·博士论坛·

# 一种并行数据输入的循环冗余校验码算法设计

尹震宇　赵　海　孙佩刚　林　恺　罗玎玎
(东北大学信息科学与工程学院,沈阳 110004)
E-mail:cmy@neuera.com

**摘　要**　文章首先介绍了CRC的数学原理,继而讨论了一种利于硬件实现的并行数据输入CRC算法的推导方法及其实现方法。最后,采用该文提出的设计算法,使用VHDL设计并实现了CRC-6运算模块,与其它算法实现的CRC模块相比,在使用的资源增加不大的情况下,可以获得较高的性能。

**关键词**　循环冗余码校验　多项式运算　线性编码　数据校验

**文章编号** 1002-8331-(2006)27-0001-02　**文献标识码** A　**中图分类号** TP302.7;TP316.2

# The Design of Cyclical Redundancy Check Arithmetic Based on Parallel Data Input

YIN Zhen-yu　ZHAO Hai　SUN Pei-gang　LIN Kai　LUO Ding-ding
(School of Information Science & Engineering, Northeastern University, Shenyang 110004)

**Abstract**: In this paper, the principle of Cyclical Redundancy Check(CRC) is described. Furthermore, the CRC arithmetic with the parallel data input structure is described. It can be easily implemented by using VLSI. Also, in this paper, a CRC-6 module is designed and implemented by using VHDL as the example. The CRC module which is designed by using this method has high performance.

**Keywords**: Cyclical Redundancy Check, polynomial arithmetic, linear coder, data error check

随着计算机技术的不断发展，数据通讯的应用越来越广泛。在数据传输过程中,由于传输距离、突发的干扰信号等许多可能出现的因素影响,数据在传输过程中常会发生无法预测的错误。为防止这些错误所带来的影响,一般在通讯时采取数据校验的办法,而循环冗余码校验(Cyclical Redundancy Check,简称CRC)是最常用的一种校验方法[1,2]。

本文讨论了一种利于硬件实现的并行数据输入的CRC算法及其实现方法。并且采用本文提出的算法，设计实现了CRC-6运算模块,在测试中达到600Mbps的峰值数据吞吐率。

## 1　CRC生成的代数描述

CRC校验的基本原理是采用线性编码理论，其数学基础是多项式代数。在运算中,使用多项式来表示要处理的数据,其中每一组二进制码均被看作多项式的系数[1,3]。例如,二进制码110011可以使用一个6次多项式$A(x)$表示,具体表示方法如式(1)所示。

$$A(x)=1\cdot x^5+1\cdot x^4+0\cdot x^3+0\cdot x^2+1\cdot x^1+1\cdot x^0 \tag{1}$$

公式(2)为生成$r$位CRC校验码的生成基本表达式。为了便于实现,规定算法中的加法采用的是无进位模2加法。

$$\frac{B(x)\cdot x^r}{G(x)}=Q(x)+\frac{R(x)}{G(x)} \tag{2}$$

在公式(2)中,$B(x)$表示发送端要发送的一组$k$位的二进制码序列。$G(x)$为生成多项式,在CRC码的计算中,将要发送的数据$B(x)$左移$r$位,与生成多项式$G(x)$做除法运算。在公式(2)的左项中,$Q(x)$代表除法操作后的整数商部分,而$R(x)$代表的是除法操作后的余数部分。其中$R(x)$即为需要求得的$r$位CRC校验码。在数据发送的时候,将$r$位的校验码$R(x)$附着在$k$位的二进制码原始信息后面,生成发送序列$T(x)$,如式(3)所示,并将生成的$T(x)$发送出去。

$$T(x)=B(x)\cdot x^r+R(x) \tag{3}$$

在接收端,通过除法运算$T(x)/G(x)$,计算并求其余数,如果最终的余数为0则表示所发送的数据没有差错,否则认为所发送的数据包含错误[3]。

CRC通常分为以下几种标准:CRC-8、CRC-16、CRC-CCITT、CRC-32等[4]。但在一些嵌入式或者移动计算环境中,通讯协议可能更多的是采用自定义的通讯格式，要生成的CRC码并非是常用的标准的8位16位的CRC码,例如需要生成一个7位的或者15位的CRC校验码。对于采用基于数字逻辑电路方式设计的CRC生成模块，同时要求并行输入数据的情况下，并没有一个通用的设计方法可以生成这些特定位数的CRC模块。

下面介绍硬件实现CRC算法的基本方法，基于串行移位寄存器的CRC生成原理。

**基金项目**:国家高技术研究发展计划资助项目(编号:2001AA415320);国家自然科学基金资助项目(编号:69873007)
**作者简介**:尹震宇(1979-),男,博士研究生。赵海(1959-),男,教授,博士,博士生导师。

## 2 基于串行移位寄存器的 CRC 实现

由于 CRC 计算公式中的加法采用的是无进位的模 2 加法,因而,对于 CRC 的实现上可以使用一个线性反馈的移位寄存器(LFSR)来实现[5-7],其电路逻辑结构图如图 1 所示。

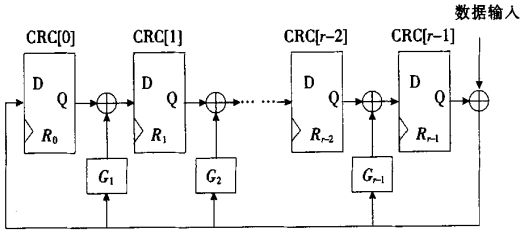


图 1 LFSR 逻辑原理图

在图 1 中,每一个锁存器代表一个 CRC 码位,例如,要生成 $r$ 位的 CRC 码,整个逻辑电路中就要有 $R_0$、$R_1$、$R_{r-1}$ 一共 $r$ 个锁存器。$G_i$ 为生成多项式 $G(x)$的系数,当 $G(x)$中的第 $i$ 位取 1 的时候,表示这个地方是个通路,将数据与 $R_i$ 的输出数据做异或运算;当取 0 的时候,表示在 $R_i$ 的输出端不需要异或操作,$R_i$ 的输出数据直接传送给 $R_{i+1}$。下面以生成一个 6 位的 CRC-6 为例进行分析,在这里,首先规定生成多项式 $G(x)$为:

$G(x)=x^6+x^2+x^1+1$

其中,图 2 所示为其运算的逻辑原理图:

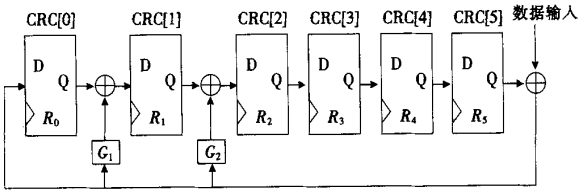


图 2 CRC-6 运算的逻辑原理图

从图 2 中可以看出,要发送的 $n$ 位数据 $B(x)$,在时钟的驱动下,每个时钟将一位数据送入这个电路中,经过 $n$ 个时钟的处理,就可以在锁存器中得到要求的 CRC 码。

上面提到的方法在硬件上设计简单,但是此种方法实际上仅仅适用于串行数据进行 CRC 校验计算。由于每一位需要一个时钟周期进行移位和计算,这对于并行数据而言是不适合的。对于采用硬件方法进行并行数据输入的 CRC 校验计算也有很多解决方式,但对于一些"非标准"的 CRC 校验没有一种通用的方法。因而,需要通过 CRC 的基本数学关系寻找实现的方法。

## 3 CRC 并行算法的实现

下文将以 4 位并行输入的 CRC-6 为例说明将串行 CRC 算法向并行 CRC 算法的转换过程。由 CRC 串行运算的逻辑结构图中可以得到公式(4)、(5)和(6)。

$$R(x)=\sum_{a=0}^{r-1} r_a x^a \tag{4}$$

$$R_b=\sum_{a=0}^{r-1} r_{a,b} x^a \tag{5}$$

$$r_{a,b}=r_{a-1,b-1}\oplus(g_a \& (b_b\oplus r_{r-1,b-1})) \tag{6}$$

这里,$R_b$ 表示的是在计算第 $b$ 位原始数据后,其 CRC 的值。$r_{a,b}$ 表示 $R_b$ 的第 $a$ 位寄存器的内容。$b_b$ 表示要输入的第 $b$ 位原始数据。在串行实现中可以推导出公式(7)、(8):

$$r_{0,2b}=r_{r-1,2b-1}\oplus b_{2b} \tag{7}$$

$$r_{a-1,2b}=r_{a-2,2b-1}\oplus(g_{a-1} \& (b_{2b}\oplus r_{r-1,2b-1})) \tag{8}$$

其中 $a-1=1,2,\cdots,r-1$。

通过对其进行并行化的扩展,可以推导出并行输入的 CRC 计算公式为:

$$r_{0,2j+1}=b_{2j-2}\oplus r_{2,2j-3} \tag{9}$$

$$r_{1,2j+1}=b_{2j-1}\oplus b_{2j-2}\oplus r_{2,2j-3}\oplus r_{3,2j-3} \tag{10}$$

$$r_{2,2j+1}=b_{2j}\oplus b_{2j-1}\oplus b_{2j-2}\oplus r_{2,2j-3}\oplus r_{3,2j-3}\oplus r_{4,2j-3} \tag{11}$$

$$r_{3,2j+1}=b_{2j+1}\oplus b_{2j}\oplus b_{2j-1}\oplus r_{3,2j-3}\oplus r_{4,2j-3}\oplus r_{5,2j-3} \tag{12}$$

$$r_{4,2j+1}=b_{2j+1}\oplus b_{2j}\oplus r_{0,2j-3}\oplus r_{4,2j-3}\oplus r_{5,2j-3} \tag{13}$$

$$r_{5,2j+1}=b_{2j+1}\oplus r_{1,2j-3}\oplus r_{5,2j-3} \tag{14}$$

上述公式将 CRC 计算的数据输入由按位串行输入的变换为 4 位并行输入,其输出 CRC 位数为 6 位。

利用公式(9)~公式(14),就可以实现一个并行数据输入的 CRC 模块,下面是使用 VHDL 语言的 CRC 生成电路描述:

```
function CRC6
( Data:std_logic_vector(3 downto 0);
  CIN:std_logic_vector(5 downto 0))
return std_logic_vector is
variable B:std_logic_vector(3 downto 0);
variable R:std_logic_vector(5 downto 0);
variable CRC:std_logic_vector(5 downto 0);
begin
B:=Data;
R:=CIN;
CRC(0):=B(0) xor R(2);
CRC(1):=B(1) xor B(0) xor R(2) xor R(3);
CRC(2):=B(2) xor B(1) xor B(0) xor R(2) xor R(3) xor R(4);
CRC(3):=B(3) xor B(2) xor B(1) xor R(3) xor R(4) xor R(5);
CRC(4):=B(3) xor B(2) xor R(0) xor R(4) xor R(5);
CRC(5):=B(3) xor R(1) xor R(5);
return CRC;
end CRC6;
```

## 4 性能与评价

通过本文所讨论的 CRC 推导方法,可以很容易地推导出不同的并行输入位宽以及生成 CRC 码位宽的并行输入的 CRC 码算法,并且推导出的公式利于采用 VHDL 或者 VerilogHDL 等硬件描述语言的编程实现。

本文所描述的设计,最终下载到 Xilinx 公司的 Spartan2 系列的 FPGA 芯片中运行,用来完成一套嵌入式系统的通讯协议模块的 CRC 校验工作。表 1 中列出了本设计(CRC-6)和通过 LFSR 实现的 CRC 校验模块的性能比较。

表 1 本设计和 LFSR 实现的 CRC 性能比较

| | CRC-6 | LFSR |
|---|---|---|
| 吞吐率峰值/Mbps | 600 | 200 |
| 最大工作频率/MHz | 150 | 200 |
| 使用的 Slice | 136 | 62 |

表 2 中列出了原系统中通过软件实现的 CRC 校验模块(CRCS-6)和本次设计的 CRC 校验模块(CRC-6)的性能比较。其中 CRCS-6 的运行环境在 ATMEL 公司的 AVR 系列处理器上,处理器的主频为 16MHz。

(下转 5 页)

表 1　一个不对称网络的性能参数

| 工作站 | 类别 | $k_1=2,k_i=1$ $(i=2,\cdots,5)$ | | $k_1=3,k_i=1$ $(i=2,\cdots,5)$ | | $k_1=4,k_i=1$ $(i=2,\cdots,5)$ | |
|---|---|---|---|---|---|---|---|
| | | $w_i^r$ | $\sigma_i^{2r}$ | $w_i^r$ | $\sigma_i^{2r}$ | $w_i^r$ | $\sigma_i^{2r}$ |
| 1 | 实时 | 0.262 7 | 0.033 2 | 0.275 0 | 0.040 3 | 0.280 8 | 0.044 5 |
| | 非实时 | 0.985 0 | 1.073 9 | 0.526 4 | 0.214 9 | 0.441 6 | 0.108 4 |
| 2 | 实时 | 0.284 8 | 0.039 8 | 0.297 0 | 0.047 8 | 0.302 4 | 0.052 3 |
| | 非实时 | 0.527 2 | 0.264 8 | 0.580 1 | 0.366 8 | 0.605 6 | 0.425 1 |
| 3 | 实时 | 0.283 4 | 0.039 6 | 0.296 7 | 0.047 6 | 0.302 6 | 0.051 9 |
| | 非实时 | 0.526 4 | 0.264 6 | 0.577 3 | 0.358 4 | 0.600 5 | 0.414 5 |
| 4 | 实时 | 0.284 0 | 0.039 4 | 0.296 0 | 0.047 1 | 0.302 9 | 0.051 7 |
| | 非实时 | 0.526 1 | 0.259 7 | 0.578 4 | 0.362 5 | 0.606 8 | 0.432 6 |
| 5 | 实时 | 0.283 8 | 0.039 2 | 0.295 6 | 0.046 8 | 0.301 9 | 0.051 4 |
| | 非实时 | 0.524 3 | 0.262 0 | 0.575 3 | 0.355 3 | 0.602 7 | 0.414 9 |

$s_i^1=s_i^2=0.1,a_i\lambda_i^{1\prime}=0.75,\rho'=0.85,u=0.15,u_i=0.2u,i=1,\cdots,5$

的等待队列会变得不稳定。可以看出随着 $k_1$ 的增大，站 1 非实时帧的等待时间减小，但是所有站的实时帧的平均等待时间和方差都在增大。所以网络设计者应当根据对两类帧不同侧重，选择合适的服务参数。

表 2 是个对称的网络例子，即 5 个站平均分配总负荷，但两类帧承担工作站的负荷比仍然为 2:3。仿真 $1\times10^6$s 后发现实时帧的等待时间都低于由(5)式计算出的上界值，而非实时帧的等待时间则没有呈现确定的上界。同本文网络介质访问方法相比，实时环[7]有着类似之处在于每一个工作站也生成实时与非实时两类帧。在每次 SAT 信息(环控制信息)到达后两类帧被授权传输，被授权的数目均被限制，然而这只能使实时帧一部分等待时间有上限，即从被授权到被传输之间的时间段。

表 2　一个对称网络实时帧的等待时间上界

| $k_i(i=1,\cdots,5)$ | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| 仿真结果 | 1.049 9 | 1.620 0 | 2.640 3 | 3.794 8 |
| 解析结果 | 1.450 0 | 2.350 0 | 4.150 0 | 6.950 0 |

$s_i^1=s_i^2=0.1,a_i\lambda_i^{1\prime}=0.75,\rho'=0.85,u=0.15,u_i=0.2u,i=1,\cdots,5$

## 5　结论

本文仿照工业实时系统提出了受限泊松帧到达过程，使相邻帧到达时刻的间距不小于一个固定值。并且提出了一种新的基于轮询的网络模型，其中每一个工作站都到达两类数据帧，一类是实时帧到达服从受限泊松分布，一类是非实时帧到达服从标准泊松分布。本文采用穷尽式策略传输实时帧，而用 $k$-有限式策略传输非实时帧。通过计算机仿真的方法为一个给定的网络选择合适的服务参数。

本文分析了实时帧的等待时间上界，仿真结果表明实时帧的等待时间总是低于它的上界。所以研究结果可以应用在安全关键领域，如运输和工业自动化。(收稿日期：2006 年 8 月)

## 参考文献


1.Stefano Vitturi, Daniele Miorandi.Hybrid ethernet/IEEE 802.11 networks for real-time industrial communications[C].In:ETFA 10th IEEE Conference on,2005:443~449

2.曹春生等.实时无线局域网介质访问控制协议的仿真[J].系统仿真学报,2005;17(3):718~721

3.Decotignie J D.Ethernet-based real-time and industrial communications[J].Proceedings of the IEEE,2005;93(6):1102~1117

4.Francisco B C et al.Real-time communication in unconstrained shared ethernet networks:the virtual token passing approach[C].In:ETFA 10th IEEE Conference on,2005:425~432

5.Krommenacker N,Divoux T,Roundeau E.Using genetic algorithms to desigh switched Ethernet industrial networks[J].Industrial Electronics,2002;1(1):152~157

6.Jorge A Cobb,Miaohua Lin.On-time timed-token protocols[C].In:Global Telecommunications Conference 2002,IEEE,2002:2255~2259

7.Marco Conti et al.Design and analysis of RT-Ring:a protocol for supporting real-time communications[J].IEEE Transactions on Industrial Electronics,2002;49(6):1214~1226

8.L C Hwang,C J Chang.An exact analysis of an asymmetric polling system with mixed service discipline and general service order[J].Computer Communications,1997;20:1292~1300


(上接 2 页)

表 2　本设计和软件实现的 CRC 性能比较

| | CRC-6 | CRCS-6 |
|---|---|---|
| 吞吐率峰值/Mbps | 600 | 1.2 |
| 工作频率/MHz | 150MHz | 处理器主频 16MHz |
| 占用资源 | 136Slices | 约 50 行汇编 |

通过比较可以得出，本文提出的设计，可以达到 600Mbps 的峰值吞吐率，而和 LFSR 相比，占用的 Slices 资源仅仅提高了约一倍。

## 5　总结

本文提出了一种并行输入结构的 CRC 校验码电路的推导方法，并给出了由此算法实现的 4 位输入的 CRC-6 的代码设计例子。通过与 LFSR 方法和基于 AVR RISC 处理器汇编语言实现的 CRC 运算模块的性能作比较，最终得出，采用此种设计方法设计的 CRC 模块，可以在使用资源增加不大的情况下，获得较高的性能。

此外，通过本文描述的推导算法所生成的 CRC 计算公式，便于使用 VHDL、VerilogHDL 等硬件描述语言描述。并且，本文所设计的模块也适于封装成库应用于 VLSI 或者 SoC 环境。

(收稿日期：2006 年 8 月)

## 参考文献


1.Telecommunications and information exchange between system local and metropolitan area networks specific requirements Part 3[S].IEEE Std802.3,Institute of Electrical and Electronics Engineers,Inc,1998:62~65

2.Rajesh Nair.A symbol based algorithm for hardware implementation of cyclic redundancy check(CRC)[C].In:Proceedings of the 1997 VHDL International User's Forum(VIUF'97),1997:81~82

3.William S Shay.Understanding Data Communications and Networks[M].3rd,Toronto:Thomson Learning,2003:421~446

4.Simon Haykin.Communication Systems[M].4th,New York:John Wiley & Sons Inc,2001:531~534

5.Chris Borrelli.IEEE 802.3 Cyclic Redundancy Check[S].Xilinx XAPP-209,San Jose:Xilinx Inc,2001:2~4

6.James R Armstrong,F Gail Gray.VHDL Design Representation and Synthesis[M].Second Edition,New York:Prentice-Hall International,Inc,2000:150~162

7.Charles H Roth.Fundamentals of Logic Design[M].5th,Tornoto:Thomson Learning,2003:213~226