文章编号:1673 - 0062(2011)03 - 046 - 04

一种快速循环冗余校验电路的设计

张 莹[1],冯 军[2]

(1. 南华大学 机械工程学院,湖南 衡阳 421001;2. 南华大学 电气工程学院,湖南 衡阳 421001)

摘 要:本文简单介绍了循环冗余校验的基本原理. 以国际标准 CRC-CCITT 为研究对象,从串行实现的电路结构出发,通过理论推导,得出了基于逻辑设计的高速 CRC 并行实现矩阵递推公式. 分别设计了这两种结构的 CRC-CCITT 硬件实现电路,并利用 ModelSim6.2 软件进行了功能和时序仿真:用 16bit 位宽的并行 CRC 电路对 32bit 数据进行计算,经过 2 个时钟周期得到校验码.

关键词:循环冗余校验;现场可编程门阵列;高速并行算法

中图分类号:TN911.22 **文献标识码**:B

A Design of Fast CRC Circuit

ZHANG Ying[1], FENG Jun[2]

(1. School of Mechanical Engineering, University of South China, Hengyang, Hunan 421001, China;
2. School of Electric Engineering, University of South China, Hengyang, Hunan 421001, China)

Abstract: The basic principles of CRC are briefly introduced in this paper. Taking the study of CRC-CCITT cyclic redundancy check code in international standard, starting from the circuit architecture of CRC serial implementation, the matrix recurrence formula for CRC parallel implementation based on logic analysis is presented by detailed theoretical derivation. The two structures of CRC-CCITT circuits are designed, and the function simulation and timing simulation are carried out using ModelSim6.2; 32 bit data was calculated by 16bit parallel CRC circuit, and only two clock period later the CRC-CCITT code was obtained.

key words: CRC; FPGA; high-speed parallel algorithm

0 引 言

在数字通信系统,特别是射频数字通信系统中,由于信道传输特性不理想以及加性噪声的影响,接收端所收到的数字信号不可避免的会发生错误. 循环冗余校验[1](Cyclic Redundancy Check):是一种实现简单、校验能力很强、在数据通信领域中广泛使用的一种差错校验方法,其特

收稿日期:2011 - 08 - 19

基金项目:衡阳市科技局科技计划基金资助项目(2010KG45)

作者简介:张 莹(1979 -),女,湖南岳阳人,南华大学机械工程学院实验师,硕士. 主要研究方向:事机电一体化.

征是信息字段和校验字段的长度可以任意选定.

1　CRC 编码原理

CRC 码共由两部分构成:K 位有效信息数据和 R 位 CRC 校验码,一起构成一个总长为 $N=K+R$ 位的二进制序列. 其中校验码的具体生成过程为:假设发送的有效信息用数据多项式 $M(x)$ 表示,将 $M(x)$ 左移 R 位,则可表示成 $M(x)\times 2^R$,这样 $M(x)$ 右边空出的 R 位即是校验码的位置. 用 $M(x)\times 2^R$ 除以生成多项式 C(事先指定的 R 次多项式)得到商 $Q(x)$ 和余数 $R(x)$,其中余数 $R(x)$ 就是校验码.

即:

$$\frac{X^R M(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)} \qquad (1)$$

在发送端发送数据时,把校验码(余数)加到有效信息数据之后一同发出,将这一组由效信息码和校验码组成的数据块称为一个码元,设为 $T(x)$,则有:

$$T(x) = X^R M(x) + R(x) \qquad (2)$$

在接收端收到的任何码元即多项式 $T(x)$ 都应被生成多项式 $G(x)$ 整除,若码元在传输过程中发生错误,则接收的码元可能不能被整除而有余数,因此可以用余数是否为零来判断接收到的码元是否正确.

传输错误的码元也可能出现被 $G(x)$ 整除的情况,这是 CRC 校验无力消除的,但可以通过选择多项式 $G(x)$ 和增加冗余位数,使校验码 $R(x)$ 多项式的位数增多,来降低发生这种错误的概率.

2　常用的 CRC 编码

CRC 生成多项式由协议规定,国际标准中常用的生成多项式如表 1 所示.

表 1　国际常用的 CRC 码

Table 1　CRC code commonly used in international

标准	生成多项式
CRC-12	$X^{12}+X^{11}+X^3+X^2+X+1$
CRC-16	$X^{16}+X^{15}+X^2+1$
CRC-CCITT	$X^{16}+X^{12}+X^5+1$
CRC-32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$

生成多项式的结构和检错效果是经过严格的数学分析与实验后确定的. CRC-12 码通常用来传送 6 bit 字符,CRC-16 及 CRC-CCITT 码则用是来传送 8 bit 字符,其中美国采用 CRC-16,而欧洲国家所采用 CRC-CCITT. CRC-32 码常用在点对点的同步传输中.

3　CRC 的算法与电路

3.1　CRC 电路串行实现

串行算法[2-3]实现原理比较简单,串行 CRC-CCITT 校验码产生器的原理图如图 1 所示. 只需要移位寄存器和异或门等基本的逻辑器件,很适合 FPGA 等硬件电路. 线性移位寄存器一次移一位,完成除法功能,异或门完成不带进位的减法功能. 如果商为‘1’,则从被除数的高价位减去除数,同时移位寄存器右移一位,准备为被除数的较低位进行运算. 如果商为‘0’,则移位寄存器直接右移一位.

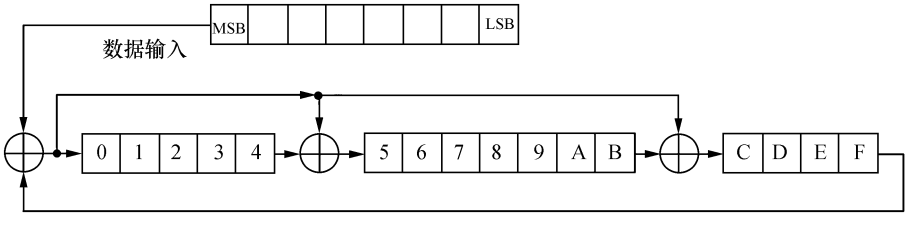

图 1　线性移位寄存器实现 CRC-CCITT 原理图

Fig. 1　Schematic diagram of CRC-CCITT realized by linear shift register

串行法一个时钟周期只能计算一位数据,只适用于数据串行传输的场合,接入并行处理的 CPU 时会大大降低效率.

3.2　CRC 电路并行实现

并行算法可以在一个时钟内对多位数据进行编码,提高计算速度. 信息码一次并行输入,经过必要的处理时间即可输出编码结果,大大缩短了处理时间,具有很大的优越性. 目前采用的 CRC 并行算法有查表法[4]和计算法[5-6].

以传送 8 bit 字符的查表法为例,因为 8 bit 数

据与固定的生成多项式进行 CRC 运算后只有 2^8 即 256 种可能的组合值,因此可以先计算好它们的 CRC 值并存入一张表中,这样便大大简化并加快了 CRC 的计算.

计算法一般算法比较复杂,但是可以省去规模庞大的 CRC 余数表(余数表随着并行位宽的增大呈指数增加),下面是一种利用矩阵递推的 CRC 并行算法.

设生成多项式为 $G(g_0g_1\cdots g_m)$,输入数据为 $D(d_0d_1\cdots d_m)$,输出为 $X(x_0x_1\cdots x_m)$.

构造一个 $m \times m$ 矩阵

$$F^1 = \begin{bmatrix} g_{m-1} & 1 & 0 & \cdots & 0 \\ g_{m-2} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 \\ g_1 & 0 & 0 & \cdots & 1 \\ g_0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

若以 $X(0)$ 表示寄存器的初始状态,$X'(0)$ 表示寄存器的次态,并以⊗符号表示相"与"以后再"异或"运算,⊕ 符号表示"异或"运算,则:

$$X'(0) = F^1 \otimes X(0) \oplus D$$

$$X'(1) = F^2 \otimes X(1) \oplus D,$$

其中

$$F^2 = \left[\begin{array}{cc|cccc} F_1^1 \otimes g' & g_{m-1} & 1 & 0 & \cdots & 0 \\ F_2^1 \otimes g' & g_{m-2} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 1 \\ F_{m-2}^1 \otimes g' & g_1 & 0 & 0 & \cdots & 0 \\ F_{m-1}^1 \otimes g' & g_0 & 0 & 0 & \cdots & 0 \end{array}\right]$$

可以递推出:

$$X_m' = F^m \otimes X_m \oplus D$$

其中

$$F^m = [F^{m-1} \otimes g' \mid \cdots F \otimes g' \mid g'],$$

$$g' = (g_{m-1}\cdots g_1 g_0)^{\mathrm{T}}.$$

每次的并行处理位数为 m,则经过 $\frac{k+m}{m}$ 个时钟周期,电路输出 CRC 附加码. 并行 CRC 电路如图 2 所示,F^m 为电路的控制矩阵.

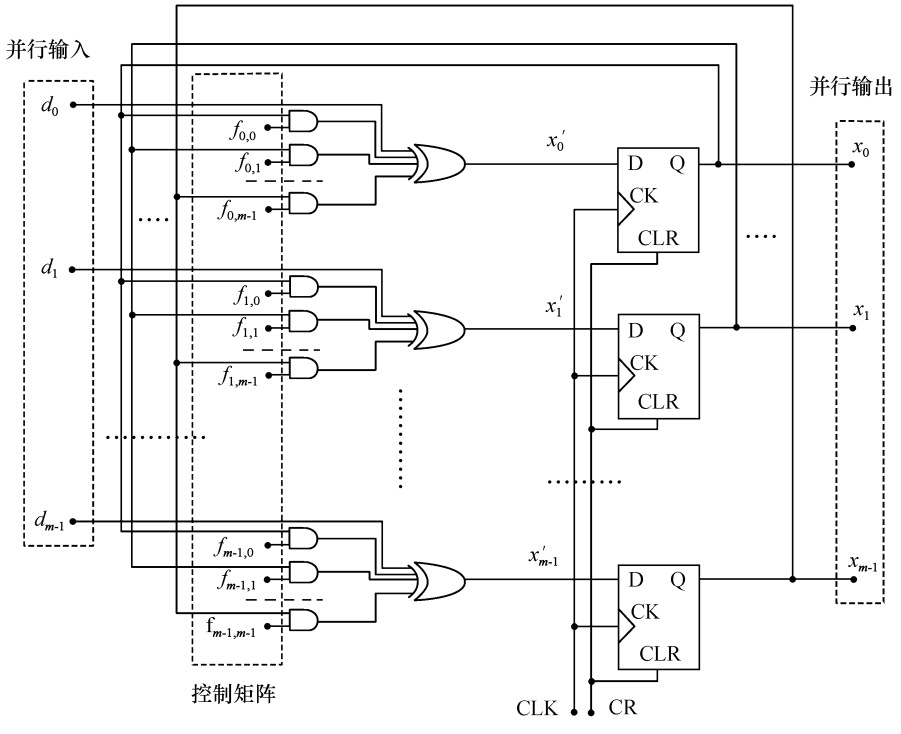

图 2 CRC 并行实现原理图

Fig. 2 Schematic diagram of parallel CRC

3.3 CRC 算法的仿真验证

用 VHDL 语言分别对上述两种算法完成电路描述以后,本文利用 ModelSim6.2 仿真软件对程序进行功能和时序仿真[7].

首先对线性移位寄存器实现 CRC-CCITT 电路进行仿真,仿真的有效数据为 32 位 16 进制数"ABCD0734",经过 32 个时钟周期以后,生成的校验码是 16 进制数"7F8F". 如图 3 所示.

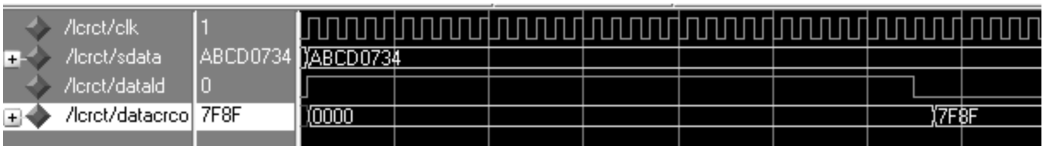

图 3　串行实现时序仿真图

Fig. 3　Timing simulation diagram of serial implementation

对并行实现 CRC-CCITT 电路进行仿真,仿真的有效数据同样为 32 位 16 进制数“ABCD0734”,经过 2 个时钟周期以后,生成的校验码是 16 进制数“7F8F”,和串行电路相同,如图 4 所示.

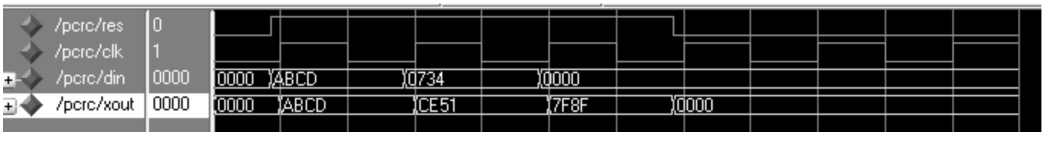

图 4　并行实现时序仿真图

Fig. 4　Timing simulation diagram of parallel implementation

4　结束语

本文提出的并行算法简单且易于实现,只需修改并行位宽和生成多项式即可实现包括 CRC-12,CRC-CCITT,CRC-32 等常用算法在内的所有 CRC 算法. 但是,随着并行位宽的增大,电路所占资源量会迅速增加,在系统设计中可根据实际对速度与系统资源量的关系综合考虑. 本文中利用 16 bit 位宽的并行 CRC 算法对 32 bit 数据进行计算,经过 2 个时钟周期得到校验码.

参考文献:

[1] 杨杰,朱建锋,安建平. 无线传输中的循环冗余校验码纠错应用扩展[J]. 北京理工大学学报,2005,25(8):726-729.

[2] 李宥谋,房鼎益. CRC 编码算法研究与实现[J]. 西北大学学报,2006,36(6):895-898.

[3] 程军,陈贵灿,姜飞. USB 数据传输中 CRC 校验码的并行算法实现[J]. 微电子学与计算机,2003,20(3):77-80.

[4] 刘会杰,张乃通. 基于查表法的快速 CRC 算法设计[J]. 通信技术,2002,124(4):8-10.

[5] 将安平. 循环冗余校验码(CRC)的硬件并行实现[J]. 微电子学与计算机,2007,24(2):107-109.

[6] 刘新宁,王超,胡晨,等. 一种快速 CRC 算法的硬件实现方法[J]. 电子器件,2003,26(1):88-91.

[7] 黄智伟. FPGA 系统设计与实践[M]. 北京:电子工业出版社,2005.