|  |
| --- |
| CRC-学习总结 |
|  |
|  |
| 作者：HateHanzo |
| 联系方式：HateHanzo@163.com |

版权所有 侵权必究

**修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修改文件 | 描述 | 作者 | 时间 |
| v1.0 |  | 初稿 | HateHanzo | 20210605 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**缩略语与约定符号**

**缩略语**

|  |  |  |
| --- | --- | --- |
| **英文简写** | **英文全称** | **中文描述** |
| AHB | Advanced High-performance Bus | 先进高性能总线 |
| SRAM | Static Random-Access Memory | 静态随机存取存储器 |
| FSM | Finite State Machine | 有限状态机 |
| SR-Latch | Set-Reset-Latch | SR锁存器 |
| ICG | Integrated Clock Gating Cell | 门控时钟 |

**约定符号**

1、对于某信号名A，若无特殊声明，A\_d1表示该信号延时一拍，A\_d2表示延时两拍，以此类推。如ahb\_addr\_d1表示信号ahb\_addr延时一拍。

**目 录**

[**修改记录** I](#_Toc92556729)

[**缩略语与约定符号** II](#_Toc92556730)

[**目 录** 0](#_Toc92556731)

[**1 CRC概述** 1](#_Toc92556732)

[**2 CRC的简单数学基础** 1](#_Toc92556733)

[**3 串行CRC电路的数学推导** 1](#_Toc92556734)

[**4 并行CRC电路的数学推导** 2](#_Toc92556735)

[**5 初值问题** 3](#_Toc92556736)

[**6 用矩阵求解CRC并行电路** 3](#_Toc92556737)

[**参考文献** 3](#_Toc92556738)

[**附 录** 5](#_Toc92556739)

**1 CRC概述**

循环冗余校验(Cyclic redundancy check,CRC)码是一种优秀的检错码，常用于各种通信场景。将原始数据M经过某种数学运算f(x)后，得到CRC码R，这是一个编码的过程。将R添加到M末尾一并发送出去，接收端对收到的CRC码采取同样的数学运算f(x)，如果余数为0则说明本次接收到的数据正确，否则数据出现错误。

**2 CRC的简单数学基础**

参考[1]，CRC的基本原理是将要发送的信息码抽象为一个多项式M(x),然后定义一个生成多项式g(x)，信息码补若干位0得到M(x)x^(n-k)，然后用M(x)x^(n-k)除以g(x)得到余式，也就是CRC码，此处直接用文献[1]的例子来说明。

定义的生成多项式g(x)=x^16+x^12+x^5+1，那么其对应的二进制数为(10001000000100001)

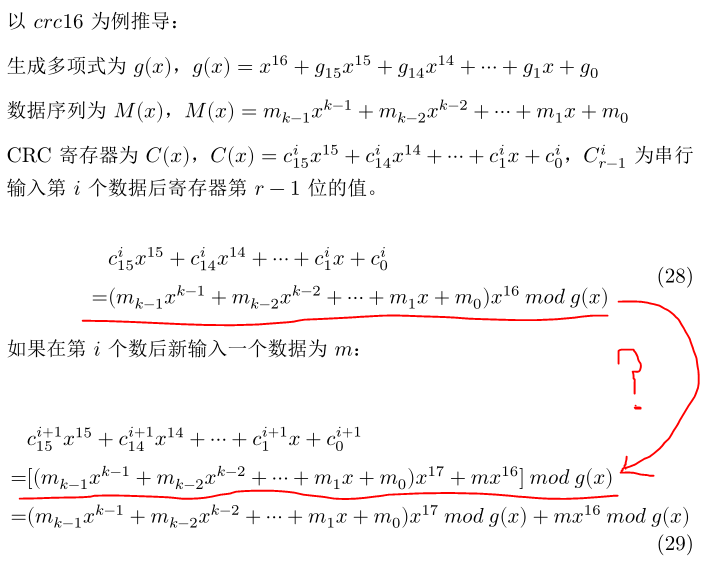
设要发送的数据为(1000001)，那么其对应的多项式为

M(x)=x^6+1(M(x)=1\*x^6+0\*x^5+0\*x^4+0\*x^3+0\*x^2+0\*x^1+x^0)

（未完待续。。。）

**3 串行CRC电路的数学推导**

文献[1]的第3章给出了详细的推理过程，可自行学习，有个比较疑惑的地方是公式(28)如何到公式(29)呢？



后面自己思考了一下，可以从这样的角度去理解，前面输入了i个数据以后再输入第i+1个数据m，那从寄存器的角度去理解的话，会将所有原寄存器的值向“前”推一个时钟周期，注意这里向“前”的意思指的是寄存器数据流的方向，原寄存器的值向“前”推一个时钟周期在数学上就是就是乘以一个x，而新增的数据则可以用m\*x^16来表示。举个简单的例子，输入4个数据后寄存器的值可以表示为

那么在此基础上输入一个新的值*m*，那么有

事实上，串行CRC电路有个万能模板[2]，如图 1所示。本原多项式

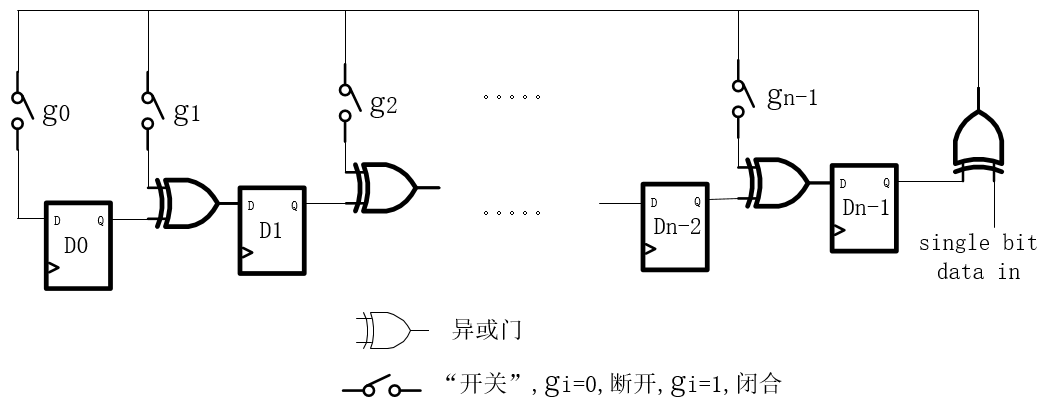


图 1 串行CRC万能模板电路

举个例子，未完待续。。。

**4 并行CRC电路的数学推导**

串行CRC电路简单但效率较低，每次只能输入单bit数据，那么如何将其改为可以输入多bit数据的电路呢？比如将其改为输入数据位宽为[7:0]的电路。首先需要清楚一个前提是，两者电路的寄存器数量是相同的，如果一次性输入8个bit数据后寄存器的状态与串行输入8个bit数据后寄存器的状态是相同的，那么就可以认为二者电路在功能上是等价的。本人通过阅读文献[3]的1.2小节弄明白了推导的过程，接下来结合文献[4]中的串行CRC电路给出具体的推导过程。

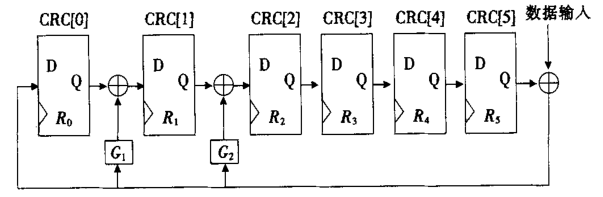


图 2 文献[4]中的串行CRC电路

图 2是文献[4]中的串行CRC电路，其本原多项式为。将其改为4bit数据并行输入的过程如下：

用串行CRC电路完成4个bit数据的编码需要4个时钟周期，设表示的是编号为*j*的寄存器经过*i*个时钟后的值。依据图 2，表示的是寄存器经过了4个时钟周期后的值，,表示的是寄存器经过了3个时钟周期后的值，^是异或符号，D0表示第4个输入的数据，CRC一般是MSB先输入，所以D0是最后输入的数据。不断的通过图 2反推，可以得到，就表示的是寄存器的初始值，所以有。

接下来只需要分别推出、、、、和寄存器初值/输入值的关系式就行了。

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |

由(1)、(2)、(3)、(4)可得

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

对于

|  |  |  |
| --- | --- | --- |
|  |  | (6) |
|  |  | (7) |
|  |  | (8) |

由(6)、(7)、(8)可得

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

剩下、、推理过程类似。图 3是文献[4]给出的并行CRC电路的VHDL描述，可用作推理结果参考。

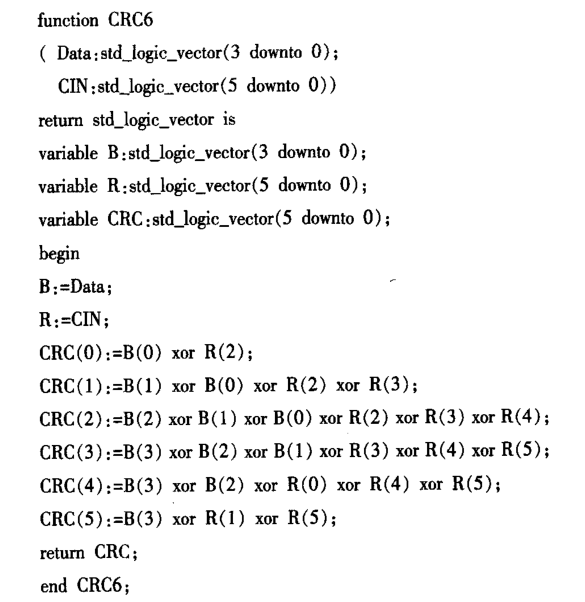


图 3 4bit并行CRC电路VHDL描述

**5 初值问题**

CRC初值问题文献[1]讲得很清楚了，此处总结一下大佬们的观点。

1. 利用长除法求CRC，初值就相当于在信息码前面添加数据。比如长除法求CRC初值为0x84cf，那么就相当于在信息码前面添加0x84和0xcf然后求CRC，此时求出的CRC与没有添加0x84cf求出的CRC是不同的，但是如果初值为全0，那么有没有添加初值求出来的CRC结果都是相同的。
2. 利用3.1节推导出的线性移位寄存器CRC电路，其寄存器的初始值是一个CRC。有一个特例就是当寄存器的初值为0x0000时，利用长除法和线性移位寄存器求出的CRC是相同的，这是因为0x0000的CRC就是0x0000，而长除法计算CRC在前面添加0x0000和没添加的结果也是一样的，所以初值为0的线性移位寄存器CRC与长除法计算CRC结果相同
3. 不同的信息码可以计算出相同的CRC，这是CRC“盲点”的一种理解。大佬Ross N. Williams的观点是CRC初始值的选择不影响CRC算法的有效性，仅仅是提供一个寄存器值开始运算的固定起点。实际工程运用中，有些信息值出现的可能性比较高，由于零字节的CRC结果仍然为零，而零字节在实际应用中充当信息的前导字节非常普遍，这样CRC在它运行的时候就无法“计算”出前面有多少个零字节，所以将初值值设定为非零是比较明智的做法。

**6 用矩阵求解CRC并行电路**

**参考文献**

1. 曹嘉辉，CRC校验一探究竟.pdf
2. IC君，<https://zhuanlan.zhihu.com/p/59666086>
3. 郭宝增, 吴鹏飞. 深亚微米并行CRC32编码芯片的设计和实现[J]. 河北大学学报：自然科学版, 2014, 34(1):5.
4. 尹震宇, 赵海, 孙佩刚,等. 一种并行数据输入的循环冗余校验码算法设计[J]. 计算机工程与应用, 2006, 42(27):1-2.

**附 录**