文章编号:1001-2486(2011)06-0048-07

一种改进的旋转CRC数据校验设计方法*

王永庆,张民选
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:旋转CRC同时使用两个生成多项式产生校验数据。之前的方法校验能力弱,报文丢失后的检错失效率很高。提出了一种新的旋转CRC设计方法,通过比较,选取合适的更高次的生成多项式组合,并且修改了检验生成与检测机制,形成MR-CRC。FPGA实现结果表明,这种方法能够在较低逻辑复杂度的基础上提高校验能力,从而改善数据通信的可靠性,而且对系统性能影响甚小。通过比较16位MR-CRC与32位传统CRC的实现数据发现,前者在所用资源减少10%的情况下,频率提高了25%。

关键词:旋转循环冗余校验;生成多项式;数据通信;检错失效
中图分类号:TN911.75 **文献标识码**:A

Design and Implementation of a Modified Rolling CRC Scheme

WANG Yong-qing, ZHANG Min-xuan
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Rolling cyclic redundancy code (Rolling CRC) uses two generator polynomials for code generation. The degree of error detection capability provided by current implementations is not strong enough to check missing of a whole packet in a series of identical ones. A modified Rolling CRC, named MR-CRC, was presented for the data error checking, which adopted higher degree polynomials, chose the proper combination and revised the schemes used in original generator and detector. The result of FPGA implementation shows that this method has low logic complexity, can improve error detection and provide advanced reliability with little performance depression. Compared with the FPGA results of 16B rolling CRC and traditional 32B CRC, the former can improve the frequency by 25% with 10% less LUTs.

Key words: rolling CRC; generator polynomial; data communication; undetected error

在现代数据通信和计算机通信中,由于信道上各种复杂因素的影响,如噪声、线路间的串扰等,所传输的信号往往会受到干扰,严重时甚至会造成误码,使数据发生错误以致阻断通信。为了提高通信的可靠性和减少误码率,一方面可以通过对特定电路进行精心设计,提高电路的稳定性和可靠性;另一方面则是对数据采用某种编码来进行差错控制,通过少量的附加电路,使之能发现某些错误,甚至能确定出错位置,进而实现自动纠错的功能。CRC(Cyclic Redundancy Check),即循环冗余校验码,是一种实现简单、校验能力很强的编码方式,在众多的信道编码方法中得到广泛的应用。CRC码具有占用系统资源少、软硬件均能实现等优点,是对数据传输过程中进行差错检测的一种很好的手段。当检测到数据信息有错时,可通过软硬件控制对方重发,直至数据信息正确为止。

CRC校验码由一个生成多项式产生,k阶生成多项式可产生$k-1$位的冗余校验码。适当选取生成多项式可以使CRC码检出所有奇数位的随机误码,以及突发长度小于$k-1$的突发误码[1]。

在CRC校验中,将所传数据序列看成是高阶多项式,也称为位序列多项式,用$M(x)$表示,将此多项式用预先规定的生成多项式$g(x)$去除,再将其余数附加在所传数据的尾部发送出去。在接收方,将接收到的完整数据用同样的生成多项式$g(x)$去除,若除得的余数值为0,则认为所接收的数据是正确的,否则,接收的数据有误码。

CRC码的编码过程如下:

设待校验的信息码M有n位,即$M=(m_{n-1},m_{n-2},\cdots,m_1,m_0)$,这$n$个信息位可以看作多项式

* 收稿日期:2011-06-20
基金项目:国家自然科学基金资助项目(60970036)
作者简介:王永庆(1973—),男,副研究员,博士。

的系数,用多项式 $M(x)$ 表示为

$$M(x) = m_{n-1}X^{n-1} + m_{n-2}X^{n-2} + \cdots + m_1X^1 + m_0 \quad (1)$$

如果所采用的生成多项式 $g(x)$ 的阶为 k,则先在式(1)的两端乘以 X^k,得到

$$X^kM(x) = m_{n-1}X^{n-1+k} + m_{n-2}X^{n-2+k} + \cdots + m_1X^{1+k} + m_0X^k \quad (2)$$

将 $X^kM(x)$ 除以生成多项式 $g(x)$,除法采用模 2 除,得到商多项式 $q(x)$ 和余数 $R(x)$,即

$$X^kM(x) + R(x) = q(x)g(x) \quad (3)$$

余数多项式 $R(x)$ 可以表示为

$$R(x) = r_{k-1}X^{k-1} + r_{k-2}X^{k-2} + \cdots + r_1X + r_0 \quad (4)$$

将式(2)和式(4)代入式(3),得

$$\begin{aligned} q(x)g(x) &= X^kM(x) + R(x) \\ &= m_{n-1}X^{n-1+k} + m_{n-2}X^{n-2+k} + \cdots \\ &\quad + m_1X^{1+k} + m_0X^k + r_{k-1}X^{k-1} \\ &\quad + r_{k-2}X^{k-2} + \cdots + r_1X + r_0 \end{aligned} \quad (5)$$

式(5)所对应的码组包含 $n+k$ 位,即

$$M' = (m_{n-1}, m_{n-2}, \cdots, m_1, m_0, r_{k-1}, r_{k-2}, \cdots, r_1, r_0) \quad (6)$$

从 M 到 M' 就是 CRC 的编码过程,m_{n-1}, m_{n-2},$\cdots$,m_1,m_0 为 n 位原始信息码;r_{k-1},r_{k-2},$\cdots$,r_1,r_0 为生成的 k 位校验码。

在接收端,将接收到的 $n+k$ 位码除以相同的生成多项式 $g(x)$,如果余数为 0,表示接收到的数据信息正确,否则表示数据在传输过程中发生了错误。

首先,对于给定的阶数 r,其检错能力为:

1)检查出信息码元的单个错,即任何一位错;

2)检测出信息码元中的离散的双位错;

3)检查出全部信息码元的奇数位错;

4)检查信息码元中,长度等于 r 或小于 r 的突发错;

5)以 $(1-(1/2)^{r-1})$ 的概率,检查出信息码元中的长度为 $(r+1)$ 的突发错。

可以发现,CRC 的校验能力与生成多项式本身及其阶数有着密切关系。阶数低的生成多项式实现逻辑相对简单,但是校验能力较弱,阶数高的生成多项式具有更高的校验能力,但是实现复杂度也更大。如何利用相对简单的校验多项式来实现更强的数据校验能力,是链路可靠性设计的目标之一。

1　相关研究

对于任意阶 r,通常有许多既约本原多项式,每个多项式都给出了不同的一组编码。表 1 给出了常见的 16 位和 32 位 CRC 与对应的生成多项式。其中生成多项式采用简写方式,即去掉多项式中最高项的系数 1,把其余各项的系数(二进制的 0 或者 1)序列转换为十六进制表示。

编码的性能使用检错失效率 P_{ud} 来衡量。检错失效就是虽然发生了错误,但是编码却不能发现的那些错误[4-5]。一般而言,如果错误独立发生,并且每位的出错概率为 1/2,那么 P_{ud} 等于 $(2^{-r}-2^{-n})\sim 2^{-r}$,其中 n 是编码长度,r 是本原多项式的阶[6-7]。这对于所有本原多项式和任意码长都是成立的。

文献[8]研究了 CRC 的检错失效率问题,认为 2^{-r} 是 $p<1/2$ 时 P_{ud} 的上界这一结论并不成立。其中,p 是符号出错概率。实际上,这只有在编码采用了自然分组长度时才有效,对于缩短码则不成立。例如 16 阶本原多项式的循环长度是 $2^{16}-1$。如果分组长度是完整的,那么所有具有相同校验位的编码性能相同。如果受保护的分组长度小于该长度,如 512 字节,则称该多项式用于缩短循环码。这时,采用具有相同校验位的不同本原多项式后进行编码,其检错失效率差别很大。

表 1　常用 CRC16 和 CRC32[2-3]

Tab. 1　Common CRC16 and CRC32

CRC 名称	对应生成多项式
CRC16Q	9948
CRC16QS	897D
CRC16-IBM	8005
CRC16-CCITT	1021
CRC16-T10	8BB7
CRC16-DNP	3D65
CRC16-DECT	0589
CRC32-IEEE	04C11DB7
CRC32C	1EDC6F41

文献[9]使用硬件来研究 CRC 的检错失效问题,比较不同 CRC 多项式的检错性能,发现有的多项式的检错性能会显著降低。就是说,当 $p<1/2$,并且采用缩短循环码,那么 P_{ud} 可能比 2^{-r} 大几个数量级。在自然分组长度下,P_{ud} 作为 p 的函数,独立于本原多项式,但是在缩短分组长度时,情况却不同。采用缩短分组长度时,一个本原多项式可能比其他具有相同次数的本原多项式好得多,这种码字的特点是多项式中大约有一半的非 0 项。

还可以使用更高阶的 CRC 多项式提高检错能力,但是这也会导致更大的开销。一方面,需要增加报文中校验位数,从而增加了冗余信息的长度和节点间数据传输的时间,减少了网络传输的有效带宽;另一方面,也会增加硬件复杂度,导致性能降低。当需要更强的错误检测能力时,这就面临着一种选择,要么以损失有效带宽为代价提高错误检测能力,要么以错误检测能力不足为代价提高有效带宽。

为了提高链路检错能力,研究人员提出了旋转 CRC 机制[10],并且在 Intel 新型系统中得到了应用[11]。当使用 8 阶的生成多项式计算 CRC 时,旋转 CRC 机制可以提供 8 阶 CRC 的所有检错能力,以及检测所有长度小于 15 的突发错、大多数长度为 16 的突发错。另外,旋转 CRC 机制把剩余错误减少到 $1/2^{16}$。这类似于使用 16 阶生成多项式进行 CRC 计算所达到的检错效果,但是在报文中,仅仅使用 8 位的校验码,节省了额外的编码开销,提高了传输效率。

如果错误的发生概率比较小,使用 Intel 的旋转 CRC 方法是增加检错能力的一种方法,不但可以检测突发数据错误,而且不会增加每个报文的开销。

2 改进的旋转 CRC 设计(MR-CRC)

当直接应用 Intel 的设计时,我们发现有报文出错甚至丢失现象,通过分析,我们发现,Intel 在实现 8 位旋转 CRC 时有两种假定:第一是假定链路质量比较高,错误出现概率很低;第二是报文在所有通道上采用源同步并行传输,报文不会完全丢失。

而我们的设计环境与 Intel 假定的环境有所不同:

1)链路使用光电混合技术,存在光电转换,链路传输质量可能较低;

2)报文长度较大,Intel 使用的链路层报文只有 96 位,而我们的链路层报文可达 256 位。但是报文中只有 16 位可用于校验,并且报文中不包含序列号;

3)报文在链路上进行传输时可能被完全丢弃。

此时,Intel 的方法有两个缺点,一是 8 阶的 CRC 校验能力太弱,尤其是报文长度变大时;另外,如果丢弃中间的一个完整报文,接收方可能不会有任何觉察,就是说,如果不附加额外的信息指示,如序列号,接收方仅仅使用校验信息来检查报文时,一旦丢失报文且不能检测到的话,就会对处理逻辑甚至通信协议产生不良影响。

为了克服原始旋转 CRC 机制的这两个缺点,我们采用两种手段来改进其校验性能,降低其检错失效率。

首先采用两个 16 位的 CRC 生成多项式,替代原始的 8 位生成多项式,提高校验能力。因为在报文数据较长时,使用 8 阶的校验会限制检错能力,需要使用更高阶的 CRC 生成多项式。国际上有多种已经存在的 CRC16 标准和新研究成果,如表 1 所示。

当使用 16 阶的旋转 CRC 时,除了可以提供 16 阶 CRC 的所有检错能力,还可以检测所有长度小于 31 的突发错、大多数长度 32 的突发错。另外,旋转 CRC 机制把剩余错误减少到 $1/2^{32}$。这就接近 CRC32 的检错能力,但是在报文开销上却少使用 16 位,提高了有效传输带宽。

由于存在多种 16 阶 CRC,我们需要选择出那些比较合适的生成多项式。对 CRC 生成多项式的选择基于两个因素:

1)在缩短码中,检错失效率要低。研究人员已经发现,CRC16-IBM,CRC16-CCITT 的漏检概率都大于 2^{-r},而 CRC16Q,CRC16QS,CRC16-T10 的检错能力超出表 1 中其他 16 位 CRC[5,9]。

2)硬件复杂度要合理,不能成为限制系统的瓶颈。因为在高速数据通信中,通常使用硬件电路来实现 CRC 校验过程。不同的 CRC 实现所需门级数不同,级数越多,信号传播时间越长,从而限制了电路频率,降低了传输性能。因此需要根据其最终逻辑复杂度来选择。

对于表 1 中给出的各种 CRC16 编码,我们使用 FPGA 对这几种 CRC 使用的资源和实现后的结果进行了比较,如表 2 所示。我们主要关注的资源指标是 Slice(FPGA 组成单元)、LUT(FPGA 查找表结构)占用量以及综合频率。

表 2 CRC16 资源占用及性能比较

Tab. 2 Resource and performance comparison of CRC16

CRC	Slices	LUTs	频率(MHz)
CRC16-CCITT	154	293	402
CRC16-DECT	157	312	395
CRC16-DNP	147	277	319
CRC16-IBM	120	222	455
CRC16-T10	160	338	430
CRC16Q	169	334	417
CRC16QS	153	329	406

从表 2 中可以看出,各多项式无论在资源占

用还是可实现频率上都有着显著的差别,这对于工程实现有着重要影响。根据不同的实现目标,选择不同的多项式来进行性能和资源的折中。对于常用的 CRC16-IBM,其实现性能最高,而且资源使用最少,对于检错失效率不是很高的应用来说,是理想的 16 位 CRC 选择。我们考虑的是在不产生性能瓶颈的情况下,优先选择那些检错失效率低的多项式[5,9],其中 CRC16Q,CRC16QS 和 CRC16-T10 性能和资源利用率接近,都在我们的候选范围中。

其次,来分析在 Intel 的实现中报文丢失却被漏检的原因。

在 Intel 实现的旋转 CRC 机制中,使用两个不同的 8 阶的生成多项式,GA 和 GB。对于每个报文 P_i,使用两个生成多项式,利用传统的 CRC 计算方法产生两个不同的校验码 CSA_i 和 CSB_i。在旋转 CRC 中,P_i 上携带的校验码 CS_i 是 CSA_i 和 CSB_{i-1}的异或,定义 CSB 的初始值 CSB_0 为 0,因此 $CS_1 = CSA_1 \oplus CSB_0 = CSA_1$。如果在 P_i 时检测到错误,则可能意味着 P_{i-1} 就出现了错误,因此,在随后的重传请求中携带的报文序列号应该是 P_{i-1}对应的序号。发送方 CRC 生成机制如图 1 所示。接收方 CRC 检测机制如图 2 所示。

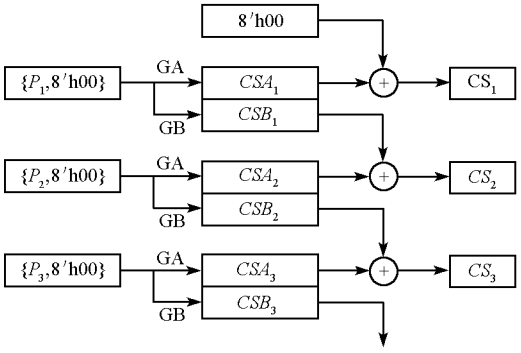

图 1　原始旋转 CRC 生成机制
Fig. 1　Original rolling CRC generator

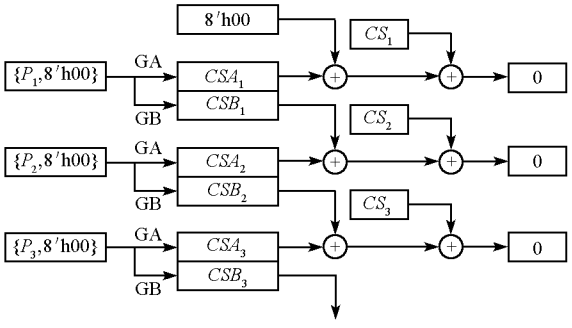

图 2　原始旋转 CRC 校验机制
Fig. 2　Original rolling CRC detection scheme

可以看出,CSA_i 和 CSB_i 的产生仅跟当前报文有关,CS_i 也仅与前一个报文和当前报文有关。如果连续报文中数据相同,其 CSA_i 和 CSB_i 以及 CS_i 的结果也将相同,中间丢失一个报文,不影响后面报文校验的正确性。也就是说,如果发送一连串数据完全相同的报文,中间丢失一个报文,接收方无法靠 CRC 检测到这种情况,此时检错失效率为 1。如果报文数据是完全随机的,接收方使用报文 P_i 对应的 CS_i 来检测 P_{i-1},检错失效率与 CS_i 的位数有关,在 8 阶的情况下,失效率为 $1/2^8$[1]。

我们对 CRC 计算过程中使用的填充数据进行修改,把计算校验时填充的 0 替换为上一个报文的最终校验,这样,每个报文携带的校验码信息不仅与上个报文的数据有关,而且与以前的发送历史有关,中间丢失报文后,将导致后继的报文校验出错。这样,接收方可以不依赖报文序列号等额外手段,仅仅使用校验来判断是否发生了报文丢失。

修改后的旋转 CRC 机制称为 MR-CRC (Modified Rolling CRC)。MR-CRC 中发送方 CRC 生成机制如图 3 所示。P_i 表示第 i 个报文中准备携带的实际数据,CS_i 是第 i 个报文最终携带的校验码。计算第一个报文的校验码时,取 $CS_0 = 0$,$CSB_0 = 0$,使用两个生成多项式分别计算得到 CSA_1 和 CSB_1,报文最后携带的 $CS_1 = CSA_1 \oplus CSB_0 = CSA_1$。计算其他报文的校验码时,首先使用当前报文数据 P_i 与上一个报文的校验码 CS_{i-1} 用来计算 CSA_i 和 CSB_i,而不是填充 0。可以看出,报文携带的校验码 CS_i 与历史发送的所有报文数据有关,而不仅仅取决于当前报文和上一个报文中的数据。

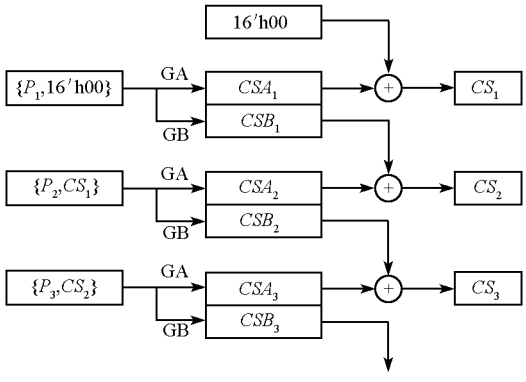

图 3　MR-CRC 生成机制
Fig. 3　MR-CRC generator

MR-CRC 的接收方 CRC 检测机制如图 4 所

示。P_i 表示第 i 个报文中携带的数据，CS_i 是第 i 个报文携带的实际校验码。计算第一个报文的校验码时，取 $CS_0=0$，$CSB_0=0$，使用两个生成多项式分别计算得到 CSA_1 和 CSB_1，使用报文携带的 $CS_1 \oplus CSA_1=0$ 来验证报文的正确性。验证其他报文的校验码时，首先使用当前报文数据 P_i 与上一个报文的校验码 CS_{i-1} 用来计算 CSA_i 和 CSB_i，然后用 $CSA_i \oplus CSB_{i-1} \oplus CS_i=0$ 来验证报文是否正确。

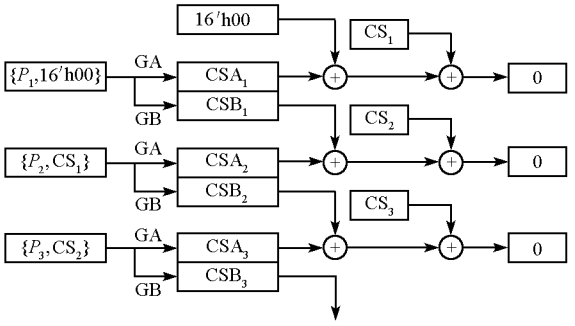

图 4 MR-CRC 校验机制

Fig. 4 MR-CRC detection scheme

可以看出，在 MR-CRC 中，即使报文数据完全相同，生成的校验码也可能不同，因此接收方的检错失效率与报文数据没有明显的关系。如果报文 P_i 与 P_{i-1} 所含数据相同，在 P_i 丢失后，原始旋转 CRC 无法检测到错误。而 MR-CRC 有极大的概率检测到这种报文丢失，报文 P_i 的丢失将直接影响到 P_{i+1} 和 P_{i+2} 的校验，检错失效的情况出现在正确接收到了 P_{i+1} 与 P_{i+2}，在 16 阶的情况下，这种概率只有 $1/2^{16} \times 1/2^{16} = 1/2^{32}$。

3 实 现

实际 CRC 硬件设计中，CRC 的计算可以有两种方式：一种是串行方式，另一种是并行方式[12]。串行方式就是按位产生 CRC，也就是通过一个线性反馈移位寄存器按每次输入一位的方式来产生。这种方式结构规则，但每个时钟周期只能计算 1 位的结果，速度较慢。为了提高处理的速度和效率，上述 CRC 的产生和检查都是可以在传输的过程中通过硬件按并行方式进行计算的。并行方式是每次输入一组并行数据，同时产生出 CRC 结果。并行方式处理速度较快，更适合做高速协议的处理。

我们针对 256 位的报文组织进行了实现，其中数据宽度 240 位，CRC 校验码 16 位，CRC 计算采用上述 MR-CRC 机制。

MR-CRC 的逻辑图如图 5 所示，主要模块的作用描述如下：

1）gen_cs 根据前一报文的 csb 和当前数据生成当前报文的 cs；

2）gen_csa 生成当前报文的 csa；

3）gen_csb 生成当前报文的 csb；

4）gen_precs 得到前一报文的 cs；

5）gen_precsb 得到前一报文的 csb。

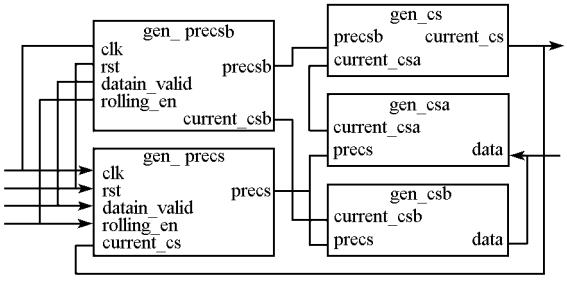

图 5 MR-CRC 的逻辑图

Fig. 5 MR-CRC implementation blocks

我们采用 Xilinx ISE 平台，选用器件 Xilinx FPGA Virtex5 XC5VLX30 FF676 – 1，首先对 3 种 16 位 MR-CRC 的不同组合进行了综合实现，结果如表 3 所示，MR-CRC 使用两个生成多项式。我们发现不同的组合资源利用率差别不大，但是其实现频率差别很大，CRC16QS&CRC16-T10 的组合具有最高的性能和较少的资源利用率。我们对 16 位 MR-CRC、CRC32 和 CRC16 的综合后可达到的最大频率和资源占用进行了比较，发现 MR-CRC 无论在资源还是综合频率上都是处于 CRC32 和 CRC16 之间，16 位旋转 CRC 无论在运行频率还是资源占用上比 CRC32 都有明显的优势。比较 CRC16QS&CRC16-T10 与 CRC32-IEEE 可以发现，在使用资源减少 10% 的情况下，频率提高了 25%。

表 3 MR-CRC 和 CRC32 性能指标

Tab. 3 Resource and performance comparison of MR-CRC and CRC32

CRC	Slices	LUTs	频率(MHz)
MR-CRC（CRC16QS&CRC16-T10）	284	653	370
MR-CRC （CRC16Q&CRC16-T10）	277	664	312
MR-CRC(16Q&16QS)	281	658	330
CRC32C	330	707	299
CRC32-IEEE	316	704	296

另外，我们比较了 MR-CRC 与 CRC32 使用的异或逻辑运算的最大值，这可以反映出它们的逻

辑复杂度。MR-CRC 使用的 CRC16-T10、CRC16QS、CRC16Q 的最长异或操作表达式分别为 139、140、138,比较接近。CRC32C 和 CRC32-IEEE 中的最大长度分别是 153 和 147,在逻辑复杂度上要高于前面的 MR-CRC。

从表 2 和表 3,我们可以得到下列结果:

1)CRC16 使用的资源少于 CRC32,无论是 Slices 还是 LUTs;

2)CRC16 频率高于 CRC32;

3)旋转 CRC16 使用的 Slices 和 LUTs 介于 CRC16 和 CRC32 之间;

4)旋转 CRC16 的频率要高于某些 CRC16;

5)旋转 CRC16 的频率明显高于 CRC32。

我们使用 Verilog 实现了上述校验过程,并且进行了模拟。发送端模拟对比结果如图 6 所示,(a)为原始旋转 CRC 波形,输入数据 0ff(txp_lit)固定不变,生成的校验(current_cs)也不变化;(b)为 MR-CRC 波形,输入数据 0ff 固定不变,但是生成的校验(current_cs)不断变化。这是因为 precs 保留了上一个报文的校验码,使用它来参与本报文的校验码生成。

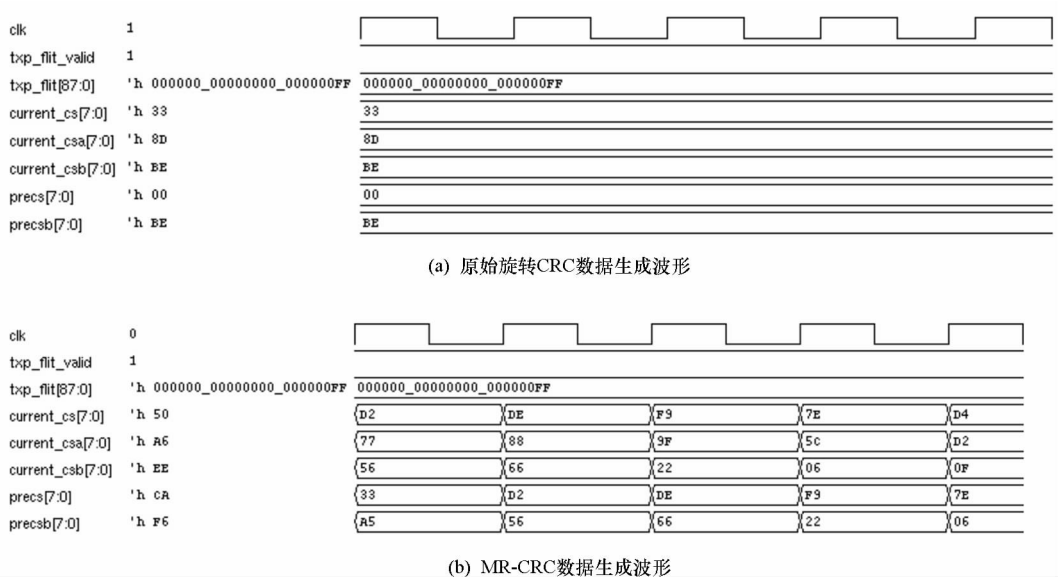

图 6　原始旋转 CRC 与 MR-CRC 发送端生成校验比较

Fig. 6　Generator diagram comparison of original rolling CRC and MR-CRC

接收端模拟对比结果如图 7 所示,其中 rxp_flit_valid 为低电平时表示此处丢失了某个报文,rolling_crc_noerror 表示是否检测到报文的丢失。(a)为原始旋转 CRC 波形,丢失报文对于信号 rolling_crc_noerror 没有变化,即没有检测到报文的丢失。(b)为 MR-CRC 波形,rxp_flit_valid 为低电平后,rolling_crc_noerror 随之变低,表示所收数据发生了校验错,这就达到了检测丢失报文的目的。

4　结 论

本文提出了一种新型的 CRC 校验方法 MR-CRC,它同时使用两个生成多项式来生成校验码,在不消耗额外带宽的情况下提高了数据传输过程中的错误检测能力。通过 FPGA 实现,我们发现具有相似校验能力的不同多项式无论在逻辑复杂度、所占用的芯片面积还是性能上都有显著的差别,需要根据实际情况进行资源、性能上的折中,这对于工程实现具有重要的借鉴意义。当使用 16 阶的 MR-CRC 时,性能非常接近使用的 CRC16,但是 MR-CRC 机制除了可以提供 16 阶 CRC 的所有检错能力,还可以检测所有长度小于 31 的突发错,大多数长度 32 的突发错,非常类似 CRC32。相比 CRC32,16 阶 MR-CRC 具有较低的逻辑复杂度,在所用资源减少 10% 的情况下,频率提高了 25%,可以取代 CRC16 用于改善错误检测能力。

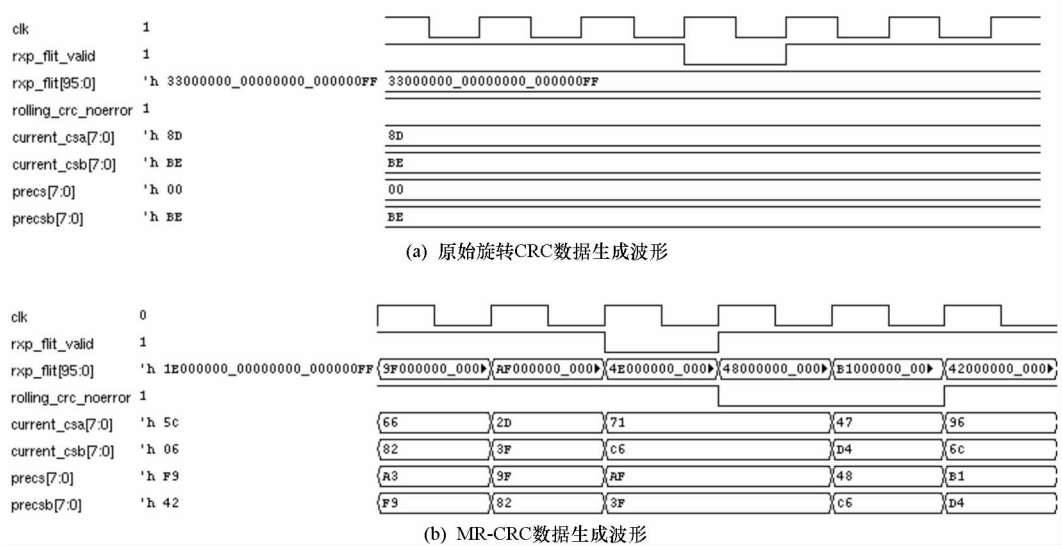

图 7 原始旋转 CRC 与 MR-CRC 接收端校验比较

Fig. 7 Detector diagram comparison of original rolling CRC and MR-CRC

参 考 文 献:

[1] 王新梅,肖国镇. 纠错码原理与方法[M]. 西安:西安电子科技大学出版社, 2001.

[2] Wolf J K, Blackeney R D. An Exact Evaluation of the Probability of Undetected Error for Certain Shortened Binary CRC Codes [C]//Proceeding MILCOM, 1988: 1521 – 1526.

[3] Chun D, Keil J. Special Hardware for Computing the Probability of Undetected Error for Certain Binary CRC Codes and Test Results [J]. IEEE Transactions on Communications, 1994, 42(10): 2769 – 2772.

[4] Sharma D D. Rolling CRC Scheme for Improved Error Detection [DB/OL]. [2011 – 04 – 29]. http://www.freepatensonline.com/y2004/0098655.html.

[5] Maddox R A, Singh G, Safranek R J. Weaving High Performance Multiprocessor Fabric [M]. California: Intel Press, 2009.

[6] Lange C, Ahrens A. On the Undetected Error Probability for Shortened Hamming Codes on Channels with Memory [C]// Proceedings of IMA Int. Conf., 2001: 9 – 19.

[7] Fujiwara T, Kasami T, Kitai A, et al. On the Undetected Error Probability for Shortened Hamming Codes [J]. IEEE Transactions on Communications, 1985, 33: 570 – 574.

[8] Chen W, Lin R. Fast Calculation Algorithm of the Undetected Errors Probability of CRC Codes [C]//The First International Workshop on Information Networking and Applications, 2005: 480 – 483.

[9] 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, 17(2): 56 – 62.

[10] Baicheva T, Dodunekov S, Kazakov P. Undetected Error Probability Performance of Cyclic Redundancy-Check Codes of 16-bit Redundancy [J]. IEEE Proceedings on Communications, 2000, 147:253 – 256.

[11] Koopman P, Tridib C. Cyclic Redundancy Code (CRC) Polynomial Selection for Embedded Networks [DB/OL]. [2011 – 04 – 29]. http://www.ece.cmu.edu/~koopman/roses/dsn04/koopman04_crc_poly_embedded.pdf.

[12] Greg C. Catalogue of Parameterized CRC Algorithms [DB/OL]. [2011 – 04 – 29]. http://regregex.bbcmicro.net/crc-catalogue.htm.