文章编号:2095-476X(2012)06-0065-04

USB 3.0设备中并行循环冗余校验的研究与实现

滕立伟[1], 李小花[2]

(1. 中国海洋大学 信息科学与工程学院, 山东 青岛 266100;
2. 昆明理工大学 信息工程与自动化学院, 云南 昆明 650500)

摘要:针对串行循环冗余校验(CRC)算法不适于高速传输且不易于硬件实现的问题,结合USB 3.0设备中CRC的特点,推导出一种并行CRC算法,并用Verilog硬件编程语言加以实现. 仿真试验表明,并行CRC校验算法具有更高的数据吞吐率,能降低时钟频率,易于硬件实现.

关键词:USB 3.0设备;循环冗余校验;并行算法
中图分类号:TN919 **文献标志码**:A DOI:10.3969/j.issn.2095-476X.2012.06.018

Research and implementation of the parallel CRC in USB 3.0 device

TENG Li-wei[1], LI Xiao-hua[2]

(1. *College of Information Science and Engineering, Ocean University of China, Qingdao* 266100, *China*;
2. *Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming* 650500, *China*)

Abstract: Aiming at the problem that the serial CRC check algorithm is not suitable for high-speed transmission and not easy to hardware realization, a parallel CRC algorithm was deduced base on the characteristics of the CRC in the USB device, and was implemented with Verilog hardware programming language. The simulation results showed that parallel CRC algorithm has a higher data throughput rate and can reduce the clock frequency. It's easier to implement in hardware.

Key words: USB 3.0 device; cyclic redundancy check(CRC); parallel algorithm

0 引言

为了提高数据传输的有效性,通常在发送数据时对数据进行编码,并加入校验位来检测接收数据是否正确. 常用的校验码有奇偶校验码、海明码和循环冗余校验CRC(cyclic redundancy check)码. 这些编码方式都是按照一定的编码规则,在信息位后增加冗余位后一起发送. 接收端在接收到信息位后,按照相同的编码规则得到校验位,与接收的校验位比较即可得知信息位是否正确. 其中,CRC是USB 3.0协议采用的数据校验方式. CRC码不仅具有很强的检测能力,而且实现简单,在数据传输中被广泛应用. 由于串行CRC算法本身的局限性,以提高时钟频率为代价来提高数据吞吐率已不能满足高速传输的要求,且不易于硬件实现,因此需要采用更快的并行算法. 本文在CRC原理的基础上,针对USB 3.0设备的具体应用进行研究,由串行CRC算法推导出一种并行CRC算法,以提高数据吞

收稿日期:2012-07-17

作者简介:滕立伟(1987—),男,山东省烟台市人,中国海洋大学硕士研究生,主要研究方向为多媒体通信网络与系统设计.

吐率,降低时钟频率.

1 USB 中的 CRC 算法

1.1 CRC 原理

CRC 码是一种截短循环码,属于线性分组码. 其编码的过程为:当发送一个 k 位的信息序列 $T(x)$ 时,根据 k 可以使用一个特定的 $r+1$ 位生成多项式 $G(x)$,再将信息序列 $T(x)$ 左移 r 位,低位补零,用 $T(x)x^r$ 除以生成多项式 $G(x)$ 得到余数 $R(x)$ 和商 $Q(x)$,其中 $R(x)$ 即是校验位. 将 $R(x)$ 附加在信息位后一并发送. 接收端接收到数据(包括校验位)后,用同一个生成多项式 $G(x)$ 去除,若余数为零则可判断所接收到的信息位正确;否则,表明传输过程中发生了错误.

由上述过程可知

$$T(x)x^r = Q(x)G(x) + R(x) \qquad ①$$

余数 $R(x)$ 作为校验位附加在信息位 $T(x)$ 后. 此时实际发送的数据

$$T(x) = T(x)x^r + R(x) \qquad ②$$

由式①②可得

$$T(x) = Q(x)G(x) + R(x) + R(x) \qquad ③$$

由于采用的是模 2 运算,所以接收端校验时相当于用 $Q(x)G(x)$ 除以生成多项式 $G(x)$. 如果信息序列在发送、传输和接收过程中没有发生错误,校验时余数为 0.

移位算法是常用的 CRC 编码算法,其编码电路是一种串行结构的除法器[1],图 1 为 CRC-16 的硬件电路框图.

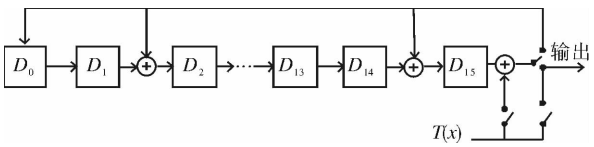

图 1 CRC-16 硬件电路实现框图

CRC 编码电路工作过程如下:

1)在 $T(x)$ 输入时就可以进行除法运算,同时,将信息位 $T(x)$ 送到输出,形成编码的前半段;

2)所有的信息序列 $T(x)$ 输入完后,移位寄存器中的数据即是校验位,将校验位紧接着信息序列输入即可完成编码.

1.2 USB 3.0 的包结构

USB 3.0 协议中有超高速、高速、全速和低速 4 种传输模式. 高速、全速和低速模式传输的数据包有 Token 令牌包、Data 数据包和握手包 3 种[2]. 令牌包分为 IN 令牌、OUT 令牌、SETUP 令牌和 SOF 令牌,其中 SOF 令牌包与前三者的包结构略有不同,但数据位数相同. 各数据包的结构如图 2 所示.

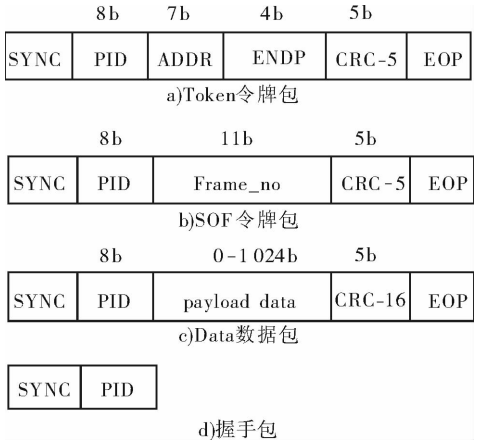

图 2 高速模式中的 4 种包结构

CRC 码用来在令牌包和数据包中保护所有的非 PID 字段. PID 字段存在自身的校验方式,故其不在 CRC 的校验范围内. 其中,令牌包采用 CRC-5 对 ADDR,ENDP 或者 Frame_no 字段提供 11 b 的数据校验,其生成多项式为 $G(x) = x^5 + x^2 + 1$;Data 数据包采用 CRC-16 对有效数据位提供校验,其生成多项式为 $G(x) = x^{16} + x^{15} + x^2 + 1$;握手包用来报告数据事务的状态,能表示数据成功接收、命令的接收或拒绝等,仅由一个 8 b 的 PID 构成,不需要校验.

超高速(SS)模式规定了 4 类型的包:Link Management Packet(LMP),Transaction Packet(TP),Data Packet(DP)和 Isochronous Timestamp Packet(ITP)[3]. 所有的包都有一个 16 B 包头,其中 TP,LMP 和 ITP 仅由 1 个包头构成;DP 由 Data Packet Header(DPH)和 Data Packet Payload(DPP)组成,其中 DPH 为 16 B 的包头,DPP 是 0—1 024 B有效数据. 图 3 给出了 DP 的结构图,其他 3 种类型的包结构与 DPH 的结构类似.

在 DPH 中包含 12 B 头信息、2 B CRC-16 校验位和 2 B 链路控制字. 其中 2 B 的 CRC-16 校验位用于保护 12 B 头信息. 链路控制字中包含 7 b 的信息位和 5 b 的 CRC-5,CRC-5 校验位用于保护 7 b 的信息位.

在 DPP 中包含 0—1 024 B 的数据位和 4 B 的 CRC-32 校验位,其生成多项式为

$$G(x) = x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

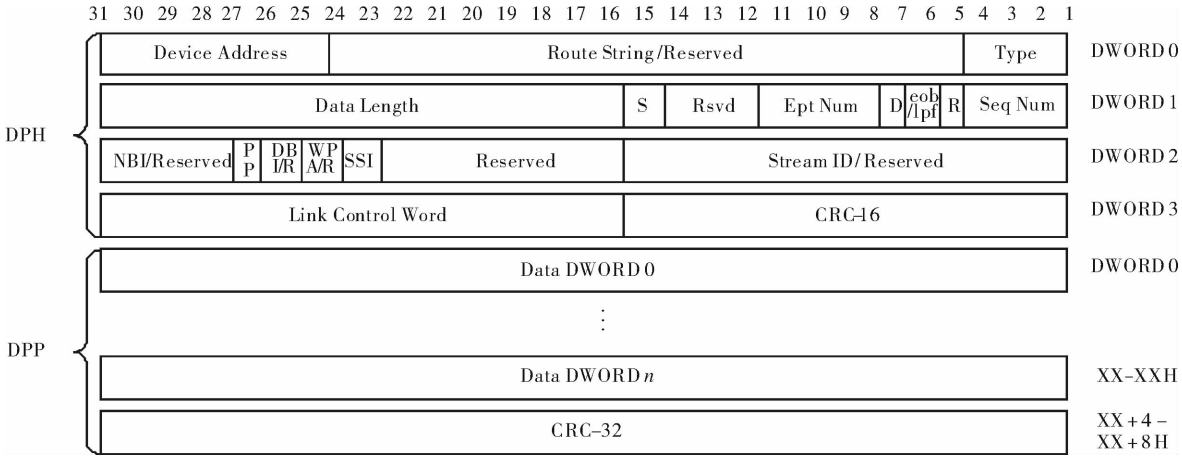

图 3　DP 结构图

2　并行 CRC 在 USB 3.0 中的实现

2.1　并行 CRC 算法的推导

图 1 所示的 CRC-16 硬件电路可以实现 USB 3.0数据传输中的 CRC 算法,但速度较慢. 在高速模式下传输速度为 480 Mb/s,超高速模式下传输速度高达 5 Gb/s,难以用硬件电路实现串行 CRC 算法. 故本文拟将串行算法改进为并行 CRC 算法. 下面以串行 CRC-16 算法推导并行 CRC 算法. 由图 1 所示电路可以得到在输入 1 位数据 din 后寄存器 crc15 和 crc14 的计算式,即

$$crc15' = crc15 \oplus crc14 \oplus din \quad ④$$

$$crc14' = crc13 \quad ⑤$$

当输入第 2 位数据后,用输入数据、crc14′和 crc15′计算得到新的 crc15. 连续输入 8 位数据,寄存器的结果根据式④⑤重复 8 次运算,最终得到串行输入 8 位数据后的 crc15.

其计算式为

crc15 = din[7]^din[6]^din[5]^din[4]^din[3]^
din[2]^din[1]^din[0]^crc7^crc8^crc_in9^crc10^
crc11^crc12^crc13^crc14^crc15

按照上述方法可以推导出其他位寄存器在串行输入 8 位数据后的计算式. 为了提高处理速度,设计中直接采用该计算式对并行输入的 8 位数据进行计算,得到 16 位的校验结果,再将新的校验结果存入寄存器,与下一次输入的并行数据进行计算,得到新的校验结果. 重复此操作,直到接收数据全部校验结束. 这样可实现并行的 CRC 校验,即在一个时钟周期内完成并行 8 位数据的校验.

2.2　并行 CRC 算法的实现

为了保护数据包开头的 0 数据位,初始时需将 CRC 生成器移位寄存器全部置 1,相当于在数据前添加一组固定的数据位[4]. 如果初始值为 0,移位寄存器内的数据将与输入数据做异或运算,数据包开头的 0 与寄存器内的 0 异或,移位寄存器中的数据仍为 0,导致数据 0 得不到保护. 同理,在接收端也以同样的方式设置移位寄存器的初始值,这样可以消除移位寄存器置 1 对 CRC 校验的影响.

在数据位校验完后,移位寄存器中的校验位要按位取反,然后高位在前进行发送,这样可以保护数据包末尾的 0 数据位. 另外,USB 设备中 CRC 采用并行方式(并行 CRC 方式将在下文介绍).

由于发送端对移位寄存器初始值置 1 和对校验位取反,在接收端有 2 种校验方式,2 种校验方式移位寄存器初始值与发送端相同,均置 1. 不同之处在于接收到的校验位是否进入校验器进行校验. 一种校验方式是与发送端的处理过程相同,仅对数据位做校验,不对校验位进行校验,将得到的校验结果取反后与接收到的校验位比较. 如果相同说明接收成功,否则说明数据位发生错误. 然而这种方式仅限于数据位数确定的情况,因为数据位确定时才可以提取出接收到的校验位. Token 令牌包中需校验的数据长度固定为 11 b,容易提取出校验位,然而对于 Data 数据包,其有效数据长度为 0—1 024 B,难以从 Data 数据包中取出校验位. 因此,此方法仅适用于 CRC-5 校验 Token 令牌包.

另一种校验方式是将接收到的数据位连同校验位一起送入校验器进行校验. 由于对校验位取反,相当于添加了一组固定的数据,这样在接收端校验的结果总是一个恒定值[1]. 对于接收的 CRC-32 数据包,这个恒定值是 32′hc704dd7b;对于接收的 CRC-16 数

据包,这个恒定值是16′h800d;对于 CRC-5 数据包,校验结果是5′h06. 接收端只需要判断校验后的结果是否与这个恒定值相同,即可判断接收数据是否正确. 设计中 CRC-16,CRC-32 采用此方法.

表1 是 CRC-16 校验器的端口说明. 设计中 crc_out[15:0]为 16 个校验寄存器,crc_in[15:0]是校验寄存器 crc1—crc15 的输入, crc_in[15:0]的值等于 crc_out[15:0],其初始值为16′hffff,din[7:0]是输入的 8 位待校验的数据. 根据并行 CRC 算法,此处给出 crc_out[0]的计算表达式为

$$crc\_out[0] = din[7] \wedge din[6] \wedge din[5] \wedge din[4] \wedge din[3] \wedge din[2] \wedge din[1] \wedge din[0] \wedge crc8 \wedge crc9 \wedge crc10 \wedge crc11 \wedge crc12 \wedge crc13 \wedge crc14 \wedge crc15$$

当前计算得到的 crc_out 送给 crc_in 作为下一次计算的寄存器值. 在发送端 CRC-16 校验时,需要将 crc_out 取反后发送;在接收端 CRC-16 校验时,不需要将 crc_out 取反,只需将其与 16′h800d 进行比较即可. crc_out[0]的硬件电路如图 4 所示,其他寄存器输出与 crc_out[0]类似.

表 1 CRC-16 校验器端口说明

端口	位宽/b	输入/输出	说明
crc_in	16	Input	校验寄存器
din	8	Input	输入数据
crc_out	16	Output	校验寄存器输出

CRC-32 的校验方式与 CRC-16 的校验方式相同. CRC-5 的设计与 CRC-16 类似,CRC-5 采用 11 b 并行输入,对接收的 Token 令牌包只需要一个时钟周期就可以得到校验结果;然后,将此校验结果取反与发送的校验结果相比较来判断数据是否正确.

3 仿真结果

本设计采用 Verilog 硬件编程语言编写,并使用 modlesim 仿真软件进行仿真. 图 5 是对 CRC-16 接收校验的仿真结果. 仿真过程中 USB 设备接收 DATA 包数据. 接收模块接收到 8 位串行数据后,将串行数据转换成8 位并行数据送给 CRC-16 校验器进行校验. 在图中竖线右侧 din = 39 为接收到的最后的 8 位数据. 在时钟的上升沿,将前一拍的输出 crc_out 传递给 crc_in 作为当拍的输入. crc_in 与并行输入数据 din(39)计算得到校验结果 crc_out 位800d. 表明数据接收正确.

将设计使用 Quartus Ⅱ进行综合,将得到门级网表下载到 ALTERA 公司的 Cyclone Ⅱ系列的 FPGA 进行验证,得到的结果如下:串行算法最大工作频率 180 MHz,吞吐率峰值 210 Mb/s,使用的 slice 为 60;并行算法最大工作频率 130 MHz,吞吐率峰值 600 Mb/s,使用的 slice 为 128.

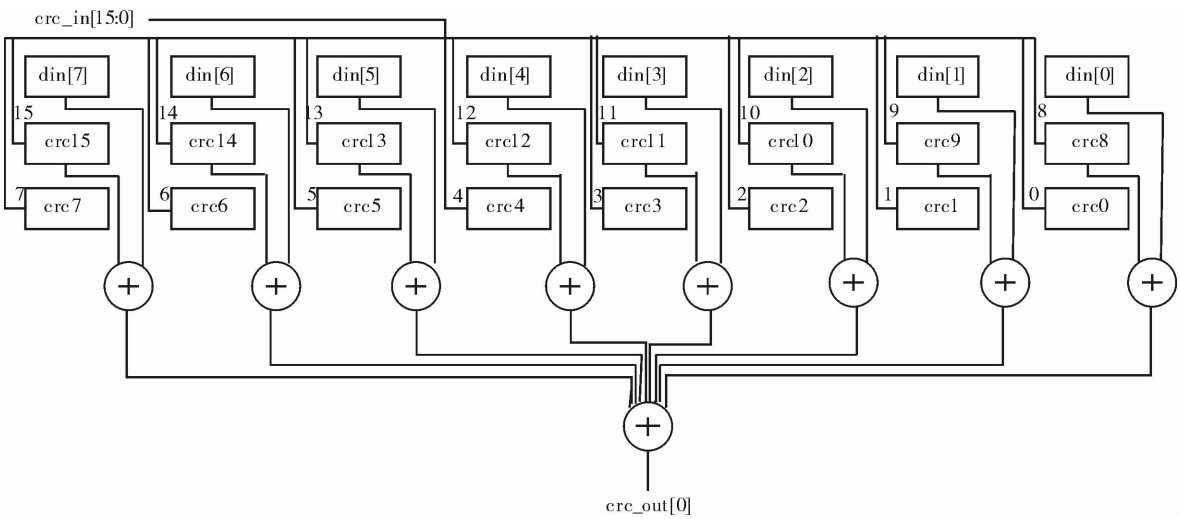

图 4 并行 crc_out[0]硬件电路

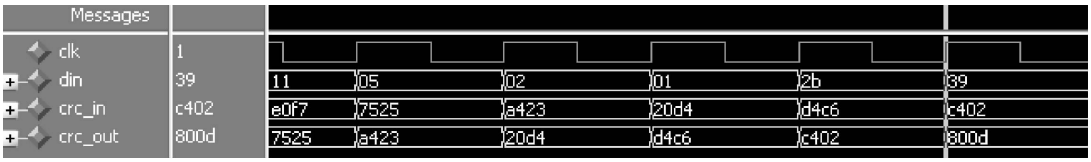

图 5 CRC-16 接收校验仿真结果

(下转第 72 页)