及输入数据值 304 的模式而变化。用于输入数据值 304 的给定集合的 CRC 值 306 倾向于具有随机的表现形式。即使向 CRC 算法 210 提供输入数据值 304 的规则模式，对应的 CRC 值 306 也将具有随机的表现形式。但是，在本发明的一种形式中，系统 200(图 2) 被配置成产生下述测试数据消息 214，所述测试数据消息 214 在一个实施例中是表面上看来随机的输入数据流，在输入数据流 214 中的每个数据项 304 被 CRC 算法 210 处理时，该随机输入数据流产生 CRC 值 306 的可预测的规则序列。

[0030] 图 4 的框图图示了根据本发明一个实施例的图 2 中所示的测试数据生成器 220 的操作。测试数据生成器 220 包含期望的 CRC 模式生成器 404 和魔数生成器 410。CRC 模式生成器 404 接收当前中间 CRC 值 (crc[n])402，并且产生下一中间 CRC 值 (crc[n+1])406，其中 n 是整数，其用作标识 CRC 值和数据值的序号。在一个实施例中，CRC 模式生成器 404 产生下述中间 CRC 值 406，所述中间 CRC 值 406 在数据产生过程的每个迭代期间增加值 2。

在其它实施例中，CRC 模式生成器 404 被配置成产生中间 CRC 值 406 的其它模式。CRC 模式生成器 404 产生的中间 CRC 值模式被称作输入 CRC 模式 222（图 2）。在一个实施例中，CRC 值 402 和 406 是 16 位中间 CRC 值。

[0031] 数据生成器 410 接收下一 CRC 值 406 和当前 CRC 值 402，并且基于值 402 和 406，以及 CRC 生成多项式 212 产生当前数据值 (data[n])412。在一个实施例中，当前数据值 412 是两字节（例如，16 位）数据值。对于数据产生过程的每个后继迭代，由 CRC 模式生成器 404 运算出的下一 CRC 值 406 变为当前 CRC 值 402。由数据生成器 410 所产生的数据值 412 的集合被称为测试数据消息 214（图 2）。

[0032] 在一个实施例中，数据生成器 410 本质上是逆向 CRC 引擎 (inverseCRC engine)，其基于 CRC 生成多项式 212 和输入 CRC 模式 222 产生数据值 412，如将进一步详细描述的那样。对于在每次迭代期间 CRC 值 406 增加 2 的情况，数据生成器 410 在每次迭代期间运算 2 字节数据值 412，其在被 CRC 算法 210 处理时得到和先前 CRC 值 402 增加 2 同样的新 CRC 值 406。在一个实施例中，为正被运算的数据值多项式 412 的未知系数分配变量名，并且在此数据值多项式 412 上运行 CRC 计算。此计算的结果是其系数是以分配的变量名表达的 16 位多项式。让每个这样的系数和期望多项式的对应系数相等，就得到了具有 16 个未知数的 16 模方程。在一个实施例中，数据生成器 410 包括传统数学计算软件以求解这 16 模方程从而在每个迭代期间确定适当数据 412。

[0033] 在另一实施例中，数据生成器 410 被配置成基于线性矩阵方程产生数据值 412。CRC 计算是可由线性矩阵方程表示的线性操作，该线性矩阵方程将 CRC 406 的下一值描述为中间 CRC 402 的当前值和输入数据 412 的当前值的函数。由于 CRC 值 222 的期望级数 (progression) 或模式已知，所以可求解线性矩阵方程来得到输入数据 412。线性矩阵方程被称为“下一状态方程”，因为线性矩阵方程将 CRC 406 的下一状态显示为 CRC 402 和输入数据 412 的当前状态的函数。在接下来对下一状态方程的描述中将假定中间 CRC 值 402 和 406 是 16 位，并且以一次处理 16 位的方式处理数据消息（例如，测试数据消息 214），其中数据消息的每个 16 位段用数据值 412 表示。在每个周期处理 16 数据位的情况下，16 位 CRC 的下一状态方程如下面方程 I 中所示：

[0034] 方程 I

[0035] $c(n+1) = A\ c(n) + B\ d(n)$

[0036] 其中：

[0037] c(n+1) ＝下一状态 CRC 向量（具有 16 个元素的列向量），代表中间 CRC 值 406；

[0038] c(n) ＝当前状态 CRC 向量（具有 16 个元素的列向量），代表中间 CRC 值 402；

[0039] d(n) ＝当前状态数据向量（具有 16 个元素的列向量），代表数据值 412；

[0040] A ＝ 16×16 的常数矩阵，元素为 1 和 0，这取决于所使用的 CRC 生成多项式；以及

[0041] B ＝ 16×16 的常数矩阵，元素为 1 和 0，这取决于所使用的 CRC 生成多项式。

[0042] 方程 I 中的运算是基于 MOD-2 数学的。矩阵（即，矩阵 A 和 B）中和向量（即，向量 c 和 d）中的元素为 0 或 1。如方程 I 所指示的，以当前状态 CRC 向量 c(n) 来乘常数矩阵 A，并且以当前状态数据向量 d(n) 来乘常数矩阵 B。这两个矩阵乘法的每个都得到新向量。进行这两个新向量的 XOR 操作，此 XOR 操作的结果是下一状态 CRC 向量 c(n+1)。

[0043] 上面方程 I 中给出的下一状态方程可以被处理成各种形式，这取决于每个周期所

处理的数据位数和所采用的CRC宽度。对于每个周期所处理的数据位数等于CRC位数的情况，由于CRC运算的特性，在下一状态方程（方程I）中矩阵A等于矩阵B。从而，对于这种情况，方程I可以写成下面方程II所给出的形式：

[0044] 方程II

[0045] c(n+1) = A(c(n)+d(n))

[0046] 如方程II所示，以矩阵A乘c(n)和d(n)，然后重新排列各项顺序，得到下面的方程III：

[0047] 方程III

[0048] A d(n) = c(n+1)-A c(n)

[0049] 以矩阵A的逆矩阵（即，Inverse[A]）乘方程III的两端，得到下面的方程IV：

[0050] 方程IV

[0051] d(n) = Inverse[A]c(n+1)-c(n)

[0052] 由于方程IV中的算术是MOD-2形式的，其中减法和加法相同并且两个操作都用XOR实现，所以方程IV可以写成下面的方程V给出的形式：

[0053] 方程V

[0054] d(n) = Inverse[A]c(n+1)XOR c(n)

[0055] 方程V示出了如何从当前CRC值402 c(n)和下一CRC值406c(n+1)，计算数据412的当前值d(n)。在一个实施例中，数据生成器410被配置成在数据产生过程的每个迭代期间基于方程V来产生数据值412。

[0056] 应该注意，方程V中给出的矩阵方程基本上是具有16个未知数（即，16个数据位412）的16个线性方程的简化表示法，因而方程组既没有过约束也没有欠约束，从而具有唯一解。对于数据宽度和CRC宽度相同的情况，通过提供当前CRC状态c(n)，以及下一期望CRC状态c(n+1)，可以基于方程V计算得到数据d(n)，d(n)将引发CRC状态改变。如果每个周期处理不止16个数据位d(n)，则可以找到引发同样CRC状态改变的多个数据模式（即，方程组欠约束从而具有多个解）。如果每个周期处理少于16个数据位d(n)，则不能保证将找到引发期望CRC状态改变的数据模式（即，方程组过约束从而可能无解）。

[0057] 图5是图示了根据本发明一个实施例用于测试CRC算法的方法500的流程图。在一个实施例中，系统200(图2)被配置成执行方法500以测试CRC算法210。在502，标识出中间CRC值的期望集合或模式222。在一个实施例中，CRC值的期望模式222由CRC模式生成器404(图4)产生。在504，基于在502标识出的中间CRC值的期望集合222以及用于正被测试的CRC算法210的生成多项式212，由数据生成器410(图4)运算得到测试数据值集合。在504运算得到的测试数据值集合被称为测试数据消息214。测试数据消息214中的每个测试数据值（例如，每16位字218）和期望CRC模式222中的中间CRC值之一相对应。在506，在504计算得到的测试数据值集合由CRC算法210处理以产生中间CRC值的测试集合。

[0058] 在508，将在506产生的中间CRC值的测试集合和在502标识出的中间CRC值的期望模式相比较。508处的比较提供了关于CRC算法210是否正常运行的指示。508处的比较可以用软件或者简单的硬件比较电路来执行。如果CRC算法210正常运行，则在506处产生的CRC值的测试集合中的每个CRC值将和在502处标识出的CRC值的期望集合中对应

的 CRC 值相等。如果 CRC 算法 210 没有正常运行（例如，CRC 值的测试集合中的一个或者多个 CRC 值和 CRC 值的期望集合 222 中的对应 CRC 值不相等），则很容易标识出发生问题的一个周期或者多个周期。在进入到测试数据消息 214 的处理的任何给定数目的时钟周期之后，该时间点处正确的中间 CRC 值就已知了。如果实际的中间 CRC 值和此时间点的期望 CRC 值不匹配，侧这指示发生了错误。

[0059] 根据一个实施例，本发明使实现 CRC 算法的过程更容易，这是因为本发明使人类观察者或者自动化过程可以在每个中间步骤察看 CRC 产生过程的结果，从而确定该过程在哪个步骤发生故障。

[0060] 本发明的实施例还可以用作 CRC 硬件的内置制造测试以确定硬件是否如设计那样运行。例如，CRC 模式生成器 404（图 4）可以作为硬件自测功能的一部分而在硬件中实现。用于产生递增 CRC 模式、递减 CRC 模式、移位 CRC 模式、切换值 CRC 模式或者其它类似模式的硬件非常小且有效率。例如，CRC 模式生成器 404 可以用简单的计数器实现以产生递增 CRC 模式，或者可以用移位寄存器实现以产生移位 CRC 模式。利用简单的比较电路，可以将由 CRC 模式生成器硬件 404 提供的 CRC 模式和由 CRC 硬件基于所接收的测试数据模式 214（图 2）产生的 CRC 值相比较，以确定 CRC 硬件是否正常运行。很容易确定发生故障的确切周期。测试数据模式 214 可以从外部通过由 CRC 硬件使用的普通数据路径被提供给 CRC 硬件，并且可以以传统方式选择测试模式。

[0061] 相反，为了在不使用产生常规 CRC 模式的 CRC 模式生成器 404 的条件下实现这样的内置自测功能，将需要通过单独的数据路径（而不是普通数据路径）将参考 CRC 模式提供给硬件，或者需要将参考 CRC 模式存储在本地存储设备（例如，ROM）中，这使得设计大大复杂化。

[0062] 本发明的实施例还提供了胜过用于规定和测试 CRC 算法的现有技术的其它优点。本发明的一种形式使用规则的 CRC 模式 222（例如，递增、递减、常数或者其它模式），该规则的 CRC 模式 222 是在 CRC 算法 210 操作之前或者同步（in lock step）产生的，从而可以在过程的每个步骤或者周期处直接比较 CRC 值。即，计算测试数据模式 214 的测试数据生成器 220 可以和 CRC 算法 210 并行运行，并且他们的结果在过程的每个步骤处进行比较，而不是仅在过程结束时才进行比较。

[0063] 本发明的一个实施例还可以用于对当前用来规定 CRC 算法的方法进行增强，这是通过有效地为同样的算法提供独立规范实现的。该替代的规范很容易实现，并且可以和实际算法同步地进行仿真，以便在每个时钟周期提供校验。本发明的一种形式使得规定 CRC 算法的过程不那么含糊，这是因为该算法是用两个独立装置有效指定的，并且两个装置的结果在过程的每个阶段都是已知的。

[0064] 此外，当对已被分割成几个子消息的消息的 CRC 进行计算时，有时需要通过下述操作而恢复计算，即重新加载由处理在前的子消息计算得到的上一 CRC 值，随后继续进行当前的子消息。在这种情况下，本发明的一个实施例使得没有必要验证当前 CRC 的值是否等于迄今为止所处理的字节数目。

[0065] 虽然本文已描述和说明了具体实施例，但是本领域普通技术人员将意识到：各种替代和 / 或等价实现可用来替代所示出和描述的具体实施例，而不背离本发明的范围。本申请意图覆盖本文所讨论的具体实施例的任何改动和变化。因而，本发明仅由权利要求及其等价物来限制。

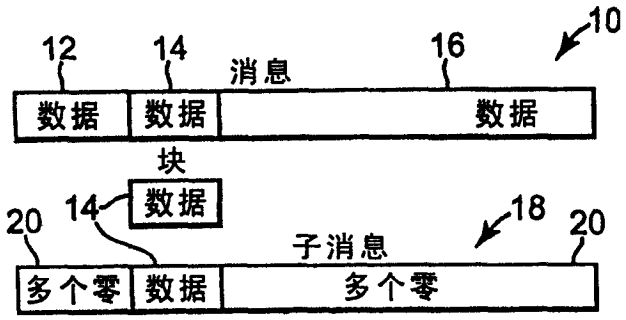

10
12
14
16
消息
数据
数据
数据
块
数据
14
20
18
20
子消息
多个零
数据
多个零

图 1

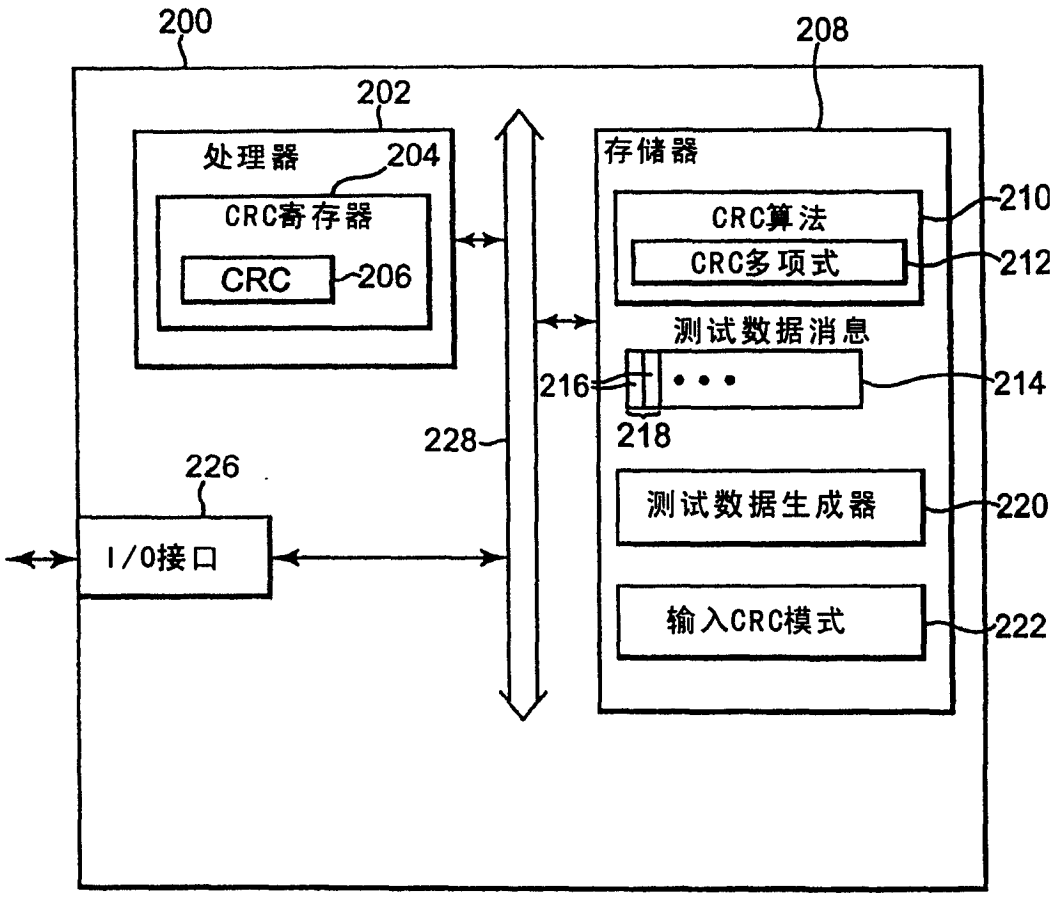

200
208
202
处理器
204
CRC寄存器
CRC
206
存储器
CRC算法
210
CRC多项式
212
测试数据消息
216
214
218
228
226
测试数据生成器
220
I/O接口
输入CRC模式
222

图 2

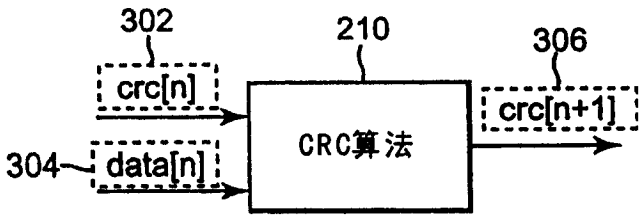

302
crc[n]
210
306
crc[n+1]
304
data[n]
CRC算法

图 3

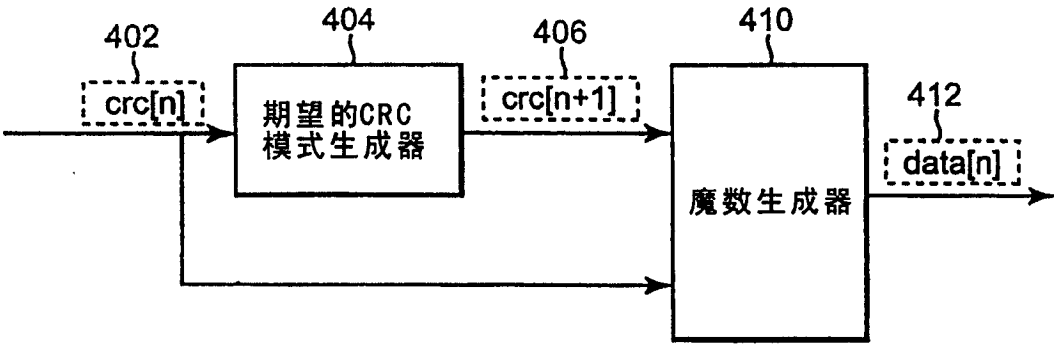

402
crc[n]
404
期望的CRC
模式生成器
406
crc[n+1]
410
魔数生成器
412
data[n]

图 4

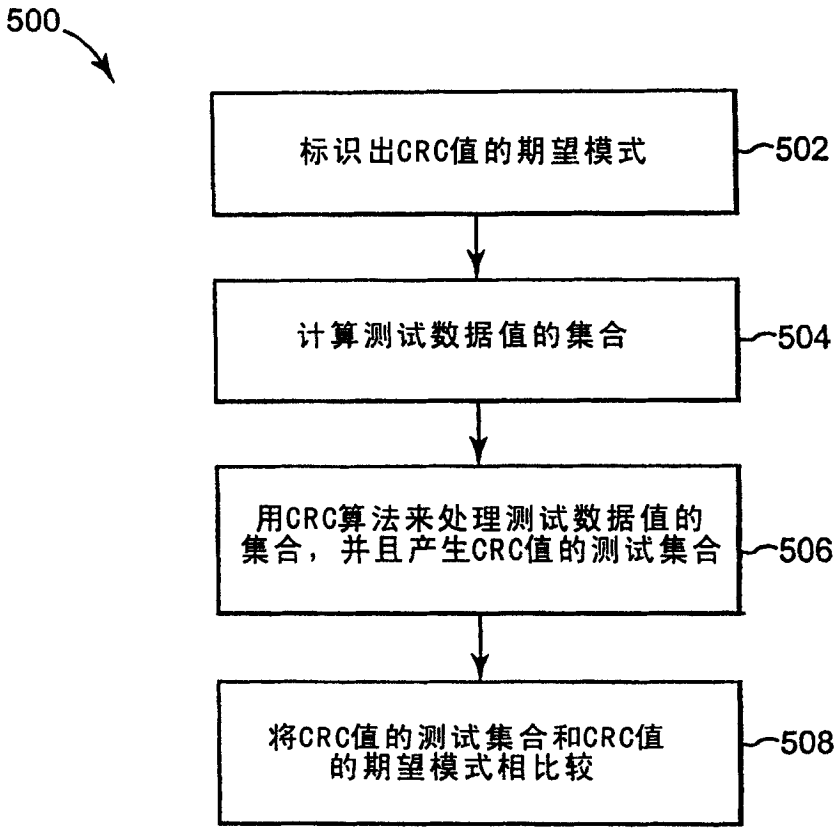

500
标识出CRC值的期望模式
502
计算测试数据值的集合
504
用CRC算法来处理测试数据值的集合，并且产生CRC值的测试集合
506
将CRC值的测试集合和CRC值的期望模式相比较
508

图 5