【自动化技术】

# CRC－16 算法与 FPGA 实现*

罗志聪[a],孙奇燕[b]

(福建农林大学 a. 机电工程学院;b. 计算机与信息学院,福州　350002)

**摘要**:以 16 位 CRC－16 校验码为例,在对 CRC 校验码原理和一般的串行 CRC 生成算法进行分析的基础上,改进了串行 CRC 算法,并进一步推导出并行 CRC 算法。利用 Quartus II 集成环境和 Verilog HDL 语言工具将算法转变为校验码生成电路,并进行验证比较,最后在 FPGA 上进行了硬件电路的仿真和实现。结果表明,并行 CRC 算法在速度方面明显优于串行 CRC 算法,但会牺牲较大的硬件空间。

**关键词**:CRC－16;串行;并行;Verilog HDL;FPGA

**中图分类号**:TN91　　**文献标识码**:A　　**文章编号**:1006－0707(2010)05－0089－04

现代数据通信要求信息传输具有高度可靠性,即误码率要足够低。然而,数据信号在传输过程中不可避免地会受到噪声干扰,或者信道不理想,从而造成的码间干扰而产生差错,即出现误码。通常信道不理想产生的误码通过均衡的方法进行改善或消除,噪声干扰产生的误码则可以通过差错控制进行消除[1]。差错控制的核心是差错编码,其中循环冗余校验码(cyclic redundancy check,CRC)由于编码和解码方法简单以及检错和纠错能力强等特点,在数据通信及测控领域中有着广泛的应用。CRC 校验码的计算利用软件硬件均能实现,是进行数据传输差错检测的一种很好的手段。本文中以 16 位 CRC－16 校验码为例,在一般的串行 CRC 生成算法基础上改进了串行 CRC 算法,并通过进一步推导给出并行 CRC 算法,最后对算法进行了比较和实现。

## 1　CRC 校验码原理

CRC 校验码的理论基础是线性编码原理。待传送的 $k$ 位二进制数据序列按接收双方预先约定的某种规则进行处理,产生一个 $r$ 位的校验码。将 $r$ 位校验码加在原始数据序列后一同发送出去,在接收端以相同规则对接收数据进行差错校验[2]。CRC 校验过程用数学计算方法描述如下:

若待传送的 $k$ 位二进制数据序列表示为

$$D(x) = d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + ,\cdots, d_i x^i + \cdots d_1 x^1 + d_0 x^0$$

$D(x)$ 序列左移 $r$ 位,即 $x^r D(x)$,将它除以生成多项式 $g(x)$,得商 $Q(x)$ 以及余数 $R(x)$,此处除法为模 2 除,$g(x)$ 是 $r+1$ 位的多项式,公式表示为

$$\frac{x^r D(x)}{g(x)} = Q(x) + \frac{R(x)}{g(x)}$$

余数 $R(x)$ 就是所求的 $r$ 位 CRC 码,将其附加在 $D(x)$ 后形成发送码 $S(x)$。接收端判断 $S(x)$ 是否能被生成多项式 $g(x)$ 整除,若能则传输无差错,否则,传输有差错。

## 2　CRC－16 校验码生成算法

### 2.1　串行算法

设需发送 16 位的二进制数据序列 $D(x)$ = 1010011101000011,采用 CRC－16 方式进行编码。生成多项式为 $g(x) = x^{16} + x^{15} + x^2 + 1$,它的二进制码是 1110000000000101[3]。运算过程如下:

*D*0　1010011101000011000000000000000000

*g*　1100000000000101

---

*D*1　0110011101000011000000000000000000

*g*　1100000000000101

---

*D*2　~~0000~~1110100000011000000000000000

*g*　1100000000000101

---

*D*3　~~00~~1010000001101010000000000

*g*　1100000000000101

---

*D*4　0110000001101000100000000

*g*　1100000000000101

---

*R*　~~00000000~~1101001110000000

经过若干次模 2 运算后,最终得到的 $R$ =

* **收稿日期**:2010－02－23

**作者简介**:罗志聪(1982—),男,福州大学硕士研究生,主要从事集成电路设计研究。

1101001110000000 即为 CRC 校验码。从上面的运算中可以总结一些规律,从而给出串行算法。首先,判断 $Dk$ 的首位。若为 1,与 $g(x)$ 进行模 2 运算;为 0 则不与 $g(x)$ 进行模 2 运算,即可跳过此次运算直接跳到后面的非零位。另外,$g(x)$ 的首位必定是 1,每次模 2 运算后,结果的首位必定被移出,因此只需考虑 $g$ 的后 16 位。构造 1 个 16 位的 CRC 寄存器 $C$,取 $D$ 前 16 位存入 $C$,每次运算将寄存器首位抛弃后左移 1 位。$D$ 的后 1 位移入寄存器中即串行输入 1 位数据,寄存器 $C'$ 代表 $C$ 移位后的状态。

$C$ 1010011101000011
$C'$ 0100111010000110
$g$ 1000000000000101
---
$C$ 1100111010000011
$C'$ 1001110100000110
$g$ 1000000000000101
---
$C$ 0001110100000011
$C'$ 0011101000000110
$g$ 0000000000000000
---
⋮
---
$C$ 0110100111000000
$C'$ 1101001110000000
$g$ 0000000000000000
---
$C$ 1101001110000000

若 $k$ 为数据序列位数,$r$ 为 CRC 码位数,采用此算法算出最终的 CRC 码共要传送 $k+r$ 个数据,即对于 16 位 CRC -16 校验码需传送 32 位数据,可见所需的时钟周期多,传送速度慢。

## 2.2 改进后的串行算法

为此试从 CRC 码数学表达式上推出某种逻辑关系,进行串行算法的改进。仍然以 16 位的数据序列 $D(x)$ = 1010011101000011 为例。CRC 码表示为

$$C_{15}^{i}x^{15} + C_{14}^{i}x^{14} + \cdots + C_{1}^{i}x^{1} + C_{0}^{i} = (d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + \cdots + d_1x + d_0)x^{16}\bmod g(x)$$

式中 $C_{r-1}^{i}$ 为串行输入第 $i$ 个数据后寄存器 $C$ 第 $r-1$ 位的值。若在第 $i$ 个数后新输入一个数据为 $d$,则

$$C_{15}^{i+1}x^{15} + C_{14}^{i+1}x^{14} + \cdots + C_{1}^{i+1}x^{1} + C_{0}^{i+1} = [(d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + \cdots + d_0)x^{17} + d^{16}]\bmod g(x) = (d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + \cdots + d_0)x^{17}\bmod g(x) + d^{16}\bmod g(x) = (C_{14}^{i} + g_{15}C_{15}^{i})x^{15} + (C_{13}^{i} + g_{14}C_{15}^{i})x^{14} + \cdots + (C_{0}^{i} + g_{1}C_{15}^{i})x + g_0C_{15}^{i} + g_{15}dx^{15} + g_{14}dx^{14} + \cdots + g_1dx + g_0d = [C_{14}^{i} + g_{15}(C_{15}^{i} + d)]x^{15} + [C_{13}^{i} + g_{14}(C_{15}^{i} + d)]x^{14} + \cdots + [C_{0}^{i} + g_1(C_{15}^{i} + d)]x + g_0(C_{15}^{i} + d)$$

可得输入 1 位数据 $d$ 前的 CRC 寄存器的值与输入后 CRC 寄存器值的逻辑关系式:

$$C_{j}^{i+1} = C_{j-1}^{i} + g_j(C_{15}^{i} + d)$$

该逻辑关系式在硬件电路上可用移位寄存器实现,如图 1 所示。

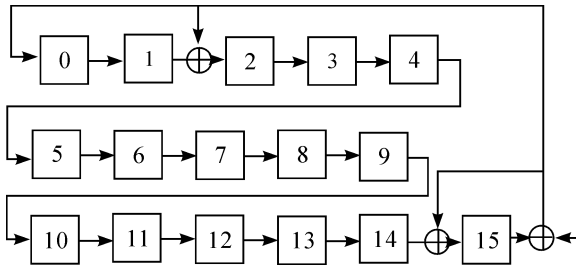


图 1 16 位 CRC - 16 编码器的结构

采用此种算法只需将 CRC 寄存器初始值清零,依次输入 $k$ 位数据后便可得到 CRC 码,而不需要再输入 $r$ 个 0,与前面的串行算法相比所需时钟周期较少。

## 2.3 并行算法

无论采用何种串行算法,它仍然是 1 个时钟周期处理 1 个数据,不适用于高速场合,如果可以在一个时钟周期内并行处理若干位数据,便可以大大地提高速度,因此在串行处理的基础上推导出了并行算法。

将 16 位串行输入数据表示为向量 $D = [d_{15}\, d_{14} \cdots d_1 d_0]$,CRC 寄存器表示为向量 $C^0 = [C_{15}^0 C_{14}^0 \cdots C_1^0 0 C_0]$,CRC 寄存器次态表示为 $C^1 = [C_{15}^1 C_{14}^1 \cdots C_1^1 C_0^1]$,$C^{16}$ 则为串行输入 16 个数据之后 CRC 寄存器的值。串行输入和并行输入最终所要得到的余数应该是相同的,因此可以通过找出 $C^{16}$ 与 $C^0$,$D$ 的关系得到并行算法[4]。由串行算法原理可知:

$$C_0^1 = C_{15}^0 + d_{15}$$
$$C_1^1 = C_0^0$$
$$C_2^1 = C_{15}^0 + d_{15} + C_1 0$$
$$C_3^1 = C_2^0$$
$$\vdots$$
$$C_{13}^1 = C_{15}^0 + d_{15} + C_{12}^0$$
$$C_{14}^1 = C_{15}^0 + d_{15} + C_{13}^0$$
$$C_{15}^1 = C_{15}^0 + d_{15} + C_{14}^0$$

写成如下行列式

$$(C^1)^{\mathrm{T}} = A(C^0)^{\mathrm{T}} + Bd_{15}$$

其中

$$A = \begin{bmatrix} 1100000000000000 \\ 1010000000000000 \\ 1001000000000000 \\ \vdots \\ 0000000000000100 \\ 1000000000000010 \\ 0000000000000001 \\ 1000000000000000 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix},$$

$$(C^2)^{\mathrm{T}} = A(C^1)^{\mathrm{T}} + Bd_{14} = A[A(C^0)^{\mathrm{T}} + Bd_{15}] + Bd_{14} = A^2(C^0)^{\mathrm{T}} + ABd_{15} + Bd_{14}$$

由此可推出

$$(C^{16})^{\mathrm{T}} = A^{16}(C^{0})^{\mathrm{T}} + A^{15}Bd_{15} + A^{14}Bd_{14} + \cdots + ABd_{1} + Bd_{0}$$

该式即为 $C^{16}$ 与 $C^{0}$,$D$ 的逻辑关系。矩阵的乘法可用 Matlab 软件算出,表 1 列出了计算结果[5]。

表 1　16 位并行 CRC 码

| | |
|---|---|
| $C_0^{16}$ | $C_0^0 + C_1^0 + C_2^0 + C_3^0 + C_4^0 + C_5^0 + C_6^0 + C_7^0 + C_8^0 + C_9^0 + C_{10}^0 + C_{11}^0 + C_{12}^0 + C_{13}^0 + C_{15}^0 + d15 + d13 + d12 + d11 + d10 + d9 + d8 + d7 + d6 + d5 + d4 + d3 + d2 + d1 + d0$ |
| $C_1^{16}$ | $C_1^0 + C_2^0 + C_3^0 + C_4^0 + C_5^0 + C_6^0 + C_7^0 + C_8^0 + C_9^0 + C_{10}^0 + C_{11}^0 + C_{12}^0 + C_{13}^0 + C_{14}^0 + d14 + d13 + d12 + d11 + d10 + d9 + d8 + d7 + d6 + d5 + d4 + d3 + d2 + d1$ |
| $C_2^{16}$ | $C_0^0 + C_1^0 + C_{14}^0 + d14 + d1 + d0$ |
| $C_3^{16}$ | $C_1^0 + C_2^0 + C_{15}^0 + d15 + d2 + d1$ |
| $C_4^{16}$ | $C_2^0 + C_3^0 + d3 + d2$ |
| $C_5^{16}$ | $C_3^0 + C_4^0 + d4 + d3$ |
| $C_6^{16}$ | $C_4^0 + C_5^0 + d5 + d4$ |
| $C_7^{16}$ | $C_5^0 + C_6^0 + d6 + d5$ |
| $C_8^{16}$ | $C_6^0 + C_7^0 + d7 + d6$ |
| $C_9^{16}$ | $C_7^0 + C_8^0 + d8 + d7$ |
| $C_{10}^{0}$ | $C_8^0 + C_9^0 + d9 + d8$ |
| $C_{11}^{0}$ | $C_9^0 + C_{10}^0 + d10 + d9$ |
| $C_{12}^{0}$ | $C_{10}^0 + C_{11}^0 + d11 + d10$ |
| $C_{13}^{0}$ | $C_{11}^0 + C_{12}^0 + d12 + d11$ |
| $C_{14}^{16}$ | $C_{12}^0 + C_{13}^0 + d13 + d12$ |
| $C_{15}^{16}$ | $C_0^0 + C_1^0 + C_2^0 + C_3^0 + C_4^0 + C_5^0 + C_6^0 + C_7^0 + C_8^0 + C_9^0 + C_{10}^0 + C_{11}^0 + C_{12}^0 + C_{14}^0 + C_{15}^0 + d15 + d14 + d12 + d11 + d10 + d9 + d8 + d7 + d6 + d5 + d4 + d3 + d2 + d1 + d0$ |

## 3　CRC 校验的仿真与实现

CRC 编码与解码的原理是一样的,因此编码解码电路基本上相同,只是在解码电路中加一个比较器。电路仿真采用 ModelSim SE 5.8c。改进串行输入 CRC 编码仿真结果如图 2 所示。16 位串行输入数据取随机值,每 16 个时钟周期完成一帧数据的编码,fs 作为帧标记。CRC 编码结果与理论值相同。并行输入 CRC 编码仿真结果如图 3 所示,并行输入 16 位数据为 1010011101000011,编码结果与串行输入完全相同,但所需时间大大减少。最后在 ZY11EDA13BE 实验箱 EP1K30QC208 器件上分别实现了串行、并行 CRC 编码、解码器,QuartusII 综合结果如图 4、5 所示。并行 CRC 编码器所占用 22 个 logic elements,34 个 pins,而串行占用 21 个 logic elements,20 个 pins,且当并行 CRC 编码阶数越高时空间占用越大,可见并行 CRC 是以空间换取时间上的优势。

## 4　结束语

本文中在对串行 CRC 算法原理分析的基础上改进了串行 CRC 算法,并进一步推导出 16 位并行数据的 CRC－16 编码表达式,最后进行了对算法的验证和 FPGA 上的硬件实现。验证结果表明测试值与理论值相符合,同时可知并行输入 CRC 编码的方法在当前周期就可以得到编码结果,大大提高了编码速度,适合应用于高速传输或存储数据校验场合[6]。但是由于空间占用大,所以可进一步对组合逻辑单元优化,减少延时、节约硬件资源。

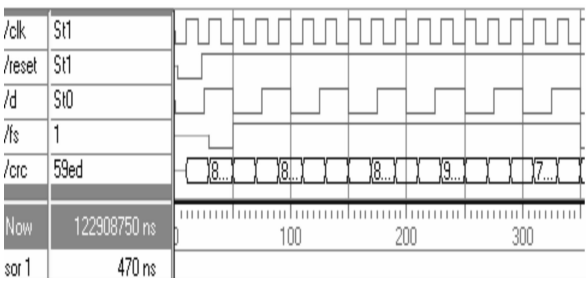


图 2　16 位改进串行 CRC 编码器仿真结果

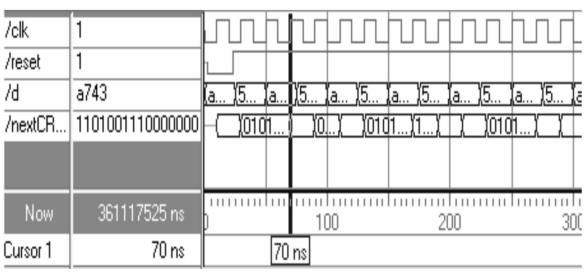


图 3　16 位并行 CRC 编码器仿真结果

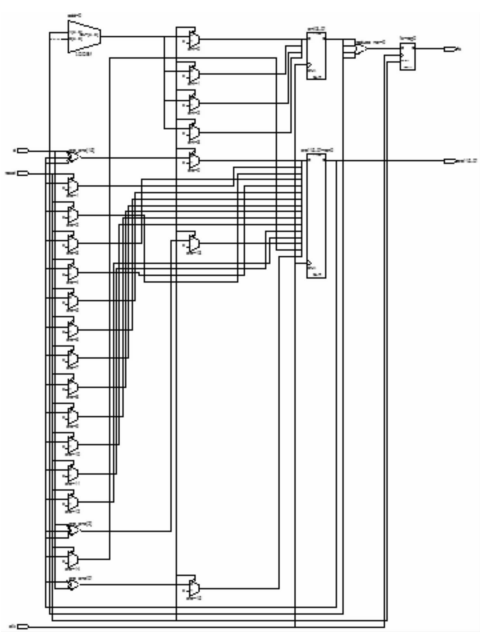

图 4　16 位改进串行 CRC 编码器综合结果

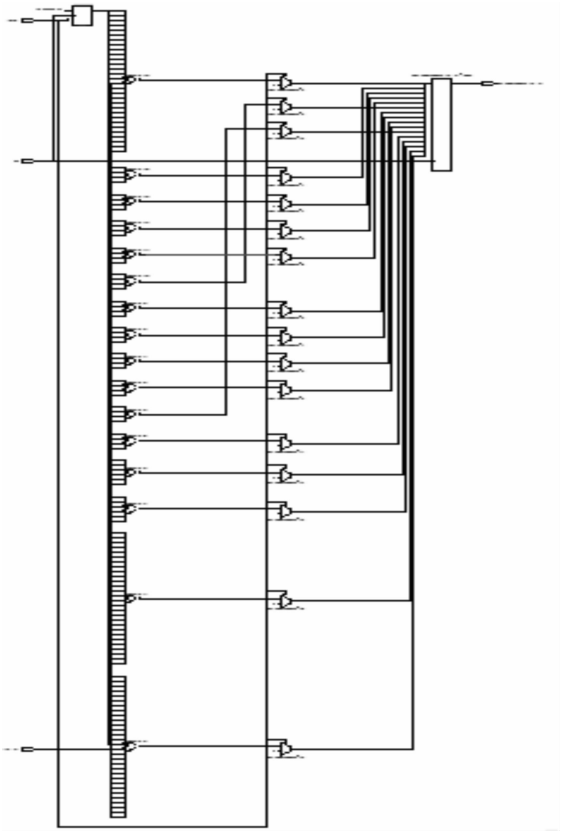

图 5 16 位并行 CRC 编码器综合结果

## 参考文献:


[1] 田耘,文波,延伟. 通信 FPGA 设计[M]. 北京:电子工业出版社,2008.

[2] 刘培培. 基于 CPLD 的循环冗余校验码的实现[J]. 北华航天工业学院学报,2009(1):1 -3.

[3] 顾文斌,王怡,马莉. 基于 FPGA 的 CRC 算法的实现[J]. 计算机与现代化,2008(5):111 -113.

[4] 王海光. 并行 CRC 算法硬件实现研究与 VHDL 设计[J]. 漳州师范学院学报:自然科学版,2007(4):51 -56.

[5] 张树刚,张遂南,黄士坦. CRC 校验码并行计算的 FPGA 实现[J]. 计算机技术与发展,2007(2):56 -58.

[6] 梁少洁,常天海. 循环冗余校验码并行算法的 FPGA 实现[J]. 广东通信技术,2008(2):57 -63.


(责任编辑 刘 舸)

(上接第 88 页)

由于温度对晶体管参数的影响,在测试过程中产生误测,特别是参数靠近上、下限的那些芯片,会由于参数值的漂移或偏差而发生误测现象,造成成品率低下。

把工作环境的温度控制在正常范围之内(一般温度控制在 20 ~25 ℃),是保证测试精度的一个十分重要的措施,一旦温度超出范围,要及时与空调中心联系。

### 1.6 湿度的影响

据有关文献报导,如周围环境的湿度增大时,由于硅片表面吸附水汽,被测芯片的性能会变恶劣。其中 P -N -P 晶体管的表面对湿汽尤为敏感。潮湿的环境使得芯片的电流增益下降,漏电流明显地增大。同时,由于水汽侵入测试设备,设备中各个零部件、元器件、接插件、引出件之间的绝缘电阻下降,漏电流增大。两者综合影响的结果,产生了很大的测试误差。

例如,当我们在潮湿的环境里测试 *Icbo* 时,其数值要比在干燥环境中测试时大几倍到几十倍。特别是到了阴雨季节,问题更为严重,几乎到了无法进行正常测试的地步。当采取措施进行局部环境干燥时,就能使测试精度为之改观。因此,把工作环境的相对湿度控制在正常范围之内,是保证测试精度的一个十分重要的问题。

## 2 结束语

针对中测过程中的误测问题,提出了多种具体方法:适当展宽脉冲电压 *Vcb*、对探针进行腐蚀、采用屏蔽线、遮光和改善温湿度等,大大提高测试的准确性。

## 参考文献:


[1] 马力男,罗卫兵,王文君. 几种 μP 监控芯片之功能比较[J]. 半导体技术,2001(4):36 -38.

[2] John F. Wakerly. 数字设计原理与实践[M]. 北京:机械工业出版社,2003.


(责任编辑 周江川)