DOI:10.14022/j.cnki.dzsjgc.2011.15.017

并行 CRC 在 FPGA 上的实现

莫元劲，黄水永
（广州数控设备有限公司 广东 广州 510165）

摘要：循环冗余码校验 CRC（Cyclic Redundancy Check）广泛用于通讯领域和数据存储的数据检错。基于 FPGA 在通讯领域和数据存储的应用越来越广泛，CRC 的编码解码模块已经是 FPGA 上的常用模块了。采用超前位计算实现 CRC 在 FPGA 上的并行运算，通过实际应用证明该算法能有效实现硬件的速度与资源合理平衡。

关键词：CRC；FPGA；超前位；并行
中图分类号：TP33.1 **文献标识码**：A **文章编号**：1674-6236(2011)15-0133-03

CRC parallel computing in FPGA

MO Yuan-jin, HUANG Shui-yong
(*GSK CNC EQUIMENNT CO.,LTD. Guangzhou* 510165, *China*)

Abstract: The Cyclic Redundancy Check (CRC) is wildly used in the data communication systems and data store systems for error checking. As FPGA become more and more important in the networks and the data access systems, CRC coding and encode have become frequently used common modules in FPGA. This paper realized the CRC parallel computing on FPGA by using bitahead computing. It shows that the algorithm can effectively achieve the balance between speed and resource of hardware through the practical application.

Key words: CRC; FPGA; lookahead; parallel

循环冗余码校验 CRC 广泛用于通讯领域和数据存储的数据检错。很多时候，人们往往选择 C 语言来实现 CRC 对数据的校验检错，实现起来也比较简单，但是其流水线处理方法大大限制了对数据流处理的速度。随着 FPGA 的大量普及，并行处理数据有着传统流水线无可比拟的巨大优势。使用超前位并行计算方法计算 CRC，对通讯领域和数据存储的快速数据处理有着重要的意义，笔者根据超前位计算方法得到在 FPGA 上面的并行 CRC 通用计算方法，并可以根据需要，实现数据级联计算，得到快速计算数据流的 CRC 方法。

1 CRC 编码解码原理

1.1 CRC 编码

设 $M(x)$为信息多项式，$G(x)$为生成多项式，得到的 CRC 位数为 r，则 $M(x)$左移 r 位，作模 2 除法，得到的商为 $Q(x)$，余数多项式 $R(x)$即是信息多项式的 CRC 校验码，数学表示为[1]：

$$\frac{M(x)\cdot x^r}{G(x)}=Q(x)+\frac{R(x)}{G(x)} \quad (1)$$

1.2 CRC 解码

为确保数据信息的正确性，信息数据经过 CRC 编码得到 CRC 校验码，校验码和信息数据一起进行数据封装。如图 1 所示。

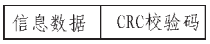

图 1 CRC 校验数据封装
Fig. 1 CRC data check frame

读取信息数据加 CRC 校验码的时候，再用相同的生成多项式进行校验，若校验结果不正确，那证明数据信息丢失或者已经被破坏。数学表示为：

$$\frac{M(x)\cdot x^r-R(x)}{G(x)}=Q(x)+0 \quad (2)$$

余数多项式结果为 0 为正确[2]，其余结果都说明数据不正确（有些像 IEEE802.3 中的 CRC，因为特别的取反处理，余数多项式不是 0，而是等于 0xC704DD7B[3]）。典型 CRC 在 FPGA 上的应用如图 2 所示[4]。

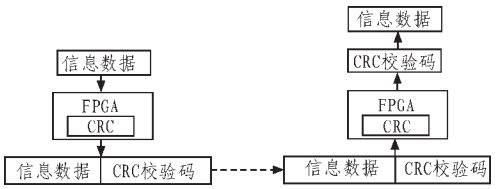

图 2 CRC 在 FPGA 上的典型应用
Fig. 3 CRC module application on FPGA

2 串行 CRC 编码解码实现

CRC 的编码解码可由一般的串行流水线算法来获得，但是很多时候硬件接口串行的，并行接口也进行串行流水线的

收稿日期：2011-05-30 稿件编号：201105127
作者简介：莫元劲（1985—），男，广东遂溪人。研究方向：总线通信、运动控制与 FPGA。

CRC 计算显然不合理。如:最常用的 CRC32 生成多项式为:$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$[5],串行流水线的硬件实现如图 3 所示[4]:

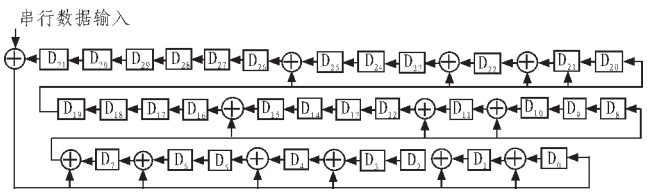

图 3　串行 CRC32 硬件实现

Fig. 3　Serial CRC32 hard ware

(图 3 中的 Dx 为 D 寄存器,⊕为异或运算。)由图(3)可见,要处理 n 个数据输入就要延迟 n 个时钟周期才能得到 CRC 的校验码。而且很多时候往往硬件接口不是串行数据输入,而是并行数据输入,这就要求要把数据流进行并行串行转换,处理来很麻烦,不能满足数据流的快速处理。下面介绍一种可以在硬件上实现,基于 FPGA 并利用超前位计算方法得到任意长度数据的并行 CRC 计算方法。该算法使得数据量很大的数据流可以在数据接收或者发送的开始时候就可以计算其 CRC 的校验值,而不是把整个需要校验的数据放到缓存区再开始计算校验值,达到真正的并行数据处理目的。

3　并行 CRC 的实现

具体的算法描述如下:

假设数据输入的位宽为 m,第 n 个周期输入数据为:

$$D_n(d)=d_n^{(m-1)}+d_n^{(m-2)}+d_n^{(m-3)}+\ldots+d_n^{(0)} \tag{3}$$

假设 CRC 生成多项式阶数为 r:

$$G(g)=g^{(r-1)}+g^{(r-2)}+g^{(r-3)}+\ldots+g^{(0)} \tag{4}$$

假设第 $n-1$ 个周期计算得到的 CRC 校验码为

$$C_{(n-1)}(c)=c_{(n-1)}^{(r-1)}+c_{(n-1)}^{(r-2)}+c_{(n-1)}^{(r-3)}+\ldots+c_{(n-1)}^{(0)} \tag{5}$$

为了计算得到对应第 n 个周期的数据输入 $D_n(d)$ 的 CRC 校验值 $C_{(n)}(c)$,首先把第 n 个周期输入数据 $D_n(d)$ 左移 r 位,扩展为 $(m+r)$ 位,其中低 r 位补 0,即:

$$D`_{(n)}(d)=d_n^{(m+r-1)}+d_n^{(m+r-2)}+\ldots+d_n^{(r)}+0^{(r-1)}+0^{(r-2)}+\ldots+0^{(0)} \tag{6}$$

同时,实现级联运算,把第 $n-1$ 个周期计算得到的 CRC 校验码 $C_{(n-1)}^{(c)}$ 合到第 n 个周期要计算的数据里面,进行模 2 除法计算得到 CRC 校验值 $C_{(n)}(c)$。根据公式(1),$C_{(n)}(c)$ 作为第 n 个周期的余数,那么有第 n 个周期要进行模 2 除法计算的数据为:

$$D``_{(n)}(d)=d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}+\ldots d_n^{(m)}\hat{}c_{(n-1)}^{(0)}\ldots+0^{(0)} \tag{7}$$

(注:^ 为异或运算符号。)

为了快速实现模 2 除法,算法应用了一种超前位计算方法。对于该超前位计算方法,利用模 2 除法运算方法,首先判断 $D``(n)(d)$ 的最高位 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}$ 是否为 1,若 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}$ 为 1 则进行除数为 CRC 生成多项式 $G(g)$ 的模 2 除法;若最高位 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}$ 为 0,那么把除数 $G(g)$ 当作 0,再作模 2 除法,得到计算结果 $D```_{(n)}(d)$,即其运算如图 4 所示:

$$\begin{array}{l} d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}+\ldots d_n^{(m)}\hat{}c_{(n-1)}^{(0)}+\ldots 0^{(0)} \\ \oplus\quad g^{(r-1)}\quad g^{(r-2)}\quad +\ldots g^{(0)} \\ \hline D```_{(n)}=d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}\hat{}g^{(r-1)}+d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}g^{(r-2)}+d_n^{(m)}\hat{}c_{(n-1)}^{(0)}\hat{}g^{(0)}+\ldots 0^{(0)} \end{array}$$

图 4　模 2 除法运算结果

Fig. 4　Result of module-2 division

然后,判断 $D```_{(n)}^{(d)}$ 次高位 $d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}g^{(r-1)}$ 是否为 1,再进行下一步的计算,其运算过程与 $D``_{(n)}(d)$ 最高位的判断计算一样,直至到第 r 步,最后得到的余数则为该数据的 CRC 校验值。

对于最高位的判断运算,因为 CRC 生成多项式 $G(g)$ 的最高位 $g^{(r-1)}$ 为 1,所以不管 $d_n^{(m+r-1)}\hat{}c^{(n-1)\ (r-1)}$ 是不是 1,作模 2 除法的结果是 $D```_{(n)}(d)$ 的最高位为 0。也就是说,不管 $D``_{(n)}(d)$ 的最高位 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}$ 是否为 0,经过第一步计算,其结果都是 $D```_{(n)}(d)$ 的最高位为 0。

考察 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}\hat{}g^{(r-1)}$,要使得它为 0,那么要 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}=g^{(r-1)}$。所以,作模 2 除法时,CRC 生成多项式 $G(g)$ 所有等于 0 的数据位不管何时,都是 0。而 CRC 生成多项式 $G(g)$ 所有等于 1 的数据位可以用 $d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}$ 代替。例如,若 $g^{(r-1)}=1, g^{(r-2)}=1, g^{(0)}=1$,则图 4 中的计算结果等于 $D```_{(n)}(d)=d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+\ldots d_n^{(m)}\hat{}c_{(n-1)}^{(0)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+\ldots 0^{(0)}$。由于 $d_n^{(m-1)}$、$d_n^{(m-2)}$、$d_n^{(m-3)}$、$\ldots+d_n^{(0)}$ 这些都是输入数据 $D_n(d)$ 各位上的已知数据,$c_{(n-1)}^{(r-1)}$、$c_{(n-1)}^{(r-2)}$、$c_{(n-1)}^{(r-3)}$、…、$c_{(n-1)}^{(0)}$ 也都是输入上一次的 CRC 数据 $C_{(n-1)}(c)$ 各位上的已知数据,同时 $g^{(r-1)}$、$g^{(r-2)}$、$g^{(r-3)}$、…、$g^{(0)}$ 是 CRC 生成多项式的各位上的已知数据,那么一次运算就可以得到 $D```_{(n)}^{(d)}=d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+\ldots d_n^{(m)}\hat{}c_{(n-1)}^{(0)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+\ldots 0^{(0)}$ 的值,而不需要通过额外的中间过程变量来计算结果,这样完全符合超前位计算的原则。

对于次高位若 $g^{(r-1)}=1, g^{(r-2)}=1, g^{(0)}=1$,则 $D```_{(n)}(d)=d_n^{(m+r-3)}\hat{}c_{(n-1)}^{(r-3)}\hat{}d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+d_n^{(m+r-4)}\hat{}c_{(n-1)}^{(r-4)}\hat{}d_n^{(m+r-3)}\hat{}c_{(n-1)}^{(r-3)}\hat{}d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}\ldots d_n^{(m)}\hat{}c_{(n-1)}^{(0)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}\hat{}d_n^{(m+r-2)}\hat{}c_{(n-1)}^{(r-2)}\hat{}d_n^{(m+r-1)}\hat{}c_{(n-1)}^{(r-1)}+\ldots 0(0)$。

同理,对于余下的 $r-2$ 次运算采用相同的计算方法进行递推。计算完 r 次之后,得到的数据就是第 n 个周期输入数据 $D_n(d)$,并且第 $n-1$ 个周期计算得到的 CRC 校验码为 $C(n-1)(c)$ 时,对应 CRC 生成多项式 $G(g)$ 的 CRC 校验值 $C_{(n)}(c)$。

例如,假设输入数据宽度为 8 位,CRC 生成多项式的为 $G(g)=g^5+g^4+g^3+1$,那么根据上述算法可以得到在 FPGA 上面的函数实现代码如下:

```
function [4:0]  CRC5_D8;
input [7:0] DATA_in;
input [4:0] CRC_last;
reg [7:0]  d;reg [4:0]  c;reg [4:0]  newcrc;
```

```
    begin
      d = DATA_in;
      c = CRC_last;
      newcrc[0] = d[6] ^ d[5] ^ d[4] ^ d[3] ^ d[1] ^ d[0] ^ c[0] ^ c
[1] ^ c[2] ^ c[3];
      newcrc[1] = d[7] ^ d[6] ^ d[5] ^ d[4] ^ d[2] ^ d[1] ^ c[1] ^ c
[2] ^ c[3] ^ c[4];
      newcrc[2] = d[7] ^ d[6] ^ d[5] ^ d[3] ^ d[2] ^ c[0] ^ c[2] ^ c
[3] ^ c[4];
      newcrc[3] = d[7] ^ d[5] ^ d[1] ^ d[0] ^ c[2] ^ c[4];
      newcrc[4] = d[5] ^ d[4] ^ d[3] ^ d[2] ^ d[0] ^ c[0] ^ c[1] ^ c
[2];
      nextCRC5_D8 = newcrc;
    end
  endfunction
```

程序中,DATA_in 为 8 位的数据输入,CRC_last 为上次的 CRC 校验值,CRC5_D8 为 CRC 计算结果输出。

如果要实现任意长度的数据 CRC 校验，则只需要把上次的数据输入得到的 CRC 校验码作为本次 CRC 校验码输入,那样就可以进行级联,计算得到任意长度的数据 CRC 校验码。

此算法充分利用 FPGA 的并行处理能力和数学算法上的超前位计算原理实现快速有效的 CRC 计算，与传统的流水线法、矩阵法[6]等相比有非常大的优势。该算法对于任意宽度的数据输入,和任意生成多项式也都通用,非常灵活,并且算法是基于递推方法,FPGA 的程序也可以由软件语言 VC++,VB 等来生成得到,使用非常方便实用。

4　结果与仿真

使用的 FPGA 芯片为 ALTERA 公司的低端芯片 EP1C6T144I7,其运算速度最快可达 320 MHz,也就是说 3.2 ns 的时间就可以把 8 位的数据的 CRC 校验码计算出来。实现级联的计算，如在图 5 中，连续输入数据流 0x35、0x65、0x9A、0x3F、0x65、0x10、0xBB、0x8C、0x27、0XB9、0x97、0x8D、0x2C，在刚接完最后一个数据 0x2C 时候就可以得到这一串数据流的 CRC 校验值 0x08。对比要接收完整个数据流再作相应的 CRC 计算有极大的优势。同时使用的 FPGA 资源非常少,只用了 23 个逻辑单元,很好地做到速度与资源的平衡。采用更高端的 FPGA 芯片,速度更高,此算法完全可以满足 10 Gbit 的以太网数据的 CRC 校验计算。

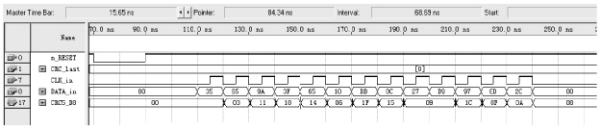

图 5　实现级联的 CRC 仿真结果时序图

Fig. 5　simulated CRC timing of data sequence

5　结束语

以上算法对于任意位数据输入,任意生成多项式都可通用,任意长度的数据通过级联可以得到 CRC 校验码输出,并且采用并行处理和超前位计算,达到非常高的运算速度。其算法的 FPGA 实现应用也非常广泛,不论是一般的串行数据通讯，还是 100 M,1 000 M 甚至 10 GM 的以太网都可以用到。对此 Xilinx,Altera 等 FPGA 公司都开发了相关的 IP(知识产权)。本算法已经成功用于 FPGA 上的串行数据发送及 1 000 MHz/100 MHz 以太网等通讯，达到快速处理数据流检错的效果。

参考文献:

[1] 曹志刚,钱亚生. 现代通讯原理[M].北京:清华大学出版社,1992.

[2] 常天海，胡鉴. 基于 FPGA 的 CRC 并行算法研究与实现[J]. 微处理机,2010,4(2):45-48.
CHANG Tian Hai,HU Jian. Applicating research of CRC parallel algorithm base on FPGA[J]. Microprocessors 2010,4(2):45-48.

[3] Chris Borrelli .IEEE 802.3 Cyclic Redundancy Check [S] 2001.

[4] Sunita Jain & Guru Prasanna. 在 Virtex-5 FPGA 中使用 CRC 硬模块[EB/OL] (2008-12). http://www.doc88.com/P-38579361672. html.

[5] ISO/IEC. IEEE Std 802.3 2000 Edition .Part3.Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.[S]. 2000.

[6] 刘昭,苏厉,金德鹏,等,10G 以太网系统中的并行 CRC 编解码器设计[J]. 电子技术应用,2004,30(4):47-50.
LIU Zhao,SU Li.,JIN De-peng,et al.Parallel CRC Coder and Decoder Designed in 10G Ethernet System. [J]. Application of Electronic Technique,2004,30(4):47-50.