数字电子时钟 Verilog 设计文档

目录

- 数字电子时钟 Verilog 设计文档
  - 目录
  - 1. 项目概述
  - 2. 设计要求
  - 3. 系统架构
  - 4. 模块详细说明
    - 4.1 `digital_clock.v`
      - `digital_clock` 模块
    - 4.2 `control_fsm.v`
      - `control_fsm_simplified` 模块
    - 4.3 `time_counter.v`
      - `time_counter` 模块
    - 4.4 `alarm_controller.v`
      - `alarm_controller` 模块
    - 4.5 `display_controller.v`
      - `display_controller` 模块
    - 4.6 `sound_controller.v`
      - `sound_controller` 模块
  - 5 小组成员分工
    -
    -
    -
    -

1. 项目概述

本项目使用 Verilog HDL 实现了一个多功能数字电子时钟系统。该系统具备时间显示（时、分、秒）、时间设置、闹钟时间设置、闹钟触发报警以及整点报时功能。设计采用了模块化的方法，将不同的功能单元分解到独立的 Verilog 文件和模块中。

2. 设计要求

● 课设要求基本功能：

- 实现24小时制时钟功能；
- 实现整点报时功能；
- 实现时间设置功能，并且可以随意切换设置和正常工作模式。

● 项目实现附加功能：

- 实现设置状态时闪烁显示；
- 实现闹钟功能；

3. 系统架构

系统采用自顶向下的设计方法，包含一个顶层模块 (`digital_clock`) 和五个功能子模块：控制状态机 (`control_fsm_simplified`)、时间计数器 (`time_counter`)、闹钟控制器 (`alarm_controller`)、显示控制器 (`display_controller`) 和声音控制器 (`sound_controller`)。顶层模块负责实例化所有子模块并建立它们之间的连接。

4. 模块详细说明

4.1 `digital_clock.v`

文件作用: 定义了整个数字时钟系统的顶层模块。它负责整合所有子模块，连接输入/输出端口，并协调各个子模块之间的信号传递。

`digital_clock` 模块

- **模块作用:** 作为设计的顶层实体，实例化所有功能子模块 (`control_fsm_simplified`, `time_counter`, `alarm_controller`, `display_controller`, `sound_controller`)。它连接外部输入（时钟、按钮、开关）和内部模块，并将最终的显示数据和声音信号输出。
- **输入:**
  - `clk_1Hz`: 1Hz 时钟信号（计时用, 闪烁）。
  - `clk_1KHz`: 1KHz 时钟信号（FSM, 显示更新）。
  - `PULSE`: 模式切换/循环按键。
  - `CLR`: 复位按键 (高电平有效)。
  - `K_1`: 时间/闹钟选择 (0:时间模式, 1:闹钟模式)。
  - `S[3:0]`: 4个开关用于数据输入 (`set_data`)。
- **输出:**
  - `hour_tens[3:0]`: 小时十位 BCD。
  - `hour_ones[3:0]`: 小时个位 BCD。
  - `minute_tens[3:0]`: 分钟十位 BCD。
  - `minute_ones[3:0]`: 分钟个位 BCD。
  - `second_tens[3:0]`: 秒钟十位 BCD。
  - `second_ones[6:0]`: 秒钟个位 7-segment / 'A'。
  - `sound`: 报时/闹钟声音输出。
- `assign` **语句:**
  - `assign alarm_mode = K_1;`: 将外部输入开关 `K_1` 直接赋值给内部信号 `alarm_mode`，用于指示当前是时间模式还是闹钟模式。
  - `assign sound = w_display_sound | w_controller_sound;`: 将来自显示控制器（设置提示音）和声音控制器（报时/闹钟声）的声音信号通过逻辑或合并，产生最终的 `sound` 输出。

4.2 `control_fsm.v`

文件作用: 定义了控制数字时钟操作模式的有限状态机 (FSM)。

`control_fsm_simplified` 模块

- **模块作用:** 实现一个简化的状态机，用于响应 `mode_button` 输入，并在不同的操作状态（空闲 `IDLE`、设置小时十位 `SET_HT`、设置小时个位 `SET_HO`、设置分钟十位 `SET_MT`、设置分钟个位 `SET_MO`）之间切

换。它输出当前是否处于设置模式以及正在设置哪一位。

- **输入:**
  - `clk`: 时钟 (例如 1KHz)。
  - `reset`: 复位 (低电平有效)。
  - `mode_button`: 模式切换按钮。
- **输出:**
  - `is_setting_mode`: 高电平有效，指示是否处于任何设置状态。
  - `setting_digit[1:0]`: 指示当前正在设置哪一位 (00:HT, 01:HO, 10:MT, 11:MO)。
- `always @(posedge clk or negedge reset)` **(按钮状态更新):**
  - **作用:** 这是一个同步 `always` 块，用于在每个时钟上升沿或复位信号下降沿更新 `mode_button_prev` 寄存器。它存储上一个时钟周期的按钮状态，用于边沿检测。
- `assign mode_button_pressed`**:**
  - **作用:** 这是一个组合逻辑 `assign` 语句，用于检测 `mode_button` 的上升沿。当当前按钮状态 `mode_button` 为高电平且上一个周期的状态 `mode_button_prev` 为低电平时，`mode_button_pressed` 信号为高，表示按钮被按下。
- `always @(posedge clk or negedge reset)` **(状态转移):**
  - **作用:** 这是状态机的核心同步 `always` 块。在每个时钟上升沿或复位信号下降沿，它将 `next_state` 寄存器的值赋给 `current_state` 寄存器，完成状态的更新。复位时，将当前状态强制设置为 `IDLE`。
- `always @(*)` **(下一状态逻辑):**
  - **作用:** 这是一个组合逻辑 `always` 块，用于计算状态机的下一个状态 (`next_state`)。它根据当前状态 (`current_state`) 和按钮按下信号 (`mode_button_pressed`) 来决定下一个状态。默认情况下，保持当前状态。状态转换顺序为 IDLE → SET_HT → SET_HO → SET_MT → SET_MO → IDLE。
- `always @(*)` **(输出逻辑):**
  - **作用:** 这是一个组合逻辑 `always` 块，用于根据当前状态 (`current_state`) 确定模块的输出信号 `is_setting_mode` 和 `setting_digit`。`is_setting_mode` 在非 `IDLE` 状态为高，`setting_digit` 根据当前 `SET_` 状态确定。

4.3 `time_counter.v`

文件作用: 定义了负责计时和时间设置的核心模块。

`time_counter` 模块

- **模块作用:** 维护当前的时、分、秒计数，处理时间进位，响应时间设置操作，并生成整点报时信号。
- **输入:**
  - `clk_1Hz`: 1Hz 时钟输入 (用于计时和设置)。
  - `reset`: 异步复位，低有效。
  - `is_setting_mode`: 高电平有效，指示设置模式是否激活。
  - `is_setting_alarm`: 高电平有效，指示目标是否为闹钟（此时时间继续运行）。
  - `setting_digit[1:0]`: 指示正在设置哪一位 (00:HT, 01:HO, 10:MT, 11:MO)。
  - `set_data[3:0]`: 当前设置位的数据输入 (0-9)。
- **输出:**
  - `hours_tens[1:0]`: 小时十位 (0-2)。
  - `hours_ones[3:0]`: 小时个位 (0-9)。
  - `minutes_tens[2:0]`: 分钟十位 (0-5)。
  - `minutes_ones[3:0]`: 分钟个位 (0-9)。

    - `seconds_tens[2:0]`: 秒钟十位 (0-5)。
    - `seconds_ones[3:0]`: 秒钟个位 (0-9)。
    - `hourly_trigger`: 整点报时触发信号 (单周期脉冲 @ 1Hz)。
- `always @(posedge clk_1Hz or negedge reset)` **(计时与设置逻辑):**
    - **作用:** 这是该模块的核心时序逻辑块。它在 1Hz 时钟的上升沿或复位信号的下降沿执行。负责处理复位、时间设置（根据输入控制信号更新时间寄存器并进行有效性检查）和正常计时（BCD 计数、进位、边界条件处理）。同时，在时间变为整点的前一个周期产生 `hourly_trigger` 信号。

4.4 `alarm_controller.v`

文件作用: 定义了处理闹钟设置和闹钟触发逻辑的模块。

`alarm_controller` 模块

- **模块作用:** 存储用户设定的闹钟时间（时、分），将当前时间与设定的闹钟时间进行比较，并在匹配时生成闹钟触发信号。同时输出存储的闹钟时间供显示。
- **输入:**
    - `clk_1KHz`: 1KHz 时钟信号。
    - `reset`: 复位信号。
    - `is_setting_mode`: 高电平有效，指示设置模式是否激活。
    - `is_setting_alarm`: 高电平有效，指示目标是否为闹钟。
    - `setting_digit[1:0]`: 指示正在设置哪一位 (00:HT, 01:HO, 10:MT, 11:MO)。
    - `set_data[3:0]`: 当前设置位的数据 (来自外部输入)。
    - `current_hours_tens[1:0]`: 当前小时十位。
    - `current_hours_ones[3:0]`: 当前小时个位。
    - `current_minutes_tens[2:0]`: 当前分钟十位。
    - `current_minutes_ones[3:0]`: 当前分钟个位。
    - `current_seconds_tens[2:0]`: 当前秒钟十位。
    - `current_seconds_ones[3:0]`: 当前秒钟个位。
- **输出:**
    - `alarm_hours_tens_out[1:0]`: 输出存储的闹钟小时十位。
    - `alarm_hours_ones_out[3:0]`: 输出存储的闹钟小时个位。
    - `alarm_minutes_tens_out[2:0]`: 输出存储的闹钟分钟十位。
    - `alarm_minutes_ones_out[3:0]`: 输出存储的闹钟分钟个位。
    - `alarm_trigger`: 闹钟触发信号。
- `always @(posedge clk_1KHz or negedge reset)` **(闹钟设置与触发逻辑):**
    - **作用:** 这是闹钟控制器的核心时序逻辑块。它在 1KHz 时钟的上升沿或复位信号的下降沿执行。负责处理复位、闹钟时间设置（根据输入控制信号更新内部闹钟时间寄存器并进行有效性检查）以及闹钟比较与触发（比较当前时间与闹钟时间，在匹配且秒数为00时产生 `alarm_trigger` 信号）。
- `always @(*)` **(输出闹钟时间):**
    - **作用:** 这是一个组合逻辑 `always` 块，将内部存储的闹钟时间寄存器 (`internal_alarm_...`) 的值持续地赋值给输出端口 (`alarm_..._out`)。

4.5 `display_controller.v`

文件作用: 定义了控制数据显示和显示效果（如闪烁）的模块。

`display_controller` **模块**

- **模块作用:** 根据当前系统模式（正常显示、设置时间、设置闹钟）选择合适的 BCD 数据源，驱动数码管显示。它负责将 BCD 码转换为七段码（特别是秒个位），并在设置模式下实现对应位的闪烁效果和发出提示音。
- **输入:**
  - `clk`: 时钟 (例如 1KHz)。
  - `reset`: 复位信号 (低电平有效)。
  - `current_hours_tens[1:0]`: 当前小时十位。
  - `current_hours_ones[3:0]`: 当前小时个位。
  - `current_minutes_tens[2:0]`: 当前分钟十位。
  - `current_minutes_ones[3:0]`: 当前分钟个位。
  - `current_seconds_tens[2:0]`: 当前秒钟十位。
  - `current_seconds_ones[3:0]`: 当前秒钟个位。
  - `is_setting_mode`: 高电平有效，指示设置模式是否激活。
  - `is_setting_alarm`: 高电平有效，指示目标是否为闹钟。
  - `setting_digit[1:0]`: 指示正在设置哪一位 (00:HT, 01:HO, 10:MT, 11:MO)。
  - `blink_clk`: 用于闪烁效果的时钟 (例如 1Hz)。
- **输出:**
  - `hour_tens[3:0]`: 小时十位 BCD。
  - `hour_ones[3:0]`: 小时个位 BCD。
  - `minute_tens[3:0]`: 分钟十位 BCD。
  - `minute_ones[3:0]`: 分钟个位 BCD。
  - `second_tens[3:0]`: 秒钟十位 BCD。
  - `second_ones_seg[6:0]`: 秒钟个位七段码 / 'A'。
  - `sound_out`: 声音输出信号（设置提示音）。
- `function bcd_to_7seg`**:**
  - **作用:** 这是一个组合逻辑函数，接收一个 4 位的 BCD 码作为输入，并返回对应的 7 位七段码。它使用 `case` 语句将 BCD 值 0-9 映射到七段码模式。
- `always @(posedge blink_clk or negedge reset)` **(闪烁及提示音控制):**
  - **作用:** 这个同步 `always` 块由 1Hz 的 `blink_clk` 驱动。它负责翻转 `blink_on` 信号以控制闪烁，并在设置模式下根据 `blink_on` 的状态翻转 `sound_out` 以产生提示音。
- `always @(posedge clk or negedge reset)` **(显示更新逻辑):**
  - **作用:** 这个同步 `always` 块由较快的 `clk` (1KHz) 驱动，用于更新最终驱动数码管的输出寄存器。它根据系统模式（正常或设置）和控制信号（`is_setting_alarm`, `setting_digit`, `blink_on`）选择数据源，处理闪烁逻辑，并将最终的 BCD 码或七段码赋值给输出端口。在设置模式下，秒钟固定显示 "00"。在查看闹钟模式下（非设置状态），秒个位显示 'A'。

4.6 `sound_controller.v`

文件作用: 定义了根据触发信号产生不同频率声音的模块。

`sound_controller` **模块**

- **模块作用:** 接收整点报时和闹钟触发信号，并根据触发类型产生相应频率和持续时间的方波声音信号。
- **输入:**
  - `clk`: 时钟输入 (例如 10kHz)。

    - `reset`: 异步复位 (低电平有效)。
    - `hourly_trigger`: 整点报时触发信号。
    - `alarm_trigger`: 闹钟触发信号。
- **输出:**
    - `sound_out`: 声音输出信号。
- `always @(posedge clk or negedge reset)` **(声音生成逻辑):**
    - **作用:** 这是声音控制器的核心时序逻辑块，由较高频率的 `clk` 驱动。它负责处理复位、检测 `hourly_trigger` 和 `alarm_trigger` 的上升沿、启动声音播放、根据触发类型选择音调、使用计数器生成方波以及控制声音的持续时间。

5 小组成员分工

000: 控制逻辑与输入处理

- **主要职责:**
    - 设计和实现 控制状态机 (control_fsm) 模块

000: 核心计时功能

- **主要职责:**
    - 设计和实现 时间计数器 (time_counter) 模块

000: 闹钟功能与声音触发

- **主要职责:**
    - 设计和实现 闹钟控制器 (alarm_controller)
    - 设计和实现 声音控制器 (sound_controller)

000: 输出显示与系统集成/验证

- **主要职责:**
    - 设计和实现 显示控制器 (display_controller) 模块
    - 负责 顶层模块 (digital_clock) 的结构实现，例化所有子模块并完成它们之间的连线。
    - 承担主要的 仿真与测试 任务，验证整个系统的功能正确性。