文章编号:1671-4598(2026)02-0167-07 DOI:10.16526/j.cnki.11-4762/tp.2026.02.022 中图分类号:TP212 文献标识码:A

基于 RS422 接口的 FPGA 在线升级优化设计

焦新泉，杨景灵
（中北大学 电子测试技术国家重点实验室，太原 030051）

摘要：随着技术的快速发展，FPGA 因其灵活性和高性能在多个领域得到广泛应用；面对不断变化的应用需求，FPGA 的在线升级技术成为关键挑战之一；采用了一种基于 RS422 接口的 FPGA 在线升级优化设计，充分利用了 RS422 的长距离传输和抗干扰优势，实现了上位机与 FPGA 开发板之间的稳定通信；通过 RS422 通信模块接收控制指令和配置数据，FPGA 控制模块负责解析指令并对 FLASH 执行擦除、写入等操作，确保固件的顺利更新；利用 ICAP 模块和 Multiboot 功能，设计实现了动态重配置，即使在升级过程中遇到意外情况也能通过 Fallback 模式恢复；为了进一步提升数据传输的可靠性，引入了 RS 编码与 CRC 校验相结合的技术方案；两者结合不仅提高了数据传输的准确性，还增强了系统的纠错能力，有效应对突发错误和噪声干扰，保证了 FPGA 在线升级过程的安全性和稳定性。

关键词：在线升级；RS422 接口；ICAP 模块；Multiboot；Fallback 模式；RS 编码

Optimization Design of FPGA Online Upgrade Based on RS422 Interface

JIAO Xinquan, YANG Jingling
(National Key Laboratory for Electronic Measurement Technology, North University of China, Taiyuan 030051, China)

Abstract: With the rapid development of technology, FPGA has been widely used in multiple fields due to its flexibility and high performance. Faced with constantly changing application requirements, the online upgrade technology of FPGA has become one of key challenges. An FPGA online upgrade optimization design based on RS422 interface is presented, which fully utilizes the long-distance transmission and anti-interference advantages of RS422, achieving stable communication between the upper computer and FPGA development board. The control instructions and configuration data are received through the RS422 communication module, and the FPGA control module is responsible for parsing the instructions and performing operations such as erasing and writing to the FLASH to effectively update the firmware. An ICAP module and Multiboot function are used to design and implement dynamic reconfiguration, which can be restored through Fallback mode even in the event of unexpected situations during the upgrade process. In order to further improve the reliability of data transmission, a technical solution combining reed-solomon (RS) encoding and cyclic redundancy check (CRC) check is introduced, which not only improves the accuracy of data transmission, but also enhances the system's error correction capability, effectively responding to sudden errors and noise interference, ensuring the safety and stability of FPGA online upgrade process.

Keywords: online upgrade; RS422 Interface; ICAP module; Multiboot; Fallback mode; RS code

0 引言

在数字化浪潮的推动下，技术的迭代与升级成为推动各行各业发展的不竭动力。在众多前沿技术中，FPGA 作为一种高度灵活的可编程逻辑设备，在数字信号处理，5G 通信网络，汽车电子与航天等领域内都有着广泛的应用。其芯片灵活性高、快速并行运算、低批量运算、可重复编程等的特点对于实现复杂算法和系统级优化至关重要。然而，随着应用需求的不断变化和系统复杂度的日益提升，如何快速、安全地实现 FPGA 在线升级，以适应新的功能需求或修复潜在的问题，成为亟待解决的关键挑战。

在产品调试的过程中，一般使用 JATG 接口进行升级，而在产品交付之后，如果依然按照这种方法，就需要将产品的外包装进行拆解，极大消耗人力物力，这种升级方式变得极为不便且成本高昂。为了解决这一问题，通常会采用更为灵活的在线升级方法，例如串口升级和网口升级。其中，网口升级虽然能够实现远程更新，但其实现相对复杂，不仅需要专门的物理层（PHY）芯片来支持网络通信，还会占用 FPGA 的部分

收稿日期:2025-01-25； **修回日期**:2025-03-04。
作者简介:焦新泉(1978-),男,博士,教授。
引用格式:焦新泉,杨景灵.基于 RS422 接口的 FPGA 在线升级优化设计[J].计算机测量与控制,2026,34(2):167-173,181.

逻辑资源，这对于资源有限的设计来说可能不是一个理想的选择[1]。相比之下，串口升级成为一种更为实用的方法。RS422 是一种差分信号传输的标准，适用于较长距离的数据传输，具有较强的抗干扰能力。通过 RS422 接口，上位机可以与 FPGA 开发板建立稳定的通信链路，进而实现对 FLASH 的操作，包括读取、写入以及擦除等动作，从而完成固件的在线更新。

尽管现有的 FPGA 在线升级技术已经取得了显著的进步，但仍面临一些挑战。首先，数据完整性与准确性保障不足，尤其是在远距离传输或者高噪声环境下，传统的校验机制如 CRC 难以满足高效纠错的需求，可能导致数据丢失或错误，影响升级成功率。其次，某些高级升级方法往往伴随着较高的实施复杂度，不仅需要专业的技术支持，而且可能涉及不同厂商之间的兼容性问题，增加了集成难度。针对以上提到的问题，提出了一种基于 RS422 接口的 FPGA 在线升级优化设计方案。该方案结合了 CRC 校验与 Reed-Solomon（RS）编码的优势，旨在提高数据传输的完整性和准确性；同时，通过优化升级流程，减少对 FPGA 逻辑资源的占用，降低系统的复杂度和成本。

1 总体方案设计

系统设计由 RS422 通信模块和 FPGA 控制模块组成。总体设计方案如图 1 所示，RS422 通信模块主要负责接收来自上位机的命令和配置数据，并将这些数据传输至 FPGA 控制模块。该模块利用 RS422 差分信号传输标准，以强大的抗干扰能力和远距离传输能力，特别适合于远程数据传输场景[2]。上位机通过 RS422 接口发送控制指令和数据包，RS422 通信模块接收这些指令和数据包，进行初步处理后，将其封装成适合 FPGA 控制模块处理的格式，并通过 RS422 接口传输至 FPGA 控制模块。同时，RS422 通信模块还接收上位机的控制信号，如开始传输、停止传输等，根据这些控制信号控制数据的发送和接收过程。

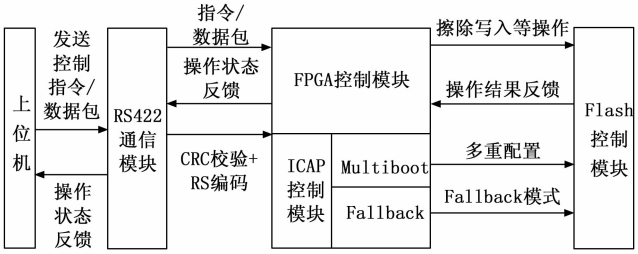

图 1 总体设计方案

FPGA 控制模块作为系统的核心处理单元，负责对 RS422 通信模块传来的数据进行解析和执行。当该模块接收到指令和配置文件后，它会解析这些信息并确定所需执行的具体操作，包括对 FLASH 存储的擦除和写入等。FPGA 控制模块通过其内部的 ICAP（在系统可编程）模块来实现对 FLASH 的精确控制。ICAP 模块的功能是在不影响 FPGA 当前运行状态的前提下，允许对其进行重新配置，这包括实现 FPGA 的重配置以及配置 Fallback 模式，以确保系统的稳定性和可靠性。

为了确保数据传输的准确性和完整性，系统引入了 CRC 校验和 RS 编码相结合的技术。在数据发送前，RS422 通信模块对数据进行 CRC 校验和 RS 编码，以增加冗余信息。FPGA 控制模块在接收到数据后，首先进行 CRC 校验，检查数据的完整性。如果 CRC 校验通过，FPGA 控制模块进一步使用 RS 编码进行错误检测和纠正，确保数据的准确性。这种结合方式不仅提高了数据传输的可靠性，还能够在一定程度上纠正传输过程中可能出现的错误，从而确保数据的完整性和正确性。通过这种设计，系统不仅提高了数据传输的可靠性和准确性，还为 FPGA 的在线升级提供了一种高效、可靠的解决方案。

2 硬件电路设计

2.1 RS422 通信模块

RS422 模块通过差分信号传输技术提供了高抗干扰能力和长距离通信能力。为了确保最佳性能和可靠性，硬件布线和电气特性参数的配置至关重要。RS422 标准使用两对差分信号线进行数据传输：一对用于发送（TX＋和 TX-），另一对用于接收（RX＋ 和 RX-）[3]。此外，还需要一对共用地线（GND）以确保信号参考电平的一致性。为了防止信号反射和回波损耗，通常会在通信链路的两端配置 120 Ω 的终端电阻，这些终端电阻有助于吸收信号能量，减少信号反射，从而保持信号完整性。RS422 模块需要配置适当的电气特性参数，包括信号电平、最大传输速率以及驱动能力等。根据 RS422 标准，信号电平范围为±2～±6 V，这有助于在存在噪声的环境中保持信号的清晰度[4]。

此外，为了进一步增强抗噪性能，可以采取隔离措施，例如使用光电耦合器或隔离变压器来隔断电源和地线间的直接连接。设计采用 ADM2582E 芯片自带隔离电路，RS422 接口电路如图 2 所示。ADM2582EBRWZ 芯片通过其电源引脚接入工作电压，并通过多个接地引脚实现稳定的参考电位。芯片内部集成的驱动使能（DE）和接收使能（RE）引脚分别控制数据的发送和接收过程，确保了数据传输的正确性和可靠性。当 DE 引脚置为高电平时，芯片配置为发送模式；当 RE 引脚置为低电平时，芯片则进入接收模式。差分信号通过引脚 A 和 B 发送，通过引脚 Y 和 Z 接收。为了确保芯片的稳定运行，电路中还包含了去耦电容，这些电容连接在电源和地之间，用于滤除可能影响芯片性能的电源噪声。

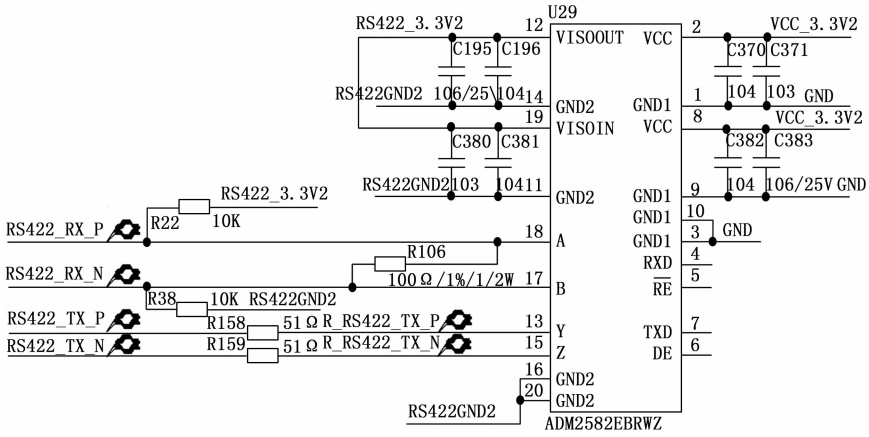

图 2　RS422 接口电路

2.2　FPGA 配置电路

设计采用 Xilinx Kintex-7 系列芯片作为核心处理单元，并选择 SPI 接口来实现 FPGA 与 FLASH 之间的数据交互。SPI 接口采用全双工串行通信模式，允许数据间的同时双向传输，提高了数据交换的效率。为了确保高效且可靠的数据传输，将 FPGA 设定为主串模式[5]。这种配置模式仅需外挂一个 SPI Flash 存储器，减少了系统的复杂性与成本。FPGA 配置电路如图 3 所示。

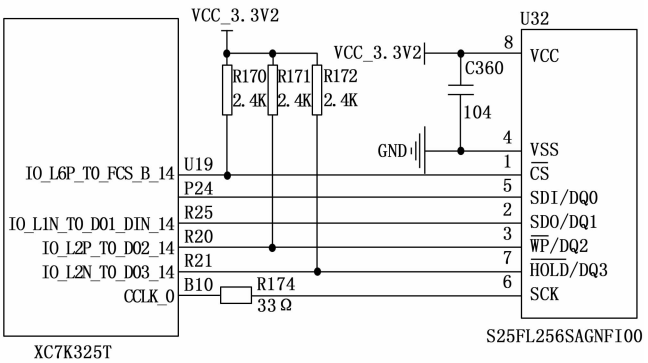

图 3　FPGA 配置电路

3　软件逻辑设计

3.1　FLASH 控制模块逻辑设计

在嵌入式系统的在线固件升级过程中，Flash 控制模块扮演了核心角色，其主要职责在于管理 FPGA 与 Flash 存储器之间的交互，涵盖设备识别、写入使能、扇区擦除、编程以及数据读取等多项操作[6]。

状态控制模块是 Flash 控制模块的重要组成部分，它不仅负责解析和分发来自主控计算机的指令，还确保在每个命令执行完毕后能够清除当前状态并准备好接收下一条命令[7]。此外，状态控制模块包含状态寄存器读取逻辑，该逻辑用于在每一步操作前后验证 Flash 的状态，确保其符合预期。设计采用 S25FL256S Flash 芯片，存储结构与地址映射表如表 1 所示。这一模块通过地址和数据总线控制器向 Flash 芯片发送精确的地址信息及数据，执行写入使能、扇区擦除、编程及读取等任务。为了保证数据完整性和存储持久性，系统还集成了错误检测与纠正逻辑，用以在数据写入或读取过程中发现并修正潜在错误。

在线升级流程始于主控计算机通过 RS422 串行接口与 FPGA 建立通信链路。FPGA 作为中间层，接收由主控计算机传来的最新固件数据，并通过 SPI 接口将其转发至 Flash 存储器。首先，FPGA 利用 S25FL256S 特有的设备识别命令（0x9F）来验证连接设备的身份，确认其符合预期。在确认设备 ID 无误之后，FPGA 发送写入使能命令（0x06），使 Flash 进入可编程状态[8]。接下来，FPGA 指定要擦除的扇区，并发送相应的扇区擦除命令（0x20）。鉴于擦除操作的异步特性，FPGA 需周期性查询状态寄存器直至状态标志显示“就绪”，表明擦除操作已完成。擦除完成后，FPGA 再次读取状

表 1　S25FL256S 存储结构与地址映射表

Block Size /kbyte	Block Count	Block Range	Half Block Size/kbyte	Half Block Count	Half Block Range	Sector Size /kbyte	Sector Count	Sector Range	Address Range (Byte Address)	Notes
64	1	BA00	32	1	HBA00	4	1	SA00	0000000h—0000FFFh	Sector Starting Address — Sector Ending Address
						…	…	…	…	
			32	2	HBA01	4	16	SA15	000F000h—000FFFFh	
…	…	…	…	…	…	…	…	…	…	
64	512	BA511	32	1023	HBA1022	4	8176	SA8175	1FF0000h—1FF0FFFh	
						…	…	…	…	
			32	1024	HBA1023	4	8192	SA8191	1FFF000h—1FFFFFFh	

态寄存器以确认擦除成功，即所有位均已被清零。对于一个 10 MB 的固件存储区而言，考虑到 S25FL256S 每个扇区容量为 64 kB，预计需要执行约 160 次扇区擦除[9]。在此过程中，必须考虑擦除不均匀、擦除次数上限以及擦除速率等因素，以确保擦除过程既高效又可靠。

当擦除阶段结束后，新的固件数据通过 RS422 接口从主控计算机传输至 FPGA，并暂存于一个先进先出（FIFO）缓存区中。FPGA 持续监测 FIFO 的状态，在 FIFO 有数据时，将通过 RS422 接口向 Flash 发送写入使能命令，随后写入命令、数据及其对应的页地址。鉴于 S25FL256S 每页的最大编程容量为 256 字节，因此需要多次重复写入使能和编程命令，直至全部数据写入完毕。当 FIFO 检测为空时，FPGA 启动一个计时器以等待可能的后续数据。若一秒钟内未收到新数据，则系统假定数据传输已完成，并禁用写入功能，安全终止整个配置流程。如此，Flash 控制模块得以有效地管理和控制 S25FL256S Flash 存储器的操作，确保固件更新过程的安全性和效率，Flash 操作流程如图 4 所示。

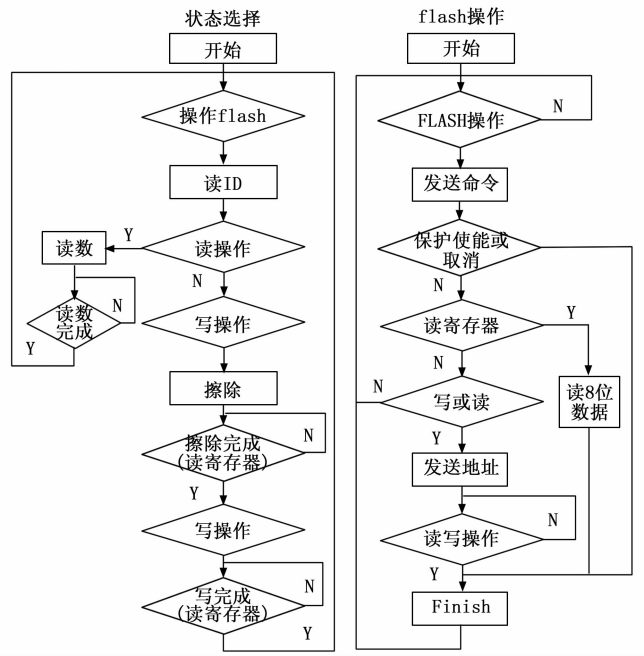

图 4 FLASH 操作流程图

3.2 ICAP 跳转模块逻辑设计

3.2.1 多重配置技术

ICAP 跳转模块是在线升级工程中的核心模块，这里的跳转指的是芯片重配置后读取更新后的区域中的程序。FPGA 多重配置技术即 Multiboot 功能是一种先进的资源复用策略，其核心在于允许用户根据实际需求，灵活地为 FPGA 加载不同的配置文件，从而实现多重配置。这一技术不仅极大地提升了 FPGA 资源的利用率，使得宝贵的硬件资源能够在不同应用场景下得到充分利用，还显著增强了设计的灵活性。通过多重配置，用户可以根据需要调整 FPGA 的功能和行为，而无需重新设计或更换硬件，这大大降低了开发成本和周期。

多重配置技术的实现离不开 FPGA 内部的 ICAPE2 模块。当需要启动重配置流程时，ICAPE2 模块会发出内部编程指令（IPROG），引导 FPGA 从 Flash 存储器的指定地址开始读取新的配置文件。这个起始地址由重载起始地址（WBSTAR）寄存器来指定，它决定了 FPGA 将从 Flash 的哪个位置开始读取新的比特流文件[10]。在 IPROG 指令发出之后，FPGA 会按照一系列精确的操作步骤进行配置文件的读取与加载。首先，FPGA 会发送一个同步字（AA995566），这个同步字是为了确保 FPGA 与 Flash 存储器之间的同步，确保数据能够准确无误地传输。接着，FPGA 会向 WBSTAR 寄存器中写入下一个比特流文件的起始地址（通常为 00000000），这个地址指定了新配置文件在 Flash 中的位置，FPGA 将从这个位置开始读取新的比特流数据[11]。最后，FPGA 会再次发送 IPROG 命令（0000000F），这个命令将正式启动配置文件的读取与加载过程，FPGA 将开始根据新的配置文件进行配置和初始化。IPROG 指令如表 2 所示。

表 2 IPROG 指令定义

配置指令	说明
FFFFFFFF	Dummy Word
AA995566	Sync Word
20000000	Type 1 NO OP
30020001	Type 1 Write 1 Words to BSTAR
00000000	Warm Boot Start Address
30008001	Type 1 Write 1 Words to CMD
0000000F	IPROG Command
20000000	Type 1 NO OP

3.2.2 Fallback 模式

在进行在线程序升级的过程中，设备可能会遭遇突如其来的掉电或其他非预期状况。一旦此类情况发生，当设备重新接通电源后，FPGA 会尝试从 Flash 存储器中加载程序以恢复正常运行。然而，由于旧有的程序在升级过程中已被擦除，而新的配置文件又未能完全写入，这导致设备在此时无法通过 RS422 接口接收来自上位机的指令和新的配置文件[12]。FPGA 也会因此失去对 Flash 存储器的控制能力，使得整个系统陷入无法自救的困境。

为了避免此类情况的发生，Xilinx 七系列 FPGA 内置了一种 Fallback 操作机制，作为 Multiboot 过程中的安全防护措施。即当系统接收到在线升级指令时，触发 Multiboot 从当前配置切换到目标配置的过程中出现错误或失败时，Fallback 机制会自动介入，使用预设的 Golden bit 流文件（一种经过验证、高度可靠的 bit 流

文件）来重新配置 FPGA，确保即使在异常情况下，系统也能迅速恢复至一个已知且稳定的工作状态[13]。Fallback 机制工作流程如图 5 所示。通过引入 Fallback 机制，该设计不仅显著提高了固件更新的可靠性，还大大增强了系统的整体稳定性。即使在程序更新过程中发生意外掉电或配置错误等异常情况，FPGA 也能迅速恢复并继续完成在线升级，从而避免了因异常情况导致的设备无法正常工作的问题[14]。

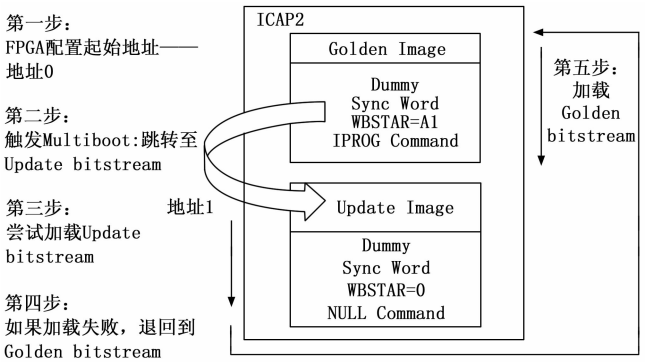

图 5　Fallback 机制工作流程

4　数据可靠性优化

4.1　RS 编码

Reed-Solomon（RS）编码是一种基于有限域理论的非二进制纠错编码技术，广泛应用于数据存储系统和传输领域。它通过在原始数据中添加冗余信息来增强传输数据的可靠性，能够在接收端纠正一定数量的符号错误，有效应对由信道干扰引起的误码问题。RS 编码的基础是有限域（也称为伽罗华域，Galois Field），记作 $GF(q)$，其中 q 是一个素数幂。最常见的 RS 编码使用的是 $GF(2^m)$，其中 m 为正整数，通常 $m=8$ 表示每个符号由 8 个比特组成，即一个字节[15]。在这种情况下，每个符号实际上是一个字节，因此非常适合处理数字通信中的二进制数据。有限域中的运算规则不同于普通算术，具有特殊的加法、乘法等运算规则，确保了其封闭性和逆元的存在性。例如，在 $GF(2^8)$ 中，每个符号是一个字节，因此每个符号可以表示为 0～255 之间的整数[16]。

RS 编码的核心思想是将原始数据分割成若干个符号，这些符号可以看作是多项式的系数，而生成多项式则是用于创建冗余符号的关键工具。具体来说，在 $GF(2^8)$ 中，每个符号是一个字节，因此每个符号可以表示为 0～255 之间的整数。假设我们有一个数据块，包含 k 个符号，RS 编码的目标是生成 n-k 个冗余符号，使得总符号数达到 n[17]。这样即使在传输过程中有最多 $(n-k)/2$ 个符号发生错误，也可以通过解码恢复原始数据。生成多项式 $G(x)$ 是一个 n-k 阶多项式，其中 n 是总符号数，k 是数据符号数。这个多项式决定了冗余符号的数量和位置。生成多项式的根是有限域中的特定元素，称为“校验根”[18]。对于 $RS(n, k)$ 编码，生成多项式通常是：

$$G(x)=(x-\alpha^{c})(x-\alpha^{c+1})\cdots(x-\alpha^{c+n-k-1}) \quad (1)$$

其中：α 是有限域中的本原元素，c 是起始指数，可以根据具体应用选择不同的参数。例如，在 $GF(2^8)$ 中，如果我们选择 $n=255$，$k=239$，则生成多项式为：

$$G(x)=(x-\alpha^{0})(x-\alpha^{1})\cdots(x-\alpha^{15}) \quad (2)$$

这意味着我们将生成 16 个冗余符号，从而允许纠正最多 8 个符号错误。

在编码过程中，首先将原始数据分成多个长度为 k 的符号序列，然后根据所需冗余符号的数量确定生成多项式 $G(x)$。接下来，将每个数据符号视为多项式 $P(x)$ 的系数，通过多项式除法得到余数 $R(x)$，这些余数就是冗余符号。最后，将原始数据符号和冗余符号组合在一起，形成最终的编码数据块[19]。在接收端，解码过程包括计算伴随矩阵，根据接收到的数据符号计算伴随矩阵，用于检测和定位错误；求解错误位置多项式，利用伴随矩阵找到错误位置多项式，确定哪些符号发生了错误；求解错误值多项式，进一步计算错误值多项式，确定每个错误符号的实际值；纠正错误，根据错误位置和错误值对错误符号进行修正，恢复原始数据。

4.2　数据可靠性优化设计

传统的数据校验方式如 CRC 校验虽然具有较强的检错能力，但并非所有错误都能被检测到。例如，当错误发生在多个位上，且这些错误位之间的关系恰好符合 CRC 多项式的特性时，可能会导致错误未被检测出来。为了进一步提高数据传输的可靠性，特别是在 FPGA 在线升级过程中可能遇到的突发错误和噪声干扰，设计了一种结合 CRC 校验和 Reed-Solomon（RS）编码的技术方案。在 FPGA 在线升级过程中，选择 RS（255，223）作为编码参数。该编码参数意味着每个数据块包含 223 个输入数据符号，每个符号为 8 比特（即一个字节），并添加 32 个冗余监督码符号。这种配置最多可以纠正 16 个符号错误，显著增强了系统的纠错能力[20]。为了进一步增强数据传输的可靠性，选择了 CRC-16 校验方式。CRC-16 是一种常见的循环冗余校验方法，能够有效检测出大多数单比特错误、双比特错误以及奇数个比特错误。尽管 CRC-16 无法检测所有类型的错误，但它与 RS 编码相结合，能够提供更全面的错误检测和纠正机制。优化后的数据帧格式如表 3 所示。

表 3　优化后数据帧格式　　bit

起始位	地址/命令	长度	数据字段	RS 编码冗余信息	CRC 校验值	停止位
1	8	8	223 * 8	32 * 8	16	1

发送端将数据分割成较小的数据块，对每个数据块

进行 RS 编码，为每包数据添加固定格式的包头，并计算 CRC 校验值，将其附加到包尾。具体的数据帧结构包括包头信息、包计数、编码后的数据块、冗余符号及 CRC 校验值。包头信息包含起始标志、数据长度、版本号等字段，用于标识数据帧的开始和基本属性；包计数用于确保数据帧的顺序性和完整性，防止数据丢失或乱序；经过 RS 编码处理的数据块包含原始数据符号和冗余监督码符号；CRC 校验值附加在数据帧末尾，用于验证数据帧的完整性。最后，将包头、包计数、编码后的数据块、冗余符号及 CRC 校验值组合成一个完整的数据帧，并通过 RS422 接口逐个发送构建好的数据帧给接收端。

接收到数据帧后，首先校验包头信息，包括验证包头格式和确认包计数的连续性。如果发现错误，接收端会立即向发送端发出重传请求，要求重新发送该数据包；一旦包头信息校验通过，接收端将继续进行 CRC 校验，对比自身计算的 CRC 值与接收到的 CRC 值。如果 CRC 校验通过，则继续下一步；否则，请求重传。为了防止因恶劣环境导致陷入无限循环，重传次数最多为 5 次。当 CRC 校验通过时，接收端便会执行 RS 解码，纠正可能存在的符号错误。RS 解码过程包括计算伴随矩阵、求解错误位置多项式、求解错误值多项式，并根据错误位置和错误值对错误符号进行修正。最后，将处理后的正确数据传递给上位机，完成整个数据传输过程。优化后的数据流程如图 6 所示。

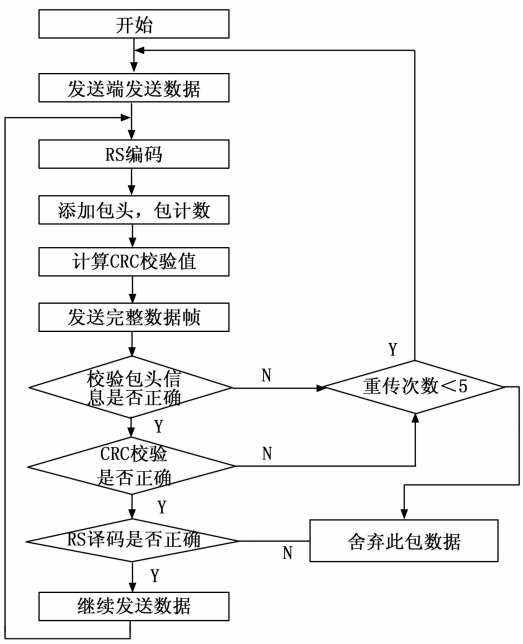

图 6 优化后数据流程图

5 测试验证

5.1 RS 编码验证

为了验证 RS 编码的有效性，使用 Vivado 软件对 RS 编解码过程进行了详细仿真。编码仿真如图 7 所示，展示了 RS 编码模块在不同阶段的信号变化和数据流情况。在本次仿真中，向 RS 编码器输入了 223 个递增的数据，作为原始数据块。每个数据符号为一个字节，这些数据通过 rs _ input _ tdata 端口逐个输入到编码模块。rs _ input _ tvalid 信号用于指示当前输入数据的有效性，而 rs _ input _ tready 信号则确认编码器已准备好接收新的数据符号。这种握手机制确保了数据传输的同步性和准确性。当所有 223 个数据符号成功输入后，RS 编码模块根据选定的生成多项式计算并添加了 32 个冗余符号。编码后的数据通过 rs _ output _ tdata 端口输出，由 223 个原始数据符号加上 32 个冗余符号组成编码后的数据。

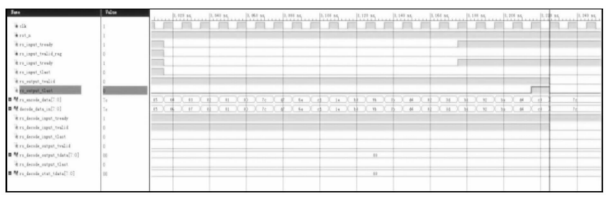

图 7 RS 编码仿真结果图

图 8 展示了 RS 解码模块的仿真结果。该图表明，解码模块在接收到编码数据后，能够正确地执行解码过程，从而确保数据传输的准确性和可靠性。为了验证 RS 编码的纠错能力，对编码器的输出增加了两个字节的误码，将编码器的输出数据 rs _ encode _ dates 中的‘04 03’更改为‘06 07’，以模拟数据传输过程中可能发生的错误。图 9 展示了引入错误码后的仿真结果，其中 decode _ date _ in 为译码器的输入，显示了错误的数据。图 10 为纠错结果仿真图，可以看到 rs _ decode _ output _ tdate 输出恢复成了正常的‘04 03’，实现了纠错，rs _ decode _ stat _ tdate 的输出值为 1e，这表明译码器检测到了输入数据中的错误；最低位为 0 说明没有超出纠错范围，实现了纠错。

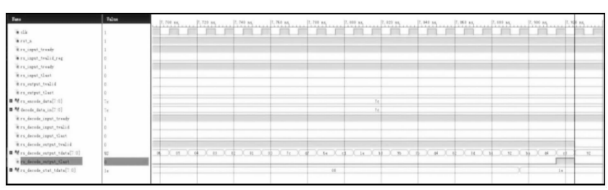

图 8 RS 解码仿真结果图

图 9 RS 误码产生仿真图

图 10　RS 纠码结果仿真图

表 4 展示了 RS 编码在处理不同分布模式和数量的误码时的性能表现。在测试中，当误码数量较少且分布较为分散时，RS 编码能够实现高达 100％的纠错成功率。这表明在低误码环境下，RS 编码具有出色的纠错能力。然而，当误码数量增加或误码分布较为集中时，可能会超出 RS 编码的最大纠错能力。此时 Fallback 机制的触发确保了系统的稳定性和可靠性，保障了系统在面对无法纠正的错误时仍能正常运行。

表 4　RS 编码可变误码数量测试表

误码数量	误码分布模式	仿真配置参数	测试结果
1 字节	随机分散	*SNR*＝20 dB 数据长度 223 B	*BER*＝0 （100％纠正）
8 字节	连续突发	突发长度 8 B 位置 120～127 B	纠错耗时 0.8 μs
16 字节	前 8 B＋ 后 8 B 分散	误码位置 0x10/0xE0	纠错后 CRC32 校验通过
17 字节	全数据段随机	误码率 7.8％	触发 Fallback 机制

5.2　在线升级验证

为了验证通过 RS422 接口对 FPGA 进行在线升级的可行性，准备了两份配置文件：一份是作为基准的原始配置文件（Golden Bitstream），另一份则是用于系统更新的目标配置文件（Update Bitstream）。通过 RS422 接口将上位机与 FPGA 开发板相连后，使用串口调试助手软件设置适当的通信参数，包括波特率、数据位和停止位等，以确保二者之间的有效通信。接下来，我们将原始配置文件及待升级的新配置文件依次上传至 FPGA 开发板，并进行了全面的功能测试，图 11 为 FPGA 部分程序烧写时序仿真图。测试结果显示，FPGA 能通过 RS422 接口顺利完成在线升级过程，整个传输过程中未发生任何丢包现象，且系统运行状态稳定可靠。此外，针对可能出现的意外断电等情况，一旦系统检测到异常情况，可以立即触发 Fallback 模式，退回初始配置文件状态，为后续的维护和故障排除提供了便利。

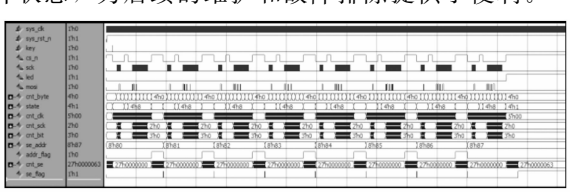

图 11　FPGA 部分程序烧写仿真时序图

为了综合验证 RS422 接口在线升级的综合性能，以配置文件大小为 15.6 MB 的升级文件在不同波特率下进行测试，如表 5 所示，在较低的波特率下，尽管传输时间较长，但传输效率较高，且对 FPGA 资源的占用相对较少。这种配置适用于长距离传输或噪声较大的环境，确保数据传输的可靠性和稳定性。中等波特率显著缩短了传输时间，适合对实时性有一定要求的应用场景。虽然资源占用略有增加，但仍保持在一个合理的范围内，能够满足大多数应用场景的需求。在高波特率下，传输时间大幅减少，非常适合需要快速升级的场景。然而，传输效率有所下降，同时对 FPGA 资源的占用也显著增加。这种配置适用于对传输速度有极高要求且资源充足的系统。

表 5　RS422 接口在线升级性能分析表

配置文件大小/MB	波特率/bps	理论传输时间	实际传输时间	传输效率/％	资源占用率/％（LUT/BRAM）
15.6	115 200	23 min12 s	24 min58 s	92.8	34/28
15.6	921 600	2 min54 s	3 min12 s	90.6	37/31
15.6	10 M	12.5 s	14.3 s	87.4	45/39

6　结束语

通过这种在线升级方式，用户能够实现远程设备升级和维护，显著提升了设备的可维护性和可用性。这种升级机制不仅降低了维护成本和停机时间，还增强了系统的安全性和可靠性。此外，它为智能化管理和远程监控提供了技术支持，使得设备能够快速适应市场变化和技术进步。通过灵活调整 FPGA 功能，用户可以延长设备使用寿命，实现更高效的资产管理。随着通信技术的发展，在线升级的效率和安全性将进一步提高，为自动化和智能化领域带来更大的潜力。

参考文献：

[1] 刘　钊，杜永锋，许知博. 基于 Xilinx-Spartan6 FPGA 的 MultiBoot 设计的实现 [J]. 电子科技，2012，25（3）：28－31.

[2] 吴海龙. 一种基于 Xilinx FPGA 的串口在线升级方法 [J]. 单片机与嵌入式系统应用，2023，23（8）：50－52.

[3] 李　霄，徐思远，胡瑾贤. 基于 FPGA 的远程快速在线升级技术研究 [J]. 舰船电子对抗，2023，46（4）：98－102.

[4] 姜晓道，赵紫君. 基于 IAP 的通用嵌入式系统在线升级功能设计 [J]. 电子制作，2022，30（6）：20－23.

[5] CAO M，JIN T，ZHAO Y，et al. On-orbit update and scrubbing design of SRAM FPGA for spacecraft [J]. Journal of Physics：Conference Series，2024，2870（1）：012017.

[6] 李　平，吴　晓，山　寿. 基于 SPI FLASH 的 FPGA 多重配置 [J]. 现代电子技术，2013，36（22）：127－130.

（下转第 181 页）