

（深圳）

实验报告

开课学期： 2024秋季

课程名称：数字逻辑设计（实验）

实验名称： 综合实验

实验性质： 综合设计型

实验学时： 6 地点： T2506

学生班级： 计算机与电子通信3班

学生学号： 2023311323

学生姓名： 咸浩洋

评阅教师：

报告成绩：

实验与创新实践教育中心制

2024年10月

|  |
| --- |
| 设计的功能描述 |
| 概述基本功能  **1. top.v**   * **功能**：顶层模块，连接其他子模块。管理发送和接收模块的输入输出。   **2. led.v**   * **功能**：控制LED状态，接收串行输入并更新显示数据。   **3. led\_display.v**   * **功能**：根据输入数据和LED使能信号控制LED显示。   **4. uart\_recv.v**   * **功能**：实现UART接收功能，从串行输入中接收数据并在接收完成后输出有效信号。   **5. sw\_cnt.v**   * **功能**：将8个开关的状态转换为输出，表示为8位数据。   **6. sw\_display.v**   * **功能**：处理开关输入并在检测到有效信号时将数据传输到UART发送模块。   **7. uart\_send.v**   * **功能**：实现UART发送功能，将8位数据通过串行方式发送。模块在状态机控制下运行，分为IDLE、START、DATA和STOP四个状态。 |
| 系统设计 |
| 用硬件框图描述系统主要功能及各模块之间的相互关系  **1. top**   * **功能**：顶层模块，连接各个子模块。负责处理输入输出信号的管理 * **数据流动**：将输入信号传递给led和sw\_display模块，将其输出连接到LED显示和数据发送   **2. led**   * **功能**：控制LED的状态，处理从uart\_recv接收的数据并更新LED显示 * **数据流动**：接收UART数据，更新内部数据和LED计数，最终将数据传递给display模块以进行可视化显示   **3. led\_display**   * **功能**：显示数据到LED，通过将数据转换为LED显示格式 * **数据流动**：根据输入数据和LED使能信号，更新LED状态   **4. uart\_recv**   * **功能**：实现UART串行数据接收。接收串行输入信号din，并在接收到完整数据后将其输出为8位数据 * **数据流动**：通过状态机监测输入信号。接收到起始位后，计数时钟以采样数据位，最后触发valid信号，输出接收到的数据   **5. sw\_cnt**   * **功能**：计数开关输入，输出当前开关状态 * **数据流动**：每次时钟上升沿检查开关状态，更新输出out   **6. sw\_display**   * **功能**：处理开关输入，并在按下S3时将开关状态发送到uart\_send模块 * **数据流动**：将8位开关输入sw中的状态传递给uart\_send模块，valid信号指示数据有效性   **7. uart\_send**   * **功能**：实现UART串行数据发送。接收时钟、复位、有效信号和待发送数据，将数据转换为串行形式输出 * **数据流动**：当valid信号为高时，模块开始发送data。数据通过状态机控制发送过程，包括起始位、数据位（8位）和停止位。输出为dout |
| 模块设计与实现 |
| 包括各子模块设计思路，输入、输出端口及关键代码  **1. 顶层模块 top**  **功能**  top 模块是整个设计的顶层模块，负责连接各个子模块并处理输入输出信号。它接收时钟信号 clk 和复位信号 rst，并将输入信号传递给相应的子模块。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + S3: 开关信号   + sw[7:0]: 8位开关输入   + din: 数据输入信号 * **输出端口**:   + dout: 数据输出信号   + led\_en[7:0]: LED 使能信号   + led[7:0]: LED 状态信号   **关键代码**    **2. 模块 led**  **功能**  led 模块负责接收数据输入 din，处理数据并控制 LED 使能信号和状态信号。它将输入数据转换为 LED 显示的内容，并通过状态信号控制 LED 的亮灭。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + din: 数据输入信号 * **输出端口**:   + led\_en[7:0]: LED 使能信号   + led[7:0]: LED 状态信号   **关键代码**    **3. 模块 display**  **功能**  display 模块负责将输入的数据转换为适合于 LED 显示的格式。它根据 LED 计数决定哪些 LED 需要显示，并根据输入数据生成相应的 LED 状态信号。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + data[31:0]: 输入数据，包含要显示的内容   + led\_cnt[7:0]: LED 计数，指示哪些 LED 需要被激活 * **输出端口**:   + led\_en[7:0]: LED 使能信号，指示哪个 LED 被使能   + led[7:0]: LED 状态信号，表示当前显示的内容   **主要实现逻辑**   1. **数据转换**: 使用 case 语句将 4 位输入数据映射到对应的 LED 显示格式。 2. **LED 使能**: 通过 led\_cnt 控制哪些 LED 被激活。 3. **时间控制**: 通过计数器 time\_cnt 和信号 time\_end 来控制 LED 的轮换显示。 4. **状态更新**: 在时钟上升沿更新 LED 显示和使能信号。   **关键代码**    **4. 模块 uart\_recv**  **UART接收模块**  **功能**  uart\_recv 模块负责接收 UART 数据。它通过检测输入信号 din 的变化，识别起始位、数据位和停止位。该模块在接收到完整的数据字节后，输出有效信号 valid 和接收的数据 data。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + din: UART 数据输入信号 * **输出端口**:   + valid: 数据有效信号，指示接收到的数据有效   + data[7:0]: 接收到的数据字节   **主要实现逻辑**   1. **状态机**: 模块采用状态机实现，主要有四个状态：IDLE、START、DATA 和 STOP。根据输入信号和时钟计数器的值进行状态转换。 2. **时钟计数**: 使用 clk\_cnt 变量来计数时钟周期，以确保在正确的时钟边缘读取数据位。 3. **数据接收**: 在 DATA 状态下，接收每个数据位，并在 STOP 状态下发出有效信号。   **关键代码**    **5. 模块 sw\_cnt**  **功能**  sw\_cnt 模块的主要功能是读取 8 个开关的状态，并根据这些状态生成一个 8 位的输出信号。每个输出位对应一个开关的状态，若开关被按下（为 1），则输出相应位为 1；否则输出为 0。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + sw[7:0]: 8 个开关的状态输入 * **输出端口**:   + out[7:0]: 开关状态的输出信号     **6. 模块 sw\_display**  **功能**  sw\_display 模块的主要功能是读取开关状态，并通过 UART 发送这些状态。在此过程中，模块还实现了对按键 S3 的去抖动处理，以确保可靠的数据采集。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + S3: 按键输入信号   + sw[7:0]: 8 个开关的状态输入 * **输出端口**:   + dout: UART 发送的数据输出   **主要实现逻辑**   1. **按键去抖动**: 使用定时器 timer\_div\_10ms 来去抖动按键 S3，避免因开关抖动导致的误触发。定时器计数到 10 毫秒后，更新 limit\_div\_10ms 使其有效。 2. **状态更新**: 在 limit\_div\_10ms 触发时，如果 S3 按下，则将当前开关状态 sw\_in 赋值，并设置 valid 为 1，指示数据有效。 3. **开关状态读取**: 使用 sw\_cnt 模块读取开关状态，将结果传递给 UART 发送模块 uart\_send。   **关键代码**    **7. 模块 uart\_send**  **功能**  uart\_send 模块负责将数据通过 UART 协议发送。它实现了从 IDLE 状态到 START、DATA 和 STOP 状态的状态机，并根据设定的波特率控制发送的时序。  **输入输出端口**   * **输入端口**:   + clk: 时钟信号   + rst: 复位信号   + valid: 数据有效信号，指示何时发送数据   + data[7:0]: 要发送的 8 位数据 * **输出端口**:   + dout: UART 数据输出信号   **主要实现逻辑**   1. **状态机**: 模块使用一个状态机来管理数据发送的各个阶段，包括 IDLE、START、DATA 和 STOP。根据 valid 信号和波特率计数器的状态进行状态转移。 2. **波特率控制**: 通过 baud\_counter 控制发送速率。BAUD\_TICKS 根据系统时钟频率和设定的波特率计算得到。 3. **数据发送**: 在 DATA 状态下，从 data\_to\_send 中逐位发送数据。每次发送一位后，更新 bit\_count，直到发送完 8 位数据后进入 STOP 状态。 4. **输出控制**: 控制 dout 信号的状态，确保在不同的状态下输出正确的 UART 信号。   **关键代码**        UART接收模块：   1. **IDLE**: 等待接收数据的状态，监测输入信号 din 是否为低电平（表示开始位）。 2. **START**: 检测到开始位后，进入此状态，并开始计时，准备接收数据位。 3. **DATA**: 逐位接收数据，直到接收完整的8位数据。 4. **STOP**: 接收完数据后，等待停止位，并确认接收完成   **复位逻辑**: 当 rst 信号为高时，所有寄存器和状态被重置。  **状态转移**:   * **1. IDLE 状态** * **输入**: din = 1（表示线路空闲） * **现态**: state = IDLE * **次态**: 若din变为0，则转移到 START 状态。 * **输出**: valid = 0，data = 0 * **2. START 状态** * **输入**: din = 0（检测到起始位） * **现态**: state = START * **次态**: 当 clk\_cnt 达到 bit\_clk\_cnt，转移到 DATA 状态。 * **输出**: 在此状态没有数据输出，bit\_pos 计数器重置为 0。 * **3. DATA 状态** * **输入**: 连续接收数据位 * **现态**: state = DATA * **次态**: * 每接收一个数据位，bit\_pos 加 1； * 若 bit\_pos 达到 7（接收完8位数据），则转移到 STOP 状态。 * **输出**: * 在 clk\_cnt 达到 mid\_bit\_clk\_cnt 时，data[bit\_pos] <= din，接收数据位。 * valid 在数据接收完成后会在 STOP 状态中被设置为1。 * **4. STOP 状态** * **输入**: 接收到停止位 * **现态**: state = STOP * **次态**: * 当 clk\_cnt 达到 mid\_bit\_clk\_cnt 后，将 valid 设置为 1； * clk\_cnt 达到 bit\_clk\_cnt 后，转移回 IDLE 状态。 * **输出**: valid = 1（表示接收到有效数据）。   **顶层模块的RTL分析原理图截图** |
| 调试报告 |
| 仿真波形截图及仿真分析 |
| 设计过程中遇到的问题及解决方法 |
| **实验三 数码管高频轮询显示问题分析**  **现象**  在使用数码管进行高频轮询显示时，显示出现闪烁，数码管的显示内容无法稳定，时而显示正确的值，时而出现显示错误  **分析过程**   1. 发现控制信号频繁变动，且显示的数值并不稳定。 2. 检查了控制数码管的代码显示更新和时间计数的部分。 3. 通过调整时钟频率和轮询逻辑，发现闪烁问题是段选和位选更新逻辑错误导致的   **错误原因**  led\_en 和 led\_cx 的更新逻辑在复位后没有及时清除，导致在复位状态下也进行闪烁显示  **解决方法**  1.优化时间计数逻辑，确保时间计数器的更新与显示更新逻辑一致。  2.设置一个 data 数组存储待显示的数据。  3.修正 led\_en 的初始化和更新，每个循环或者是在复位后，led\_en 和led\_cx 应立即还原，并在每次时间结束后稳健地轮询  **修改后的代码** |
| 课程设计总结 |
| 完成本实验所用小时数：约12 h  其中写代码的小时数：约9 h  写报告的小时数；约3 h  **课程收获**  **1. 硬件描述语言** 理解Verilog语言的基本语法和结构，能够编写模块化的代码  **2. 项目设计与实现** 通过实际项目（计数器，数码管显示、UART发送与接收等），提升了代码设计能力  **3. 调试能力** 学习了如何使用仿真工具对设计进行验证，能够有效地找出并修复代码中的错误，通过实际测试，增强了对硬件行为的理解  **课程建议**  **好**  指导书的引导感觉不够，代码没有框架完全是从零开始手搓，课下耗时很多  （但是因为我已经做完了，所以建议给下届的同学再多加内容） |