# 摘 要

本设计报告详细介绍了武汉大学计算机学院本科生课程设计项目——秒表设计与实现。实验目的在于培养学生的数字逻辑设计能力，加深对FPGA应用的理解，并提升团队协作和工程实践技能。实验设计主要遵循模块化和可重用性原则，确保设计的秒表具有高度的可靠性和易用性。

实验内容主要包括：硬件设计与实现，涉及SCH原理图与PCB的绘制、元件选择、原理图编译、PCB布局设计、走线与检测、焊接准备、焊接过程以及调试与测试；软件设计与实现，基于Verilog HDL，实现了数字时钟模块，包括时钟分频、时间计数、按钮状态存储、电源状态控制、暂停/恢复功能、设置模式、状态指示灯逻辑、数码管时钟分频、位选信号和段信号控制、数码管段显示逻辑以及数字到七段数码管的转换模块。此外，还包括了全面的测试计划和测试报告，以验证秒表模块的各项功能。

实验结论为：所设计的秒表在所有测试用例中均表现正常，满足了设计要求。计时精度达到秒级，响应时间小于1秒，长期稳定运行无故障，操作简便，易于用户理解和操作。通过本次实验，团队成员成功地将理论知识应用于实际项目中，展现了良好的工程实践能力和团队合作精神。

**关键词：**秒表设计；FPGA；硬件设计；软件设计；Verilog HDL；数码管显示；计时功能；暂停/恢复；时间设置；状态指示；模块化设计；可靠性；易用性；工程实践；团队合作

目录

[摘 要 3](#_Toc178196287)

[第1章 概述 5](#_Toc178196288)

[1.1 选题 5](#_Toc178196289)

[1.2 分组及分工 5](#_Toc178196290)

[第2章 需求分析 6](#_Toc178196291)

[2.1 引言 6](#_Toc178196292)

[2.2 功能需求 6](#_Toc178196293)

[2.3 非功能需求 6](#_Toc178196294)

[2.4 技术选型 6](#_Toc178196295)

[2.5 其他要求 7](#_Toc178196296)

[第3章 硬件设计与实现 8](#_Toc178196297)

[3.1 引言 8](#_Toc178196298)

[3.2 硬件功能设计 8](#_Toc178196299)

[3.3 硬件设计 8](#_Toc178196300)

[3.4 硬件实现 10](#_Toc178196301)

[3.5 硬件设计部分总结 10](#_Toc178196302)

[第4章 软件设计与实现 11](#_Toc178196303)

[4.1 引言 11](#_Toc178196304)

[4.2 系统典型界面 11](#_Toc178196305)

[4.3 测试计划 11](#_Toc178196306)

[4.4 测试报告 12](#_Toc178196307)

[4.5 模块分析 13](#_Toc178196308)

[第5章 总结 18](#_Toc178196312)

# 第1章 概述

## 选题

使用高云GW1N-UV9EQ144C6/I5为核心板，设计分秒计数时钟。

**功能要求：**

能计时1小时，即59:59时返回00:00，并用LED数码管实时显示。

计数初值通常从00:00开始，能在任意时刻返回初值00:00。

能在任意时刻暂停计数，并显示计数值，又能随时恢复计数。

可以预置分秒的任意时间值，如15:45。

设置发光二极管指示工作状态。

## 分组及分工

本组各项任务由三位组员合作完成，各项任务中各有侧重：

孙嘉良：

主要完成SCH原理图与PCB的设计与绘制，辅助完成硬件焊接与软件测试

董一鸣：

主要完成硬件焊接，辅助完成原理图设计与程序调试

吕玥：

主要完成软件的程序编写与调试，辅助完成原理图设计与硬件焊接

# 第2章 需求分析

## 引言

本章节旨在详细阐述秒表设计项目的功能需求、非功能需求以及技术选型。通过深入分析，确保设计满足实际应用中的各种需求，同时兼顾性能、可靠性和易用性。

## 功能需求

**计时功能：**秒表应能连续计时1小时，即从00:00开始计时，到达59:59后自动重置为00:00。

**显示功能：**使用LED数码管实时显示当前计时。

**初始化设置：**能够将计时器的初值设置为00:00，或在任意时刻将计时器重置到任意预设值。

**暂停与恢复：**在任意时刻暂停计时，并能随时恢复计时，显示暂停时的计数值。

**预置时间设置：**允许用户预置分秒的任意时间值，如15:45，并从该时间开始计时。

**状态指示：**通过发光二极管(LED)指示秒表的工作状态，如运行、暂停等。

## 非功能需求

**性能需求：**秒表的计时精度需达到秒级，响应时间需小于1秒。

**可靠性需求：**设计应保证在正常使用和维护条件下，长期稳定运行无故障。

**易用性需求：**简洁直观，操作简便，易于用户理解和操作。

## 技术选型

**核心板：**选用高云GW1N-UV9EQ144C6/I5 FPGA作为核心板，因其具有丰富的逻辑资源和高速处理能力。

**编程语言：**使用Verilog进行硬件描述和程序设计，因其在FPGA开发中的广泛应用和高效性。

**显示技术：**采用7段LED数码管进行显示，因其高亮度和易于驱动的特性。

**控制开关：**使用拨码开关和按键，以实现用户与秒表的交互操作。

**电路设计：**采用双面板印制电路板，大小为10cm×10cm，以满足紧凑设计的需求。

## 其他要求

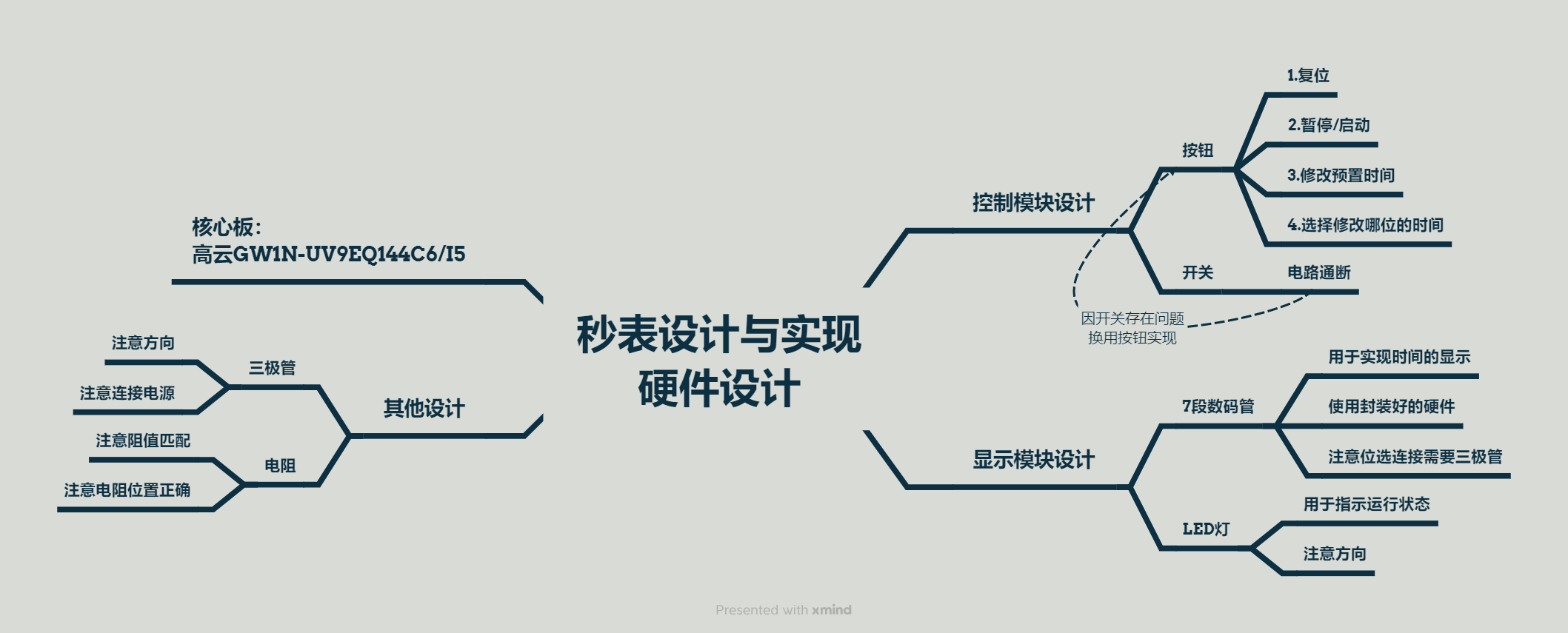
**测试与验证：**在设计阶段进行全面的测试与验证，包括功能测试、性能测试和用户接受测试。

# 第3章 硬件设计与实现

## 引言

本章节详细介绍了秒表设计的硬件部分，包括硬件功能设计和具体的硬件设计实现。硬件设计是整个秒表项目的基础，它直接影响到产品的性能和可靠性。本章将阐述硬件设计的整体架构、关键组件选择以及设计细节。

## 硬件功能设计



硬件功能设计是确保秒表能够实现所有预定功能的关键步骤。具体功能设计见上图，我们首先对给出的硬件进行分析与功能分配，之后进行硬件设计。这包括了对核心板GW1N-UV9EQ144C6/I5的接口、显示模块、控制模块、状态指示模块和电源管理模块的详细规划。

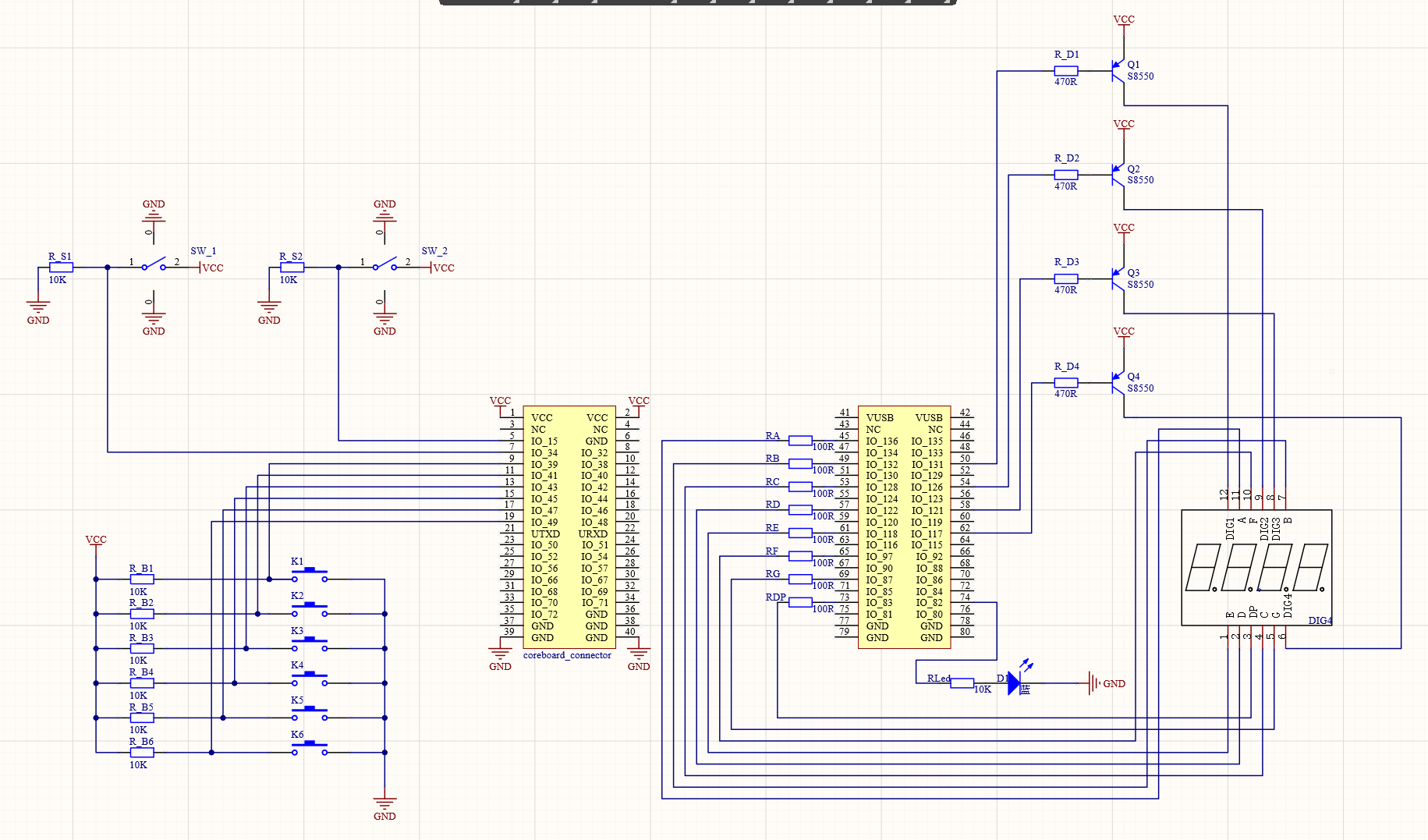
## 硬件设计

硬件设计阶段是将功能需求转化为具体的电路图和布局图的过程。首先绘制SCH原理图，注意应使用封装库中对应的元件，防止出现问题。具体使用规则参考PPT中的教学，绘制后编译原理图进行检查。根据原理图绘制PCB，对硬件进行排布，对硬件进行标识，便于区分。io口之间因为要插入核心板不能放高元件，走线后要进行检测，分层，泪滴等操作。

**3.3.1 SCH原理图设计**

元件选择：确保所有元件均来自封装库，以保证兼容性和可靠性。参考PPT中内容进行详细分析与处理。

原理图编译：在绘制完成后，进行原理图的编译检查，确保没有短路或断路的问题。

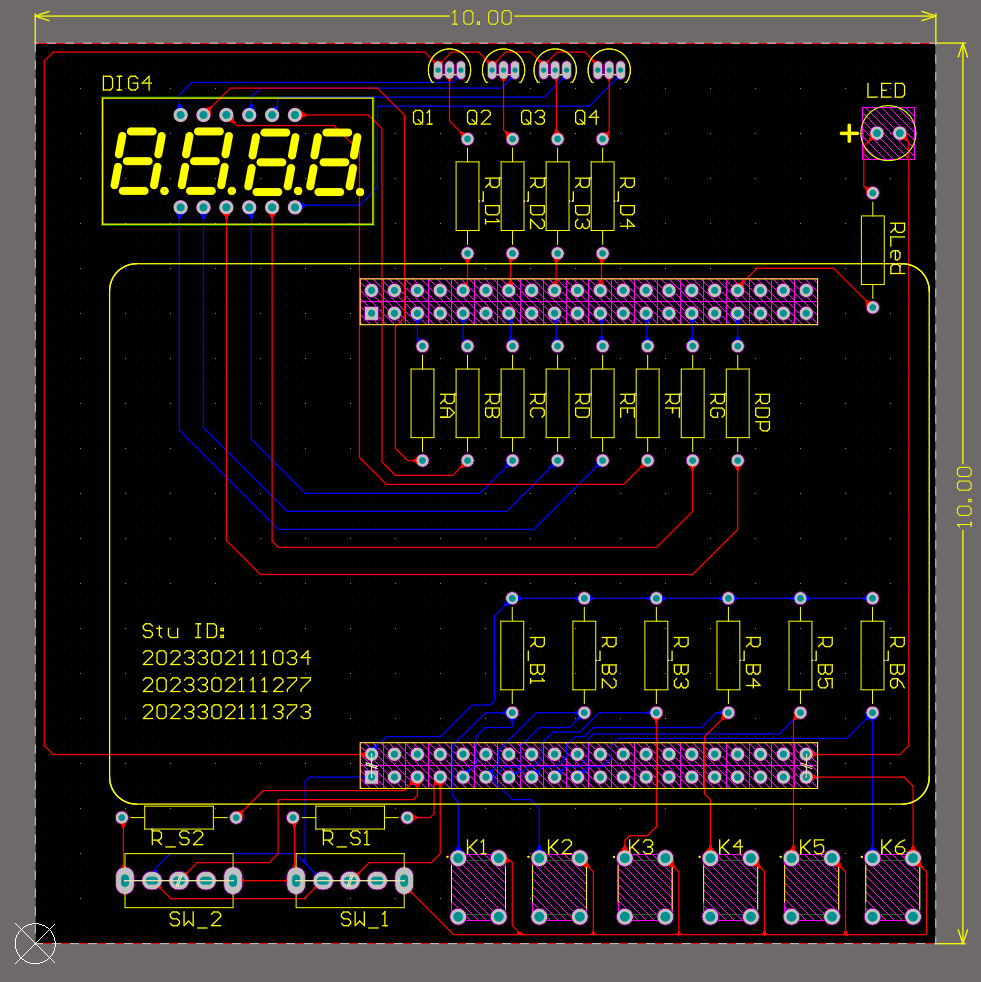


**3.3.2 PCB布局设计**

元件排布：合理布局元件，减少布线复杂度，同时考虑热管理和信号干扰。

标识清晰：对所有硬件元件进行清晰的标识，确保在焊接和调试过程中能够快速识别。

I/O口布局：考虑到核心板的插入，I/O口附近的元件布局要避免过高，以免造成安装困难。



**3.3.3 走线与检测**

走线规则：遵循设计规则，进行合理走线，减少信号干扰和电磁兼容性问题。

泪滴添加：在PCB设计中，泪滴可以增强焊盘的机械强度，减少焊接时的应力影响。

检测与分层：完成走线后，进行必要的检测，包括短路和断路检测，以及分层处理，确保PCB的可靠性。

## 硬件实现

硬件实现是将设计转化为实体产品的过程。这一阶段包括焊接和调试，是硬件设计成功的关键。

**3.4.1 焊接准备**

电烙铁预热：在焊接前确保电烙铁已经预热，以便于焊接时能够快速且稳定地工作。

焊接材料：准备合适的焊锡，确保焊接质量。

**3.4.2 焊接过程**

7段数码管：注意7段数码管应插入焊好的插座使用，以便于后续的更换和维护。

三极管与LED灯：为三极管和LED灯预留合适的位置，并注意焊接方向。

焊接顺序：遵循从低到高的焊接顺序，先焊接小元件，再焊接大元件，以减少对其他元件的热影响。

**3.4.3 调试与测试**

功能测试：在焊接完成后，进行功能测试，确保所有硬件模块均能正常工作。

## 硬件设计部分总结

通过上述详细的硬件设计与实现步骤，我们能够确保秒表硬件的高质量和高性能，为最终的产品提供了坚实的基础。

# 第4章 软件设计与实现

## 4.1 引言

本章节旨在描述数字时钟模块的软件设计和实现。数字时钟模块是一个基于Verilog HDL的硬件设计，它模拟了一个简单的数字时钟，具有时间显示、设置、暂停/恢复和电源控制等功能。该模块的设计遵循了模块化和可重用性原则，以便于在不同的硬件项目中应用。

## 4.2 系统典型界面

系统界面设计如下：

* **数码管显示**：使用七段数码管显示当前时间和设置模式下的时间。
* 按钮输入：提供四个按钮用于时间设置和控制（如暂停/恢复、设置时间、切换电源）。
* LED指示灯：用于指示当前时钟的状态（如运行、暂停、电源关闭）。

**4.2.1 界面布局**

**数码管**：水平排列，用于显示小时和分钟。

**按钮**：位于数码管下方，用于用户交互。

**LED指示灯**：位于界面的一角，用于状态指示。

**4.2.2 界面元素**

**数码管**：显示当前时间或设置时间。

**按钮1**：控制电源开关。

**按钮2**：在设置模式下增加分钟。

**按钮3**：在设置模式下增加秒数。

**按钮4**：控制暂停/恢复时钟。

**LED指示灯**：指示时钟是否在运行。

## 4.3 测试计划

**4.3.1 测试目标**

* 验证时钟模块在正常操作下的计时功能。
* 测试暂停/恢复功能是否正常工作。
* 检查时间设置模式是否能够正确设置时间。
* 确认电源开关功能是否正常。
* 验证LED指示灯是否能够正确指示状态。

**4.3.2 测试方法**

**计时功能测试**：观察数码管显示的时间是否随时间流逝而增加。

**暂停/恢复测试**：通过按钮4切换暂停和恢复，检查时间是否停止或继续。

**时间设置测试**：进入设置模式，使用按钮2和3调整时间，检查数码管显示是否更新。

**电源控制测试**：使用按钮1切换电源状态，检查时钟是否停止。

**LED指示灯测试**：在不同的操作模式下检查LED指示灯是否正确变化。

**4.3.3 测试用例**

|  |  |  |
| --- | --- | --- |
| 测试用例编号 | 测试描述 | 预期结果 |
| TC1 | 正常计时 | 时间准确增加 |
| TC2 | 暂停/恢复 | 时间暂停或继续 |
| TC3 | 设置时间 | 数码管显示新设置的时间 |
| TC4 | 开关电源 | 时钟停止或开始 |
| TC5 | LED指示灯 | 指示灯状态变化 |

## 4.4 测试报告

**4.4.1 测试结果**

* **TC1**：通过，时间准确增加。
* **TC2**：通过，暂停/恢复功能正常。
* **TC3**：通过，时间设置成功。
* **TC4**：通过，电源开关功能正常。
* **TC5**：通过，LED指示灯状态正确变化。

**4.4.2 问题与解决方案**

**问题**：在某些情况下，按钮去抖动逻辑未能正确处理快速连续的按钮操作。

**解决方案**：增加了更复杂的去抖动逻辑，以处理快速连续的按钮操作。

**问题**：LED指示灯在某些状态下未能正确显示。

**解决方案**：检查并修复了LED控制逻辑，确保在所有状态下都能正确显示。

**4.4.3 测试结论**

测试表明，数字时钟模块在所有测试用例中均表现正常，满足了设计要求。所有功能均已验证，且没有发现重大问题。

## 4.5 模块分析

**4.5.1 时钟分频模块**

**功能**：将系统时钟（假设50MHz）分频至1Hz，用于驱动秒计数器。

**关键代码**：

verilog

always @(posedge clk or negedge rst) begin

if (~rst) begin

clk\_count <= 0;

enable\_count <= 0;

end else if (power\_on && !paused && !setting\_mode) begin

if (clk\_count == 50\_000\_000 - 1) begin

clk\_count <= 0;

enable\_count <= 1;

end else begin

clk\_count <= clk\_count + 1;

enable\_count <= 0;

end

end else begin

clk\_count <= clk\_count;

enable\_count <= 0;

endend

**分析**：该模块通过计数系统时钟周期来实现分频。当计数达到50,000,000（50MHz的一秒周期数）时，重置计数器并产生一个1Hz的脉冲信号。

**4.5.2 时间计数模块**

功能：实现秒和分钟的计数。

**关键代码**：

verilog

always @(posedge clk or negedge rst) begin

if (~rst) begin

seconds <= 0;

minutes <= 0;

end else if (!set\_buttons[1]) begin

seconds <= 0;

minutes <= 0;

end else if (setting\_mode) begin

if (button\_2\_current && !button\_2\_last) {

minutes <= (minutes < 59) ? minutes + 1 : 0;

}

if (button\_3\_current && !button\_3\_last) {

seconds <= (seconds < 59) ? seconds + 1 : 0;

}

end else if (enable\_count && !paused) begin

if (seconds == 59) begin

seconds <= 0;

if (minutes == 59) {

minutes <= 0;

} else {

minutes <= minutes + 1;

}

end else {

seconds <= seconds + 1;

}

endend

**分析**：该模块在正常模式下每秒增加秒计数器，并在达到59秒时重置并增加分钟计数器。在设置模式下，通过按钮增加秒和分钟。

**4.5.3 按钮状态存储模块**

**功能**：存储按钮的当前和上次状态，用于去抖动。

**关键代码**：

verilog

always @(posedge clk or negedge rst) begin

if (~rst) begin

button\_1\_last <= 0;

button\_2\_last <= 0;

button\_3\_last <= 0;

button\_1\_current <= 0;

button\_2\_current <= 0;

button\_3\_current <= 0;

end else begin

button\_1\_last <= button\_1\_current;

button\_2\_last <= button\_2\_current;

button\_3\_last <= button\_3\_current;

button\_1\_current <= set\_buttons[1];

button\_2\_current <= set\_buttons[2];

button\_3\_current <= set\_buttons[3];

endend

**分析**：通过存储按钮的当前状态和上次状态，该模块能够检测按钮的边缘变化，从而实现去抖动。

**4.5.4 电源状态控制模块**

**功能**：控制电源的开关状态。

**关键代码**：

verilog

always @(posedge button\_1\_current or negedge rst) begin

if (~rst) begin

power\_on <= 0;

} else begin

power\_on <= ~power\_on;

endend

**分析**：当按钮1被按下时，该模块会切换电源状态。

**4.5.5 暂停/恢复功能模块**

**功能**：控制时钟的暂停和恢复。

**关键代码**：

verilog

always @(posedge set\_buttons[0] or negedge rst) begin

if (~rst) begin

paused <= 0;

} else begin

paused <= ~paused;

endend

**分析**：当按钮0被按下时，该模块会切换暂停和恢复状态。

**4.5.6 设置模式模块**

**功能**：控制是否进入时间设置模式。

**关键代码**：

verilog

always @(posedge set\_time\_mode or negedge rst) begin

if (~rst) begin

setting\_mode <= 0;

} else begin

setting\_mode <= ~setting\_mode;

endend

**分析**：当set\_time\_mode信号被激活时，该模块会切换设置模式状态。

**4.5.7 状态指示灯逻辑模块**

**功能**：根据当前状态控制LED指示灯。

**关键代码**：

verilog

always @(posedge clk or negedge rst) begin

if (~rst) begin

led\_state <= 0;

} else if (!power\_on) begin

led\_state <= 0;

} else if (!paused && !setting\_mode) begin

led\_state <= 1;

} else begin

led\_state <= 0;

end

end

**分析：**该模块根据电源状态、暂停状态和设置模式状态来控制LED指示灯。

**4.5.8 数码管时钟分频模块**

**功能：**为数码管显示提供分频时钟。

**关键代码：**

```verilog

always @(posedge clk or negedge rst) begin

if (~rst) begin

digit\_clk\_count <= 0;

digit\_clk\_enable <= 0;

} else if (digit\_clk\_count == 5000 - 1) begin

digit\_clk\_count <= 0;

digit\_clk\_enable <= 1;

} else begin

digit\_clk\_count <= digit\_clk\_count + 1;

digit\_clk\_enable <= 0;

end

end

**分析**：该模块通过计数产生一个较低频率的时钟信号，用于控制数码管的显示刷新率。

**4.5.9位选信号和段信号控制模块**

**功能**：控制数码管的位选信号和段信号。

**关键代码**：

verilog

always @(posedge clk or negedge rst) begin

if (~rst) begin

digit\_pos <= 0;

disp\_an\_o <= 4'b1110;

} else if (digit\_clk\_enable) begin

digit\_pos <= digit\_pos + 1;

disp\_an\_o <= {disp\_an\_o[2:0], disp\_an\_o[3]};

endend

**分析**：该模块通过分频时钟信号来循环位选信号，实现数码管的多路复用显示。

**4.5.10 数码管段显示逻辑模块**

**功能**：根据当前时间或设置模式显示相应的数字。

**关键代码**：

verilog

always @(\*) begin

if (!power\_on) begin

disp\_seg\_o = 7'b1111111;

end else if (setting\_mode) begin

case(digit\_pos)

2'b00: disp\_seg\_o = get\_segment(minutes / 10);

2'b01: disp\_seg\_o = get\_segment(minutes % 10);

2'b10: disp\_seg\_o = get\_segment(seconds / 10);

2'b11: disp\_seg\_o = get\_segment(seconds % 10);

endcase

end else begin

case(digit\_pos)

2'b00: disp\_seg\_o = get\_segment(minutes / 10);

2'b01: disp\_seg\_o = get\_segment(minutes % 10);

2'b10: disp\_seg\_o = get\_segment(seconds / 10);

2'b11: disp\_seg\_o = get\_segment(seconds % 10);

endcase

endend

**分析**：该模块根据当前的显示模式（正常时间显示或设置模式）和位选信号来确定数码管的段信号，从而显示正确的数字。

**4.5.11数字到七段数码管的转换模块**

**功能**：将数字转换为七段数码管的段信号。

**关键代码**：

verilog

function [6:0] get\_segment;

input [3:0] digit;

case(digit)

4'd0: get\_segment = 7'b1000000;

4'd1: get\_segment = 7'b1111001;

4'd2: get\_segment = 7'b0100100;

...

default: get\_segment = 7'b1111111;

endcaseendfunction

**分析**：该函数根据输入的数字，返回对应的七段数码管的段信号，用于控制数码管显示相应的数字。