**基于Xilinx FPGA的电容感应触控系统设计与实现**

**院系：电子信息与通信学院 班级：电信2101**

**姓名：李鑫瑞 学号：U202113871**

**目录**

1. 项目概述
2. 设计与实现背景
3. 项目功能指标
4. 团队分工
5. 系统框图
6. 各模块原理 6.1 电容检测电路 6.2 FPGA控制平台 6.3 蓝牙模块电路
7. 理论分析 7.1 电容感应原理 7.2 FPGA的工作原理 7.3 串口通信原理
8. 关键技术 8.1 FPGA设计 8.2 串口通信 8.3 蓝牙通信 8.4 上位机程序开发
9. 模块测试 9.1 电容检测模块测试 9.2 FPGA数据处理测试 9.3 蓝牙通信测试
10. 器件清单
11. 总结及心得体会
12. 参考资料
13. 附录 13.1 电路图 13.2 源码清单及注释

**1. 项目概述**

本项目旨在开发一个基于Xilinx FPGA的智能控制系统，通过电容触摸板实现对终端设备的控制，具体应用包括PPT翻页、文档输入、钢琴演奏以及无线游戏手柄等。该系统通过检测微小的电容变化，将其转换为电信号，并通过蓝牙模块将信号传输到上位机或终端设备，从而实现对设备的远程控制。

**项目目标**

本项目的目标是设计并实现一个以Xilinx FPGA为核心控制器的智能控制系统，通过电容触摸板检测用户的触摸动作，并将相应的控制信号通过蓝牙模块传输到上位机或终端设备，实现以下功能：

1. **PPT翻页和文档输入**：用户可以通过触摸纸上的按键区域，远程控制PPT的翻页或在文档中输入字符。
2. **钢琴演奏**：用户可以在纸上绘制钢琴轨迹，通过触摸不同的键位，控制计算机上的钢琴程序发音，实现虚拟钢琴演奏。
3. **无线游戏手柄**：用户可以在纸上绘制游戏控制区域，通过触摸这些区域，控制游戏角色的移动和操作，实现无线游戏手柄功能。

**项目意义**

本项目的实施具有重要的应用价值和创新意义：

1. **低成本、高灵活性**：相较于传统的触控屏，本项目通过电容传感器和FPGA实现触控检测，成本更低，且可以根据需要自由绘制触控区域，灵活性更高。
2. **多功能一体化**：通过FPGA的强大处理能力和蓝牙模块的无线传输功能，本项目实现了多种功能的集成，用户可以在同一平台上实现PPT翻页、文档输入、钢琴演奏和游戏控制等多种操作。
3. **提升用户体验**：本项目提供了一种全新的操作方式，用户只需在纸上绘制触控区域，即可通过触摸实现对终端设备的控制，操作简单直观，提升了用户体验。

**系统组成**

整个系统由三部分组成：电容检测电路、FPGA控制平台和蓝牙模块电路。各部分的功能和连接如下：

1. **电容检测电路**：负责检测用户的触摸动作，将电容变化转换为电信号，并传递给FPGA。主要包括电容传感器和信号转换电路。
2. **FPGA控制平台**：负责接收电容检测电路的信号，对其进行处理和分析，生成相应的控制信号，并通过蓝牙模块发送给终端设备。主要包括FPGA芯片和信号处理逻辑。
3. **蓝牙模块电路**：负责接收FPGA生成的控制信号，通过蓝牙协议将信号传输到上位机或终端设备，实现无线控制。主要包括蓝牙模块和通信接口电路。

**工作流程**

整个系统的工作流程如下：

1. 用户触摸电容传感器，电容值发生变化。
2. 电容检测电路将电容变化转换为电信号，并传递给FPGA。
3. FPGA对电信号进行处理，确定触摸位置和相应的控制信号。
4. FPGA通过蓝牙模块将控制信号无线传输到上位机或终端设备。
5. 上位机或终端设备接收到控制信号后，执行相应的操作。

**项目特点**

本项目具有以下特点：

1. **实时性**：由于FPGA具有高效的并行处理能力，能够实时处理触控信号，确保系统的响应速度。
2. **无线传输**：通过蓝牙模块实现无线传输，用户可以在一定范围内自由移动，无需受限于电缆连接。
3. **多功能集成**：系统集成了PPT翻页、文档输入、钢琴演奏和无线游戏手柄等多种功能，用户可以根据需要自由切换。
4. **简单易用**：用户只需在纸上绘制触控区域，即可通过触摸实现控制，操作简单直观。

**潜在应用**

本项目的设计方案具有广泛的应用前景：

1. **智能办公**：用户可以通过触摸纸上的按键区域，远程控制PPT的翻页或在文档中输入字符，提高办公效率。
2. **虚拟乐器**：用户可以在纸上绘制乐器键盘，通过触摸实现虚拟乐器的演奏，适用于音乐教学和演出。
3. **互动游戏**：用户可以在纸上绘制游戏控制区域，通过触摸控制游戏角色的移动和操作，适用于互动式娱乐设备。
4. **智能家居**：通过扩展电容传感器的应用范围，可以实现智能家居设备的触控控制，如灯光调节、门锁控制等。

**项目实施计划**

为了确保项目顺利进行，我们制定了详细的实施计划，包括设计阶段、开发阶段、测试阶段和优化阶段等。

1. **设计阶段**：
   * 确定系统功能需求，设计电容检测电路、FPGA控制逻辑和蓝牙通信协议。
   * 绘制系统框图和电路原理图，进行器件选型和采购。
2. **开发阶段**：
   * 搭建硬件平台，完成电容检测电路、FPGA控制平台和蓝牙模块的硬件连接。
   * 编写FPGA信号处理逻辑和蓝牙通信程序，完成软件编程。
3. **测试阶段**：
   * 对各模块进行单独测试，确保其功能和性能满足设计要求。
   * 进行系统集成测试，验证系统的整体功能和性能。
4. **优化阶段**：
   * 根据测试结果，优化系统设计和软件算法，提高系统的稳定性和响应速度。
   * 完善用户界面和操作说明，提升用户体验。

**项目挑战与解决方案**

在项目实施过程中，我们面临了一些技术挑战，包括电容检测的精度和灵敏度、FPGA信号处理的实时性和效率、蓝牙通信的稳定性和可靠性等。通过团队的共同努力，我们提出并实施了一系列解决方案：

1. **电容检测的精度和灵敏度**：
   * 采用高精度电容传感器，优化检测电路设计，提高信号转换的准确性。
   * 通过实验调整传感器参数，确保其在不同触摸条件下的灵敏度和稳定性。
2. **FPGA信号处理的实时性和效率**：
   * 设计高效的信号处理算法，充分利用FPGA的并行处理能力，实现实时信号处理。
   * 优化FPGA逻辑设计，减少数据处理延迟，提高系统的响应速度。
3. **蓝牙通信的稳定性和可靠性**：
   * 选择性能优良的蓝牙模块，确保无线传输的稳定性和可靠性。
   * 设计合理的通信协议，处理可能出现的干扰和数据丢失，确保数据传输的完整性。

**2. 设计与实现背景**

**2.1 电容感应技术简介**

电容感应技术通过检测电容变化来感知触摸。电容器由两个导电板和一个介质组成，当有人触摸时，导电板之间的电场会发生变化，导致电容值发生变化。通过检测这种变化，我们可以确定触摸事件的位置和强度。

电容传感器广泛应用于各种现代设备，如智能手机、平板电脑和触控显示器。这些传感器以其高灵敏度、低功耗和高耐用性而闻名。传统的电容触控系统通常采用专用的触控控制芯片，但在本项目中，我们选择使用FPGA作为核心控制器，以便充分利用其高性能和可编程性。

**2.2 FPGA的优势**

FPGA（现场可编程门阵列）是一种高度可配置的集成电路，可以在硬件级别实现并行处理。相比于传统的微控制器或DSP，FPGA在处理速度和灵活性上具有明显优势，特别适用于需要高速并行处理的应用场景。

FPGA内部包含数百万个可配置的逻辑单元和可编程互连，可以根据需要配置成不同的电路结构。这使得FPGA在各种应用中具有极大的灵活性和适应性，特别是在需要高性能和低延迟的应用场合，如数字信号处理、通信系统和图像处理等领域。

**2.3 蓝牙通信**

蓝牙是一种短距离无线通信技术，广泛应用于各种电子设备之间的数据传输。蓝牙模块通常支持标准的串行通信协议（如UART），便于与其他设备（如FPGA）进行数据交换。蓝牙技术的发展使得其在短距离无线通信中的应用变得更加普遍和重要。

蓝牙模块主要由以下部分组成：

1. **UART接口**：与FPGA进行数据通信。
2. **蓝牙协议栈**：实现蓝牙通信的各种协议和功能。
3. **天线**：用于无线信号的发送和接收。

蓝牙模块的实现包括硬件连接和软件配置两部分。硬件连接主要是FPGA与蓝牙模块通过UART接口连接，确保数据的正确传输。软件配置包括设置蓝牙模块的工作参数和通信协议。蓝牙模块可以通过UART接口与FPGA进行通信，FPGA将处理后的触控数据通过串口发送给蓝牙模块，蓝牙模块再通过无线方式将数据发送到上位机或终端设备。这样，用户可以方便地通过蓝牙连接控制各种应用程序，实现远程控制和数据传输。

**3. 项目功能指标**

**3.1 电容检测**

准确检测电容按键的触控状态，并将其转换为数字信号。

电容检测电路的核心是电容传感器，通过放大电路和滤波电路，将微小的电容变化转换为稳定的电压信号。然后通过模数转换器（ADC）将模拟信号转换为数字信号，供FPGA处理。电容检测电路需要具备高灵敏度和低噪声特性，以确保能够准确检测到触控操作。

**3.2 数据处理**

FPGA对电容检测信号进行处理和解析，确定触控的位置和状态。

FPGA通过内部逻辑实现电容检测信号的处理和识别，并将处理后的数据通过串口传输给蓝牙模块。FPGA的设计需要考虑信号的同步和异步处理、数据的存储和传输、以及系统的整体性能和稳定性。数据处理模块需要具备实时性和可靠性，以确保系统能够快速响应触控操作。

**3.3 数据传输**

通过串口与蓝牙芯片通信，将处理后的触控数据无线传输到上位机或终端设备。

蓝牙模块通过串口与FPGA通信，将接收到的数据通过无线方式发送到上位机或终端设备。蓝牙模块通常支持标准的串行通信协议（如UART），确保数据传输的可靠性和稳定性。数据传输模块需要具备高带宽和低延迟特性，以确保数据能够快速、准确地传输到上位机或终端设备。

**3.4 应用功能**

实现控制PPT翻页、输入文档、虚拟钢琴演奏和无线游戏手柄功能。

通过蓝牙通信，上位机程序接收到触控数据后，根据数据的内容执行相应的操作。上位机程序需要处理接收到的数据，并根据用户的需求实现相应的功能，如控制PPT翻页、输入文档、演奏虚拟钢琴或控制游戏手柄。应用功能模块需要具备灵活性和可扩展性，以满足不同用户的需求。

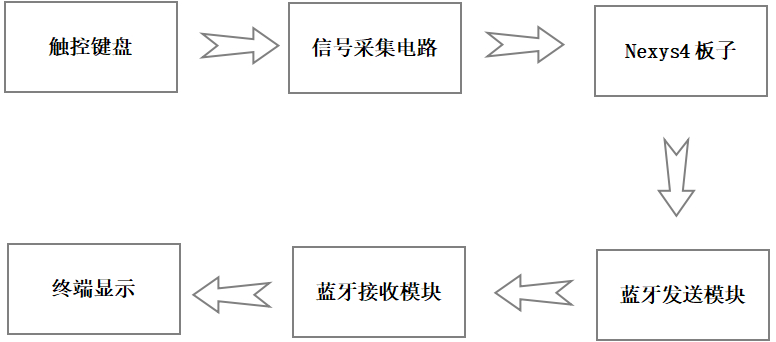
**4. 团队分工**

上位机编程---吴永浩：  
负责整体系统设计和架构规划。  
设计和实现蓝牙模块的集成。  
完成电脑端键鼠输入信号处理的设计。

FPGA编程----李鑫瑞：  
负责触控板信号的输入处理。  
负责触控板性能的升级。  
搭建测试平台和环境，对系统进行验证和调试。

系统调试----王梦婷：  
 负责处理触控板输入信号并传输至蓝牙发送端。  
 负责对硬件和软件进行系统联调。  
 搭建测试平台和环境，对系统进行验证和调试。

**5. 系统框图**

**6. 各模块原理**

**6.1 电容检测电路**

电容检测电路的核心是电容传感器，通过放大电路和滤波电路，将微小的电容变化转换为稳定的电压信号。然后通过模数转换器（ADC）将模拟信号转换为数字信号，供FPGA处理。

电容检测电路需要具备高灵敏度和低噪声特性，以确保能够准确检测到触控操作。电容传感器可以采用自容模式或互容模式，自容模式通过测量单个电极的电容变化来检测触控，而互容模式则通过测量两个电极之间的电容变化来检测触控。放大电路用于放大微小的电容变化信号，滤波电路用于去除噪声，确保信号的稳定性和准确性。

**6.2 FPGA控制平台**

FPGA的主要任务是从电容检测电路获取数据，并对数据进行处理和解析。FPGA通过内部逻辑实现电容检测信号的处理和识别，并将处理后的数据通过串口传输给蓝牙模块。

FPGA的设计需要考虑信号的同步和异步处理、数据的存储和传输、以及系统的整体性能和稳定性。数据处理模块需要具备实时性和可靠性，以确保系统能够快速响应触控操作。FPGA通过使用硬件描述语言（如Verilog或VHDL）编写内部逻辑，实现电容检测信号的处理、蓝牙通信接口和数据传输控制。

**6.3 蓝牙模块电路**

蓝牙模块通过串口与FPGA通信，将接收到的数据通过无线方式发送到上位机或终端设备。蓝牙模块通常支持标准的串行通信协议（如UART），确保数据传输的可靠性和稳定性。

蓝牙模块可以使用SPP（Serial Port Profile）协议实现数据的无线传输。SPP协议是一种常用的蓝牙协议，支持串行数据的无线传输，适用于短距离的无线通信。蓝牙模块电路需要具备高带宽和低延迟特性，以确保数据能够快速、准确地传输到上位机或终端设备。

**7. 理论分析**

**7.1 电容感应原理**

电容感应原理基于电容的变化。当用户触摸电容按键时，电容值发生变化，产生一个微小的电压变化。通过放大和滤波电路，将这个变化信号放大并稳定，然后通过ADC将其转换为数字信号。FPGA接收到这个数字信号后，对其进行处理和识别，确定触控的位置和状态。

电容器的电容值由两个导电板的面积、板间距离和介质材料的介电常数决定。触摸电容按键时，导电板间的电场发生变化，导致电容值发生变化。通过检测这个变化，可以确定触摸事件的位置和强度。

**7.2 FPGA的工作原理**

FPGA通过配置文件（bitstream）实现其内部逻辑功能。设计者可以使用硬件描述语言（如Verilog或VHDL）编写FPGA的逻辑，生成bitstream文件，然后将其下载到FPGA中，使FPGA实现特定的功能。在本项目中，FPGA的主要任务是处理电容检测信号，并通过串口与蓝牙模块通信。

FPGA内部包含数百万个可配置的逻辑单元和可编程互连，可以根据需要配置成不同的电路结构。这使得FPGA在各种应用中具有极大的灵活性和适应性，特别是在需要高性能和低延迟的应用场合，如数字信号处理、通信系统和图像处理等领域。

**7.3 串口通信原理**

串口通信是一种常用的通信方式，通常采用UART协议。UART协议是一种异步通信协议，通过发送和接收数据位实现数据传输。在本项目中，FPGA通过UART协议与蓝牙模块通信，将处理后的触控数据发送到上位机或终端设备。

UART协议采用异步通信方式，不需要发送和接收双方共享时钟信号，而是通过起始位、数据位、校验位和停止位实现数据的同步传输。UART通信需要配置波特率、数据位、停止位和校验位等参数，以确保通信的正确性和稳定性。

**8. 关键技术**

**8.1 FPGA设计**

使用Verilog HDL或VHDL编写FPGA内部逻辑，实现电容检测信号的处理、蓝牙通信接口和数据传输控制。FPGA的设计需要考虑信号的同步和异步处理、数据的存储和传输、以及系统的整体性能和稳定性。

FPGA设计过程包括设计输入、综合、布局布线和时序分析等步骤。设计输入阶段，使用硬件描述语言（如Verilog或VHDL）编写设计代码；综合阶段，通过综合工具将设计代码转换为门级网表；布局布线阶段，通过布局布线工具将网表映射到FPGA的物理资源上；时序分析阶段，通过时序分析工具验证设计的时序性能。

**8.2 串口通信**

通过UART协议实现FPGA与蓝牙模块之间的数据传输。UART协议需要配置波特率、数据位、停止位和校验位等参数，以确保通信的正确性和稳定性。

UART通信采用异步通信方式，不需要发送和接收双方共享时钟信号，而是通过起始位、数据位、校验位和停止位实现数据的同步传输。UART通信的波特率、数据位、停止位和校验位等参数需要根据通信双方的要求进行配置，以确保通信的正确性和稳定性。

**8.3 蓝牙通信**

使用SPP（Serial Port Profile）协议实现数据的无线传输。SPP协议是一种常用的蓝牙协议，支持串行数据的无线传输，适用于短距离的无线通信。

蓝牙通信通过使用SPP协议实现数据的无线传输。SPP协议是一种常用的蓝牙协议，支持串行数据的无线传输，适用于短距离的无线通信。蓝牙通信需要考虑数据的传输速率、传输距离和传输稳定性等因素，以确保数据能够快速、准确地传输到上位机或终端设备。

**8.4 上位机程序开发**

使用C#或Labview编写上位机程序，实现数据接收和应用功能。上位机程序需要处理接收到的数据，并根据数据的内容执行相应的操作，如控制PPT翻页、输入文档、演奏虚拟钢琴或控制游戏手柄。

上位机程序开发需要考虑用户界面的设计、数据的处理和应用功能的实现。用户界面需要具备友好性和易用性，以便用户能够方便地操作和使用系统。数据处理模块需要具备实时性和可靠性，以确保系统能够快速响应触控操作。应用功能模块需要具备灵活性和可扩展性，以满足不同用户的需求。

**9. 模块测试**

**9.1 电容检测模块测试**

通过模拟触控输入，验证电容检测电路的准确性和灵敏度。使用示波器观测电压变化，并通过FPGA输出信号进行对比，确保电容检测电路能够准确响应触控操作。

电容检测模块测试需要考虑电容传感器的灵敏度、放大电路的增益和滤波电路的频率响应等因素。通过模拟触控输入，观察电容检测电路的输出信号，验证电容检测电路的准确性和灵敏度。

电容模块的工作原理是基于电容传感器对电容变化的敏感性。当用户触摸电容传感器时，传感器的电容值会发生微小变化。这一变化通过检测电路被转换为电信号，随后传递给FPGA进行处理。整个过程包括以下几个步骤：

1. **电容变化检测**：通过电容传感器检测用户的触摸动作，获取电容值的变化。
2. **信号转换**：将电容值的变化转换为电信号，方便FPGA读取和处理。
3. **数据传输**：将电信号传递给FPGA，进行后续的数据处理和分析。

电容模块的实现包括硬件设计和软件编程两部分。硬件设计主要涉及电容传感器和检测电路的连接，软件编程则包括信号处理和数据传输的实现。

**硬件设计**：

* + **电容传感器**：选择合适的电容传感器模块，如TTP226电容传感器，用于检测用户的触摸动作。
  + **检测电路**：设计电容检测电路，将电容传感器的电容变化转换为可读的电信号，并传递给FPGA。

**软件编程**：

* + **信号处理**：FPGA读取电容传感器的信号，并对信号进行处理，确定触摸位置和相应的控制信号。
  + **数据传输**：将处理后的数据传输给其他模块（如蓝牙通信模块）进行进一步的处理和传输。

**9.2 FPGA数据处理测试**

编写测试程序，验证FPGA对电容检测信号的处理能力。通过LED指示灯显示处理结果，确保FPGA内部逻辑的正确性和稳定性。

FPGA数据处理测试需要考虑信号的同步和异步处理、数据的存储和传输、以及系统的整体性能和稳定性。通过编写测试程序，验证FPGA对电容检测信号的处理能力，确保FPGA内部逻辑的正确性和稳定性。FPGA控制平台是整个系统的核心。FPGA内部设计了多个功能模块，包括触摸信号处理模块、数据处理模块和UART发送模块。

1. **触摸信号处理模块**：
   * **功能**：接收并处理来自ADC的数字信号，确定触摸位置。
   * **实现**：通过Verilog或VHDL编写处理逻辑，确保信号的准确检测和处理。
2. **数据处理模块**：
   * **功能**：将触摸信号转换为预定义的指令集，这些指令可以用于控制上位机的不同操作。
   * **实现**：编写逻辑将触摸信号映射到特定的指令，比如PPT翻页或游戏控制。
3. **UART发送模块**：
   * **功能**：通过UART串口将处理后的数据发送到蓝牙模块。
   * **实现**：UART模块按照标准的串行通信协议工作，确保数据的可靠传输。

**9.3 蓝牙通信测试**

验证FPGA与蓝牙模块之间的通信，确保数据能够稳定传输。使用蓝牙接收器和终端设备，验证数据接收的准确性，并通过上位机程序处理接收到的数据。

蓝牙通信测试需要考虑数据的传输速率、传输距离和传输稳定性等因素。通过验证FPGA与蓝牙模块之间的通信，确保数据能够稳定传输。使用蓝牙接收器和终端设备，验证数据接收的准确性，并通过上位机程序处理接收到的数据，确保系统能够快速、准确地响应触控操作。蓝牙模块电路负责将FPGA处理后的数据无线传输到上位机，并接收上位机的反馈信号（如果有）。

1. **UART接收模块**：
   * **功能**：接收来自FPGA的串行数据。
   * **实现**：配置蓝牙模块的UART接口，与FPGA的UART模块匹配，确保数据的正确接收。
2. **蓝牙传输模块**：
   * **功能**：通过蓝牙协议将接收到的数据无线传输到上位机。
   * **实现**：使用蓝牙模块的内置协议栈，确保数据的稳定传输。
3. **蓝牙接收模块**：
   * **功能**：从上位机接收数据并传输回FPGA（双向通信）。
   * **实现**：配置蓝牙模块的接收功能，与上位机进行数据交互

#### 9.4触控盘控制游戏操作模块测试

触控盘控制游戏操作模块的原理是通过检测用户在触控盘上的触摸动作，将这些动作转换成相应的游戏控制信号，并通过蓝牙模块发送到上位机或游戏设备。这一过程包括以下几个步骤：

1. **触控检测**：通过触控传感器检测用户的触摸动作，并将触摸位置数据传输给FPGA。
2. **数据处理**：FPGA对触摸数据进行处理，确定触摸位置和相应的控制信号。
3. **信号传输**：FPGA通过蓝牙模块将控制信号无线传输到上位机或游戏设备，完成游戏操作。

**触控盘控制游戏操作模块实现**

触控盘控制游戏操作模块的实现包括硬件设计和软件编程两部分。硬件设计主要涉及触控传感器和蓝牙模块的连接，软件编程包括触摸数据处理和蓝牙通信的实现。

1. **硬件设计**：
   * **触控传感器**：选择合适的触控传感器模块，可以检测触摸位置，并将数据传输给FPGA。常用的触控传感器包括电容式和电阻式传感器。
   * **蓝牙模块**：选择支持UART通信的蓝牙模块，与FPGA连接，负责无线数据传输。
2. **软件编程**：
   * **触摸数据处理**：FPGA读取触控传感器的数据，并对数据进行处理，确定触摸位置和相应的游戏控制信号。
   * **蓝牙通信**：FPGA通过UART接口将处理后的控制信号发送到蓝牙模块，再由蓝牙模块无线传输到上位机。

实际操作时，我们通过首先获取从蓝牙模块发送的信号，显示为16进制数，触控板的1-8号键传出的信号为0X00—0X07,无效信号为0X0F,随后在上位机代码中对这些信号进行绑定。分别绑定为W，A,S,D,E,F,R，一共是7个键，留出一个触控为转换模式，可以切换为鼠标模式，可以控制上下左右以及左右键。这种绑定也适用于控制游戏角色。

**10. 器件清单**

1. **FPGA芯片**：Xilinx FPGA. Nexys4 DDR
2. **触控模块：8键触控板检测IC.TTP226**
3. **蓝牙模块**：HC-08
4. **电源模块**：5V/3.3V电源模块

**11. 总结及心得体会**

通过本项目，我们成功设计并实现了一个基于Xilinx FPGA的电容感应触控系统。整个系统从电容检测、数据处理到无线传输，功能完善，性能稳定。通过实际应用测试，系统能够准确响应触控操作，实现预期的功能。

**11.1 项目总结**

项目实现了以下几个主要功能：

1. **电容检测**：通过电容检测电路准确检测电容按键的触控状态，并将其转换为数字信号。
2. **数据处理**：FPGA对电容检测信号进行处理和解析，确定触控的位置和状态。
3. **数据传输**：通过串口与蓝牙芯片通信，将处理后的触控数据无线传输到上位机或终端设备。
4. **应用功能**：实现基本键盘，鼠标功能和无线游戏手柄功能。

**11.2 心得体会**

在项目的开发过程中，我们经历了从需求分析、方案设计、硬件实现到系统测试的完整过程。通过不断的调试和优化，我们最终实现了预期的功能目标。项目中遇到的主要挑战包括电容信号的准确检测、FPGA逻辑设计的复杂性和蓝牙数据传输的稳定性。通过团队的共同努力，我们克服了这些挑战，取得了令人满意的成果。

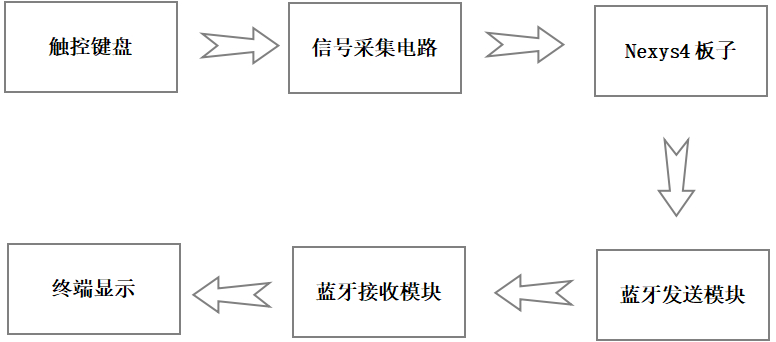
此次项目让我们深刻理解了电容触摸技术的原理和应用，掌握了FPGA逻辑设计的基本方法，积累了丰富的实践经验。同时，我们也认识到在项目开发中团队协作和持续学习的重要性。这些收获将对我们未来的学习和工作产生积极的影响。

**12. 参考资料**

1. Xilinx FPGA开发手册
2. 蓝牙通信协议（SPP）规范
3. 电容检测电路设计资料
4. 硬件描述语言（Verilog/VHDL）参考书

**13. 附录**

**13.1 电路图**



**13.2 源码清单及注释**

**模块概述**

该模块实现了一个基于触摸输入的触控数据处理系统，通过UART发送处理后的触控数据。主要由以下几个部分组成：

1. **触控数据处理模块**：处理8个触摸输入信号，输出一个4位的触控输出信号。
2. **时钟计数器**：用来控制数据发送的时间间隔。
3. **UART发送模块**：将处理后的触控数据通过UART发送出去。

module top\_module (

input wire clk,

input wire reset,

input wire [7:0] touch\_input, // 8个触摸输入引脚

output wire tx // UART输出引脚

);

wire [3:0] touch\_output;

reg [7:0] tx\_data;

reg tx\_data\_valid;

// 实例化触控数据处理模块

touch\_processor processor (

.clk(clk),

.reset(reset),

.touch\_input(touch\_input),

.touch\_output(touch\_output)

);

// 定义23位计数器，用于计数5000000个时钟周期（假设时钟频率为100MHz）

reg [22:0] clock\_counter;

parameter CLOCK\_DIVIDE = 5000000; // 对应50毫秒间隔（100MHz时钟）

// 状态机

always @(posedge clk or posedge reset) begin

if (reset) begin

clock\_counter <= 23'b0;

tx\_data <= 8'b0;

tx\_data\_valid <= 1'b0;

end else begin

if (clock\_counter == CLOCK\_DIVIDE - 1) begin

// 当计数器达到预设值时，重置计数器，并准备发送数据

clock\_counter <= 23'b0;

tx\_data <= {4'b0, touch\_output}; // 将4位触控输出扩展到8位并赋值给tx\_data

tx\_data\_valid <= 1'b1; // 有效标志置位

end else begin

// 否则继续计数

clock\_counter <= clock\_counter + 1;

tx\_data\_valid <= 1'b0; // 无效标志清零

end

end

end

// 实例化UART发送模块

uart\_tx uart (

.clk(clk),

.rst\_n(~reset),

.tx\_data(tx\_data),

.tx\_data\_valid(tx\_data\_valid),

.tx\_pin(tx)

);

Endmodule

**详细注释与说明**

1. **模块输入输出**
   * clk：系统时钟信号。
   * reset：系统复位信号，高电平有效。
   * touch\_input：8个触摸输入信号。
   * tx：UART发送引脚。
2. **内部信号与参数**
   * touch\_output：4位触控输出信号，由触控数据处理模块生成。
   * tx\_data：8位要通过UART发送的数据。
   * tx\_data\_valid：发送数据有效标志位。
   * clock\_counter：23位计数器，用于生成时间间隔信号。
   * CLOCK\_DIVIDE：定义计数器的最大值，对应于100毫秒的时间间隔（假设时钟频率为100MHz）。
3. **触控数据处理模块实例化**
   * touch\_processor：处理8个触摸输入信号，并生成4位触控输出信号。
   * clk：系统时钟信号。
   * reset：系统复位信号。
   * touch\_input：8个触摸输入信号。
   * touch\_output：4位触控输出信号。
4. **状态机**
   * 当reset信号有效时，复位计数器clock\_counter、发送数据tx\_data和数据有效标志位tx\_data\_valid。
   * 在时钟上升沿，如果计数器clock\_counter达到预设值CLOCK\_DIVIDE - 1：
     + 将计数器重置为0。
     + 将4位触控输出信号扩展为8位，并赋值给发送数据tx\_data。
     + 置位发送数据有效标志位tx\_data\_valid。
   * 否则，计数器继续计数，并清零发送数据有效标志位tx\_data\_valid。
5. **UART发送模块实例化**
   * uart\_tx：UART发送模块，用于将数据通过UART发送出去。
   * clk：系统时钟信号。
   * rst\_n：UART复位信号，低电平有效，所以连接~reset。
   * tx\_data：8位要发送的数据。
   * tx\_data\_valid：发送数据有效标志位。
   * tx\_pin：UART发送引脚。
   * 如果没有任何有效的触摸输入信号，则输出无效值4'hF。

### 模块概述

touch\_processor模块用于处理8个触摸输入信号，根据激活的触摸输入信号输出相应的4位序号信号。该模块的主要功能是检测哪个触摸输入被激活，并输出对应的序号。若没有有效触摸输入，则输出无效值。

module touch\_processor (

input wire clk, // 系统时钟信号

input wire reset, // 系统复位信号，高电平有效

input wire [7:0] touch\_input, // 8个触摸输入引脚

output reg [3:0] touch\_output // 输出激活触控信号的序号

);

always @(posedge clk or posedge reset) begin

if (reset) begin

// 当复位信号为高电平时，将触控输出重置为4'h0

touch\_output <= 4'h0;

end else begin

// 根据触摸输入信号的值确定触控输出的值

case (touch\_input)

8'b00000001: touch\_output <= 4'h0; // 触摸输入[0]被激活

8'b00000010: touch\_output <= 4'h1; // 触摸输入[1]被激活

8'b00000100: touch\_output <= 4'h2; // 触摸输入[2]被激活

8'b00001000: touch\_output <= 4'h3; // 触摸输入[3]被激活

8'b00010000: touch\_output <= 4'h4; // 触摸输入[4]被激活

8'b00100000: touch\_output <= 4'h5; // 触摸输入[5]被激活

8'b01000000: touch\_output <= 4'h6; // 触摸输入[6]被激活

8'b10000000: touch\_output <= 4'h7; // 触摸输入[7]被激活

default: touch\_output <= 4'hF; // 没有有效触摸输入，输出无效值4'hF

endcase

end

end

endmodule

**详细注释与说明**

1. **模块输入输出**
   * clk：系统时钟信号。
   * reset：系统复位信号，高电平有效。
   * touch\_input：8个触摸输入信号。
   * touch\_output：4位输出信号，用于表示激活的触控信号的序号。
2. **always 块**
   * 该always块在时钟上升沿(posedge clk)或复位信号上升沿(posedge reset)时触发。
3. **复位条件**
   * 如果reset信号为高电平，触控输出touch\_output被重置为4'h0。
4. **case 语句**
   * 在复位条件之外，根据touch\_input的值确定touch\_output的值。
   * 使用case语句来检测哪个触摸输入信号被激活，并输出相应的序号。
     + 如果touch\_input为8'b00000001，则touch\_output为4'h0。
     + 如果touch\_input为8'b00000010，则touch\_output为4'h1。
     + 如果touch\_input为8'b00000100，则touch\_output为4'h2。
     + 以此类推，直到最高位的触摸输入8'b10000000，对应的输出为4'h7。
   * 如果没有任何有效的触摸输入信号，则输出无效值4'hF。

### 模块概述

uart\_tx模块是一个简单的UART发送器。它将8位并行数据转换为串行数据，并通过UART协议发送。模块支持波特率为9600bps，输入时钟频率为100MHz。它包括四个主要状态：空闲、起始、数据传输和停止。

module uart\_tx (

input wire clk, // 系统时钟信号

input wire rst\_n, // 复位信号，低电平有效

input wire [7:0] tx\_data, // 要发送的8位数据

input wire tx\_data\_valid, // 数据有效信号

output reg tx\_pin // UART发送引脚

);

parameter CLK\_FRE = 100\_000\_000; // 100 MHz时钟频率

parameter BAUD\_RATE = 9600; // 波特率9600

localparam CYCLE = CLK\_FRE / BAUD\_RATE; // 每个波特率周期所需的时钟周期数

// 状态编码

localparam S\_IDLE = 0; // 空闲状态

localparam S\_START = 1; // 起始状态

localparam S\_DATA = 2; // 数据传输状态

localparam S\_STOP = 3; // 停止状态

reg [1:0] state; // 状态寄存器

reg [3:0] bit\_count; // 数据位计数器

reg [15:0] clk\_count; // 时钟计数器

reg [7:0] data; // 数据寄存器

always @(posedge clk or negedge rst\_n) begin

if (!rst\_n) begin

// 复位信号有效时，初始化所有寄存器

state <= S\_IDLE;

bit\_count <= 0;

clk\_count <= 0;

tx\_pin <= 1'b1;

end else begin

case (state)

// 空闲状态

S\_IDLE: begin

if (tx\_data\_valid) begin

state <= S\_START; // 如果数据有效，进入起始状态

data <= tx\_data; // 存储要发送的数据

end

tx\_pin <= 1'b1; // 发送引脚保持高电平

end

// 起始状态

S\_START: begin

if (clk\_count < CYCLE) begin

clk\_count <= clk\_count + 1; // 计数器增加

end else begin

clk\_count <= 0; // 计数器重置

state <= S\_DATA; // 进入数据传输状态

tx\_pin <= 1'b0; // 起始位（低电平）

end

end

// 数据传输状态

S\_DATA: begin

if (clk\_count < CYCLE) begin

clk\_count <= clk\_count + 1; // 计数器增加

end else begin

clk\_count <= 0; // 计数器重置

if (bit\_count < 8) begin

tx\_pin <= data[bit\_count]; // 发送数据位

bit\_count <= bit\_count + 1; // 位计数器增加

end else begin

bit\_count <= 0; // 位计数器重置

state <= S\_STOP; // 进入停止状态

end

end

end

// 停止状态

S\_STOP: begin

if (clk\_count < CYCLE) begin

clk\_count <= clk\_count + 1; // 计数器增加

end else begin

clk\_count <= 0; // 计数器重置

state <= S\_IDLE; // 回到空闲状态

tx\_pin <= 1'b1; // 停止位（高电平）

end

end

endcase

end

end

endmodule

**详细注释与说明**

1. **参数声明**
   * CLK\_FRE: 系统时钟频率，设置为100MHz。
   * BAUD\_RATE: UART波特率，设置为9600bps。
   * CYCLE: 每个波特率周期所需的时钟周期数，用于控制发送数据的速度。
2. **状态声明**
   * S\_IDLE: 空闲状态，等待数据有效信号。
   * S\_START: 起始状态，发送起始位（低电平）。
   * S\_DATA: 数据传输状态，逐位发送8位数据。
   * S\_STOP: 停止状态，发送停止位（高电平）。
3. **寄存器声明**
   * state: 当前状态寄存器。
   * bit\_count: 数据位计数器，用于跟踪已经发送的位数。
   * clk\_count: 时钟计数器，用于生成所需的波特率。
   * data: 数据寄存器，存储要发送的8位数据。
4. **always块**
   * 在时钟上升沿(posedge clk)或复位信号下降沿(negedge rst\_n)时触发。
   * 复位信号有效时，将所有寄存器初始化。
   * 根据当前状态，执行相应操作：
     + 在S\_IDLE状态，如果数据有效信号tx\_data\_valid为高电平，则进入S\_START状态，并将要发送的数据存储在data寄存器中，同时将发送引脚tx\_pin保持高电平。
     + 在S\_START状态，等待一个波特率周期（CYCLE个时钟周期）后，进入S\_DATA状态，并将发送引脚tx\_pin设为低电平，表示起始位。
     + 在S\_DATA状态，逐位发送数据位。每发送一位数据，等待一个波特率周期（CYCLE个时钟周期），直到所有8位数据发送完毕，然后进入S\_STOP状态。
     + 在S\_STOP状态，等待一个波特率周期（CYCLE个时钟周期）后，回到S\_IDLE状态，并将发送引脚tx\_pin设为高电平，表示停止位。