EE217: 硬件描述语言与系统仿真

实验报告

组号：027
组员：戴天杰
朱雨欣
李　艳
张烨忱

前　言

FPGA，现场可编程门阵列，是在PAL、GAL、CPLD等可编程器件的基础上进一步发展的产物。它是作为（ASIC）领域中的一种半定制电路出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。

硬件描述语言与系统仿真是数字电路与逻辑设计的后续课程，分理论教学和上机实验两部分。通过这门课程，我们了解了数字集成电路及其设计方法的发展现状，熟悉大规模可编程专用集成电路CPLD/FPGA的内部结构，掌握一种硬件描述语言，并具备使用VHDL进行数字电路系统设计的能力。实验课是本课程重要的教学环节,在这个过程中我们逐渐熟悉可编程专用集成电路的设计，开发流程，熟练掌握一种EDA设计工具，提高了应用计算机技术进行数字电路与数字系统的设计和辅助分析的能力。

本实验基于STEP-MXO2 V2 FPGA实验板（简称小脚丫，核心器件为Lattice公司的LCMXO2- 4000HC- MG132）、STEP-Baseboard V2 外设实验底板、Lattice公司的Diamond FPGA集成开发工具软件、ModelSim仿真软件完成。

一、实验的一般方法

用VHDL文本进行输入、编译，通过之后再进行波形仿真、下载，如有缺陷，再对源文件进行修改。基于FPGA实验板完成指定和自选创意实验项目。

二、实验的基本内容和要求

2.1 常用的 PLD 软件工具的使用

- ✓ 学会Diamond的基本使用方法，熟悉设计流程。

2.2 PLD 设计代码的综合

- ✓ 掌握可综合的PLD设计代码的编写要点；
- ✓ 用VHDL语言编写并用Diamond工具完成电路综合、下载，实现设计项目。

2.3 了解速度和性能优化

- ✓ 根据不同的PLD器件特点优化设计的方法；
- ✓ 学习了解用器件时间信息仿真和优化。

三、STEP-MXO2 V2 FPGA 实验板介绍

STEP-MXO2 V2 FPGA 实验板简称小脚丫。实验板如图 0-1 所示。

3.1 核心器件特性

- ✓ Lattice LCMXO2-4000HC-4MG132
- ✓ 4320 个 LUT（查找表）资源；

- ✓ 96Kbit User Flash，92Kbit RAM；
- ✓ 2+2 路 PLL+DLL；
- ✓ 嵌入式功能块（硬核）：一路 SPI、一路定时器、2 路 I2C；
- ✓ 支持 DDR/DDR2/LPDDR 存储器；
- ✓ 上电瞬时启动，启动时间<1ms；
- ✓ FPGA 芯片最大可用 IO 数为 104 个，除了 36 个 IO 引出到开发板 DIP40 的引脚上，FPGA 的 IO 还连接到板上的外设资源如数码管、按键、拨码开关和 LED 上。

3.2 板载资源

- ✓ 1 路 Micro USB 接口
- ✓ 2 位 7 段数码管；
- ✓ 2 个 RGB 三色 LED；
- ✓ 4 路拨码开关；
- ✓ 4 路按键；
- ✓ 8 路用户 LED；
- ✓ 36 个用户可扩展 I/O（其中包括一路 SPI 硬核接口和一路 I2C 硬核接口）；
- ✓ 板上提供系统时钟 12MHz，也可以使用片内的内部时钟作为系统时钟；
- ✓ 集成 FT232 编程器，支持 Lattice Diamond 设计工具，用户只需要一根 Micro USB 连接线就能够实现板卡的供电和 FPGA 下载编程工作；
- ✓ 可以通过板上的 Micro USB 口 5V 供电，同时在 DIP40 的第 1 脚预留了 VBUS 口，可以外接 5V 电源实现供电。

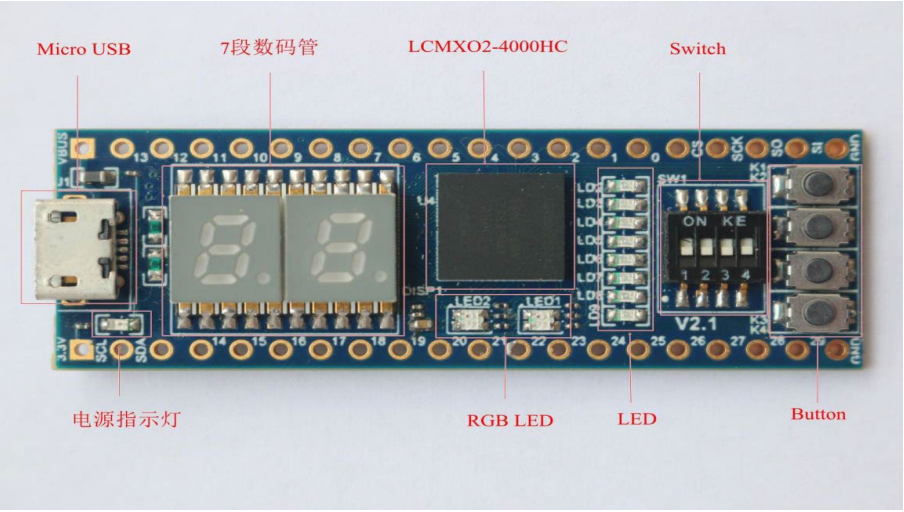

图 0-1 FPGA 实验底板图

四、STEP-Baseboard V2 外设实验底板介绍

根据STEP BaseBoard 底板的功能，整个底板划分为10个模块。外设底板如图0-2所示。

10个模块分别为：A/D、D/A模块，PS2接口模块、数码管模块、蜂鸣器模块、TFT-LCD模块、矩阵按键模块、旋转编码器模块、VGA模块、PMOD模块，USB转串口模块。

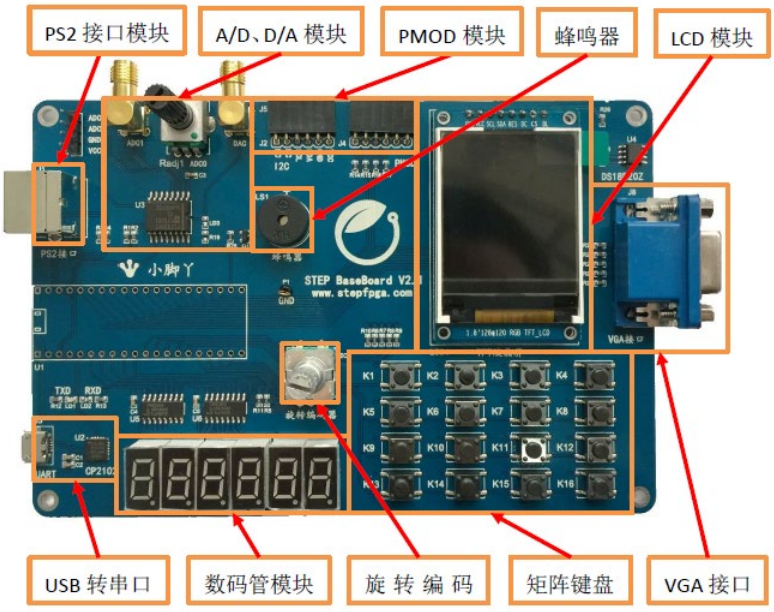

图 0-2 外设实验底板图

五、实验项目

5.1 必做实验

- ✓ 实验一　基于矩阵按键和数码管显示的键入系统设计；
- ✓ 实验二　基于PWM调制技术的呼吸灯系统设计；
- ✓ 实验三　基于数码管独立显示和三色灯的交通指示系统设计；
- ✓ 实验四　基于小脚丫底板数码管扫描显示的数字时钟系统设计。

5.2 拓展实验

- ✓ 实验五　基于小脚丫FPGA开发板和按键、蜂鸣器的小钢琴设计。

5.3 进阶实验

- ✓ 实验八　基于小脚丫FPGA的直流电压测量装置。

实验一　基于矩阵按键和数码管显示的键入系统设计

一 、实验目的

1、掌握独立和矩阵按键的原理及驱动；

2、学习基于小脚丫 FPGA 开发板和外设底板实现用 VHDL 语言编程对小脚丫上独立按键以及底板上矩阵按键进行信息采集；

3、学习基于小脚丫 FPGA 开发板和外设底板实现用 VHDL 语言编程对小脚丫进行按键消抖处理的技术；

4、了解数码管显示工作原理，掌握数码管驱动方式；

5、掌握用 VHDL 语言设计数码管显示驱动。

二 、实验内容

该任务主要是基于小脚丫二代核心板完成输入按键标识的显示，具体内容如下：

1、 对于底板上的 16 个扫描按键，按下时在数码管上显示对应的数值；

2、 对于 4 个独立按键，按下时在数码管上显示 17~20；

3、 利用延时采样或周期采样实现按键消抖。

三 、设计思路

3.1　顶层设计

LAB1_TOP.vhd 是实验 1 的顶层模块，该模块利用 component、port map 和子模块之间构建联系。在任务书中，clk 是时钟、key_code 是按键信号。脉冲发生器设计框图如图 1-1 所示。

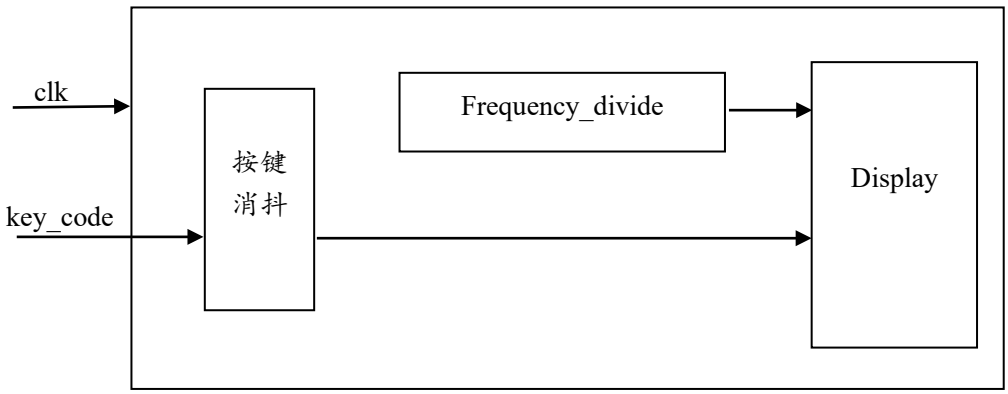

图 1-1 系统总框图

随后，我们利用 Netlist Analyzer 生成电路图，如图 1-2 所示。

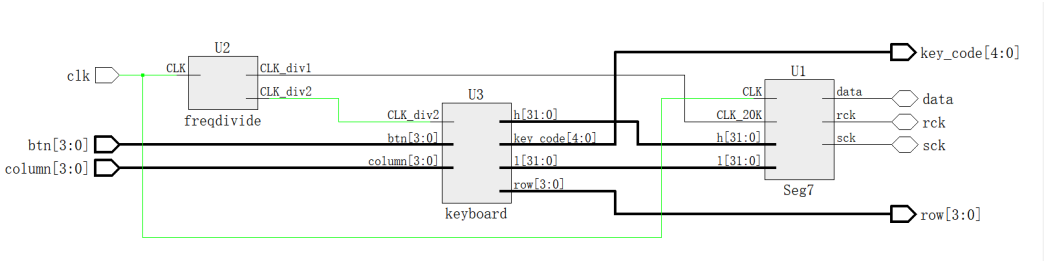

图 1-2 Lattice Diamond 生成的系统电路图

为了增强可读性，我们适当修改了部分变量名称，修改后的名称对应关系如表 1-1 所示。除顶层设计外，还包括 3.3~3.5 三个子模块。

表 1-1 变量名称对应关系

任务书中原变量名	实际变量名
clk	Clk
key_code	key_code
row	row
column	column

3.2 按键去抖

在实际工况中，按键的运作绝非是理想情况下按之则变的，实际上，机械按键在按下和放开的时候往往都存在着短时间的抖动。根据我们在《工程实践与科技创新 II-A》课堂上所授知识，我们可以利用周期采样消抖法进行按键去抖操作，根据实验说明书，我们将周期设置为了 20 毫秒。具体图示见图 1-3。

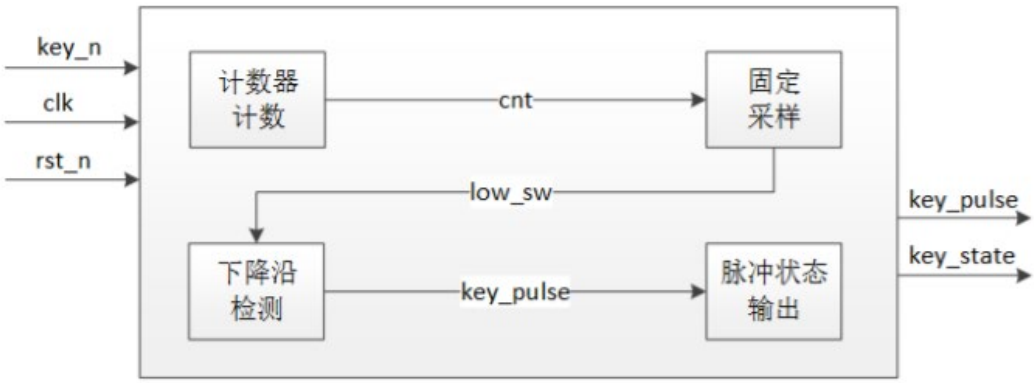

图 1-3 按键去抖图

3.3 分频模块

为了程序可读性以及可维护性，我们将系统时钟分频模块单独进行封装，本实验中主要包括脉冲调制信号的分频以及 74HC595 数据存入及发送部分的分频。通过查阅小脚丫技术手册，得知系统时钟频率为 12MHz，我们使用 400 次的计数器，分频后的时钟频率为 20kHz，输出为 clk_div1，作为七段显示器的输入时钟；使用 40000 次计数器分频得到频率为 20MHz

的时钟，输出为 clk_div2，按键读取的输入时钟。

3.4 数码管显示

数码管显示部分主要涉及 74HC595 位移缓存器的使用。通过查阅相关百度百科和 TI 官网 datasheet 可以了解到，74HC595 是串行输入、并行输出。其中，SCK 是输入时钟，RCK 是输出存储器锁存时钟线。SCK 上升沿时，数据寄存器的数据移位；SCK 下降沿时，移位寄存器数据不变。RCK 上升沿时，移位寄存器的数据进入数据存储寄存器；RCK 下降沿时，存储寄存器数据不变。在实验中，我们需要使用 2 块 74HC595，分别用于选通数码管、控制当前数码管显示内容。

74HC595 共有 3 种工作模式，分别为重置、配置数据、传输数据。在配置阶段，一共有 16 位数据，高 8 位中存储的是时间对应的 LED 字型编码，低 8 位存储的是点亮哪两个数码管（时、分或秒）的序号。在传输数据阶段，需要传输这 16 位数据，考虑到 74HC595 串行输出的特性，我们将这 16 位数据从最高位逐个输入，在 SCK 下降沿时放入数据等待，在 SCK 上升沿时数据进入 74HC595。串行输入结束后，立刻产生并行输出 RCK 信号，将信号传入数码管进行显示。具体过程如图 1-4 所示。

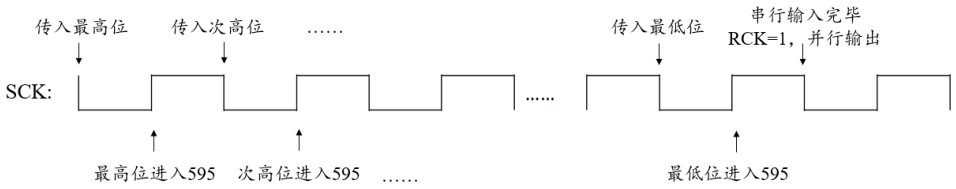

图 1-4 74HC595 串行输入、并行输出图示

3.5 按键值获取

本实验要求根据按键不同，数码管显示不同的内容，因此首先要获取按键值。按键分为两种：小脚丫板上的独立按键以及底板上的矩阵按键。

独立按键分别对应一个 I/O 接口，因此通过检测对应管脚的电压值即可获取其按键值，在程序中，只需要设置对应的变量，判断相应变量的值，1 为松开，0 为按下。

矩阵按键的电路原理图如图 1-5 所示，将按键按矩阵排列，使用行线和列线分别连接到按键开关的两端，列线通过上拉电阻连接到 VCC。当没有按键按下的时候，列线处于高电平的状态；当有按键按下的时候，列线电平由与此列线相连的行线电平决定。最后通过行列扫描法就可以判断各按键的操作状态。

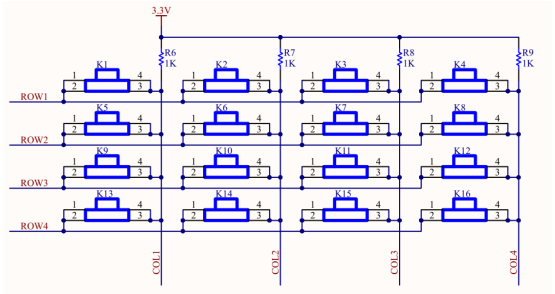

图 1-5 矩阵按键值的获取

在程序中，行作为输出端口，4 行分别输出低电平轮流循环，其余行均输出高电平。同时检测列的输出值，若出现 0 值，则表明有按键按下。此时，根据输出低电平的行号以及检测到低电平的列号，即可判断出哪个按键被按下。

四 、作品使用说明

4.1 整体效果

默认情况下，七段显示器归零显示。当按下按键矩阵当中的一个按键时，七段显示器上会显示相对应的数字，当按下四个独立按键中的任意一个时，七段显示器会显示 17~20 中与其对应的一个数字。

4.2 管脚定义

表 1-2 管脚定义

变量	管脚	功能
rck，sck，data	H3，J2，G3	74HC595
clk	C1	内部时钟
row	N8, P8, N7, P7	矩阵按键-行
column	L3, N5, P6,N6	矩阵按键-列
btn	L14, M13, M14, N14	独立按键

五 、设计中碰到的问题及解决方案

5.1 矩阵按键实现

一开始我们并未正确理解它的实现方法，在认真阅读硬件手册之后，才理解到矩阵按键是 8 个接口，4 输入 4 输出，本身就是一个交互的过程。首先对行电平进行检测，再对列电平进行检测，若检测到有高电平则代表此按键被按下，从而可以实现对行和列的定位。

六 、FPGA 资源使用

6.1 Design Summary

```
Number of registers:    115 out of  4635 (2%)
   PFU registers:          115 out of  4320 (3%)
   PIO registers:            0 out of   315 (0%)
Number of SLICEs:       188 out of  2160 (9%)
   SLICEs as Logic/ROM:    188 out of  2160 (9%)
   SLICEs as RAM:            0 out of  1620 (0%)
   SLICEs as Carry:         80 out of  2160 (4%)
Number of LUT4s:        376 out of  4320 (9%)
   Number used as logic LUTs:        216
   Number used as distributed RAM:     0
   Number used as ripple logic:      160
   Number used as shift registers:     0
Number of PIO sites used: 21 + 4(JTAG) out of 105 (24%)
```

```
Number of block RAMs:  0 out of 10 (0%)
Number of GSRs:        0 out of 1 (0%)
EFB used :         No
JTAG used :        No
Readback used :    No
Oscillator used :  No
Startup used :     No
POR :              On
Bandgap :          On
Number of Power Controller:  0 out of 1 (0%)
Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
Number of DCCA:  0 out of 8 (0%)
Number of DCMA:  0 out of 2 (0%)
Number of PLLs:  0 out of 2 (0%)
Number of DQSDLLs:  0 out of 2 (0%)
Number of CLKDIVC:  0 out of 4 (0%)
Number of ECLKSYNCA:  0 out of 4 (0%)
Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

综合上述内容可知，FPGA 寄存器资源占用了 2%，片选部分占用了 9%，逻辑运算单元同样占用了 9%。显然，资源占用较少，这一方面体现出我们代码性能优良，当然也和本实验难度较低有关。

6.2 Design Errors/Warnings

```
No errors or warnings present.
```

七 、总结

本次实验是基础实验的开端，对后续实验的进行都有着奠基作用。在本实验中，我们初步体验了 VHDL 语言的编写，感受到了硬件交互的原理，学会了如何按键去抖、分频等。

八、参考文献

[1]硬件描述语言与系统仿真课程实验要求书;
[2]STEP-MXO2 V2 硬件手册;
[3]STEP FPGA Base Board 底板硬件手册.

实验二　基于 PWM 调制技术的呼吸灯系统设计

一、实验目的

1、 了解 PWM 概念及原理；
2、 学习基于 PWM 调制技术的脉冲发生器的原理；
3、 掌握 LED 通过 PWM 调节亮度的方法；
4、 掌握用 VHDL 语言实现呼吸灯的方法。

二、实验内容

该任务采用 FPGA 实现小脚丫开发板的 LED 灯的呼吸效果，即缓缓变亮，缓缓熄灭，周而复始，实现“呼吸”的效果。并可利用按键控制呼吸灯变化速度。具体设计内容如下：

1、实现三个按键：速度增键、速度减键、复位按键；
2、实现呼吸效果：八个 LED 等周而复始的缓慢变亮，缓慢熄灭，实现“呼吸”效果；
3、实现复位功能：当按下复位按键后，呼吸灯变化速度重置为初始状态；
4、实现呼吸灯速度控制：当按下对应按键时，呼吸灯变化速度实现变快或变慢效果。

三、设计思路

3.1　顶层设计

LAB2_TOP.vhd 是实验 2 的顶层模块，该模块利用 component、port map 和子模块之间构建联系。在任务书中，clk 是时钟、rst_n 是重置信号，以及三个按键控制脉冲发生器的周期及脉宽参数，key_menu 控制周期和脉宽调节模式的切换、 key_up 依据所处的模式控制周期或脉宽参数的增加、key_down 依据所处的模式控制周期或脉宽参数的减小。脉冲发生器设计框图如图 2-1 所示。

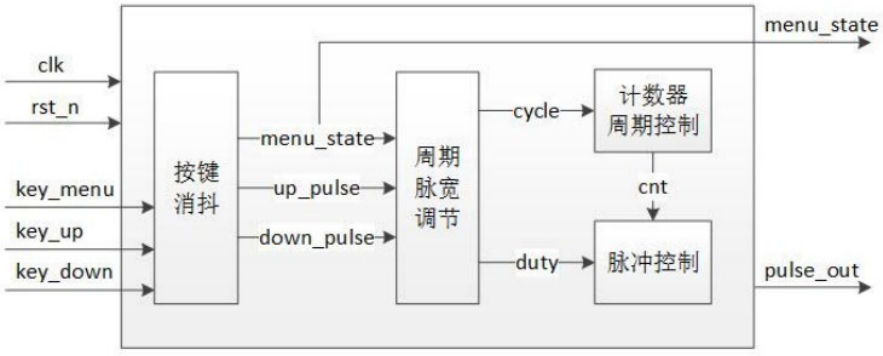

图 2-1 脉冲发生器设计框图

随后，我们利用 Netlist Analyzer 生成电路图，如图 2-2 所示。

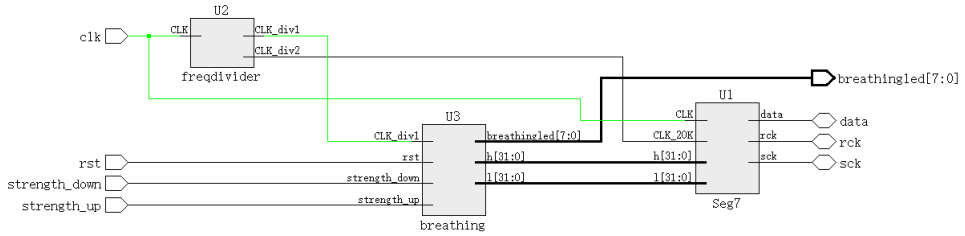

图 2-2 Lattice Diamond 生成的系统电路图

为了增强可读性，我们适当修改了部分变量名称，修改后的名称对应关系如表 2-1 所示。除顶层设计外，还包括 2.2~2.3 两个子模块。

表 2-1 变量名称对应关系

任务书中原变量名	实际变量名
clk	clk
rst_n	rst
key_up	strength_up
key_down	strength_down

3.2 分频模块

为了程序可读性以及可维护性，我们将系统时钟分频模块单独进行封装，本实验中主要包括脉冲调制信号的分频以及 74HC595 数据存入及发送部分的分频。通过查阅小脚丫技术手册，得知系统时钟频率为 12MHz，我们使用 20000 次的计数器，分频后的时钟频率为 300Hz，输出为 clk_div1，为生成脉冲调制信号做准备；使用 300 次计数器分频得到频率为 20kHz 的时钟，输出为 clk_div2，作为 Seg7 的串行输入时钟 clk20k。图 2-3 为分频模块电路图。

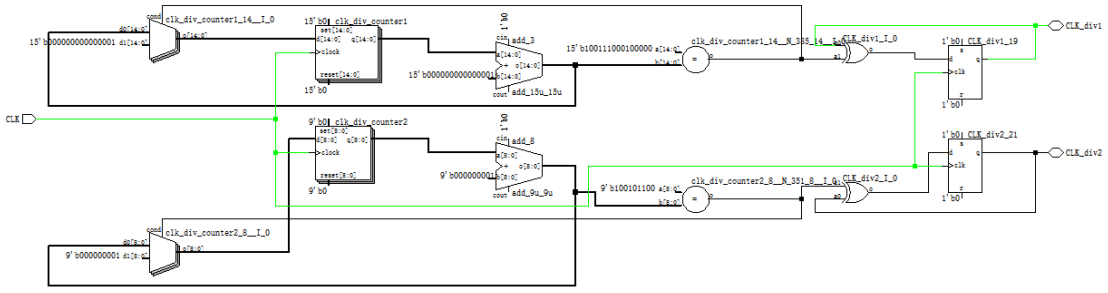

图 2-3 实验二分频模块电路图

3.3 呼吸灯模块

3.3.1 生成 PWM

脉冲宽度调制是一种对模拟信号进行数字编码的方法，根据相应载荷的变化来调制晶体管基极或 MOS 管栅极的偏置，来实现晶体管或 MOS 管导通时间的改变，从而实现开关稳压电源输出的改变。这种方式能使电源的输出电压在工作条件变化时保持恒定，是利用微处理器的数字信号对模拟电路进行控制的一种非常有效的技术。广泛应用在从测量、通信到功率控制与变换的许多领域中[4]。

视觉暂留（Persistence of Vision）现象指光对视网膜所产生的视觉在光停止作用后，仍

保留一段时间的现象。视觉实际上是靠眼睛的晶状体成像，感光细胞感光，并且将光信号转换为神经电流，传回大脑引起人体视觉。感光细胞的感光是靠一些感光色素，感光色素的形成是需要一定时间的，这就形成了视觉暂停的机理[5]。呼吸灯就是利用了这一原理，通过与定时器相结合，快速的更新 PWM 的占空比，导致 Led 灯看起来渐亮渐暗，而不会有闪烁的感觉。

在任务书中，clk 是时钟、rst_n 是重置信号，以及三个按键控制脉冲发生器的周期及脉宽参数，key_menu 控制周期和脉宽调节模式的切换、 key_up 依据所处的模式控制周期或脉宽参数的增加、key_down 依据所处的模式控制周期或脉宽参数的减小。

本实验采用比较器实现交集性调制方法生成 PWM，生成 PWM 设计框图如图 2-4 所示。其中，计数器 cnt1 生成载波信号（锯齿波），计数器 cnt2 生成调制信号（三角波），最后通过比较器得到 PWM 信号 pulse_out，从而实现呼吸灯功能。

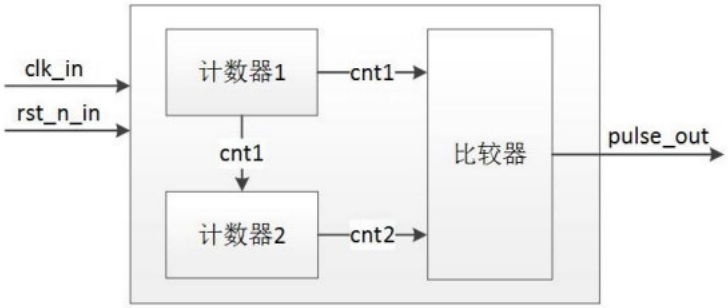

图 2-4 生成 PWM 设计框图

图 2-5 为锯齿波和三角波通过交集性调制方法生成 PWM 的实例波形图。

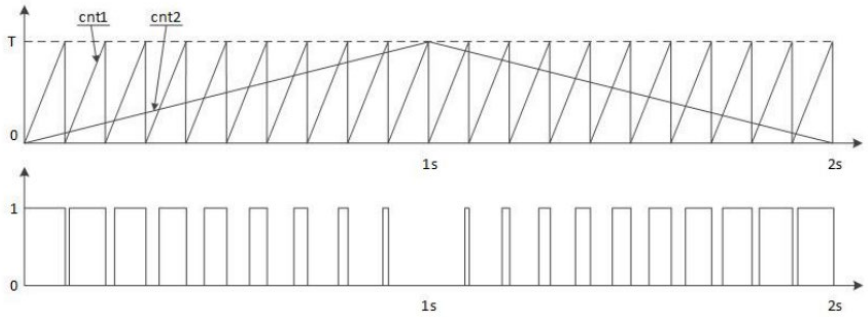

图 2-5 呼吸灯效果原理图

基于上述原理编写程序代码，本部分代码流程图如图 2-6。rst 为复位标志，当该值有效时，各参数恢复初始默认状态；sawtooth_step 和 triangle_step 分别为当前锯齿波和三角波的计数步长，max_amplitude 和 max_step 分别控制调整幅度的最大值以及调整步长的最大值，当识别到速度增或速度减按键按下时，程序修改锯齿波变化步长，从图 2-5 可以看出，任意改变一个波型的频率均能引起 PWM 波变化频率的改变，当接收到速度增指令时，增大幅度变化步长，反之，当接收到速度减指令时，减小幅度变化步长，从而实现对 PWM 波频率的控制。

生成锯齿波较为简单，只需在每个时钟上升沿按照当前步长增大计数值，当计数值超过最大幅度时归零，重新进入下一轮计数即可；而为了生成三角波，还需要使用 temp2_state 用于保存当前计数状态，即计数增或计数减状态，当计数超过计数最大值或小于 0 时更改计数状态，从而生成周期性上升与下降的三角波。

最后，如上所述，采用比较器实现交集性调制方法生成 PWM，即比较两路计数器。如果计数器 1 的值大于计数器 2 的值，则输出置‘0’，此时点亮 LED 灯；否则，输出置‘1’，LED

灯熄灭。同时，为了可视化实际模式，我们增设了数码管显示当前状态，具体数码管模块在实验一已经封装好，直接调用即可，在此不再赘述。

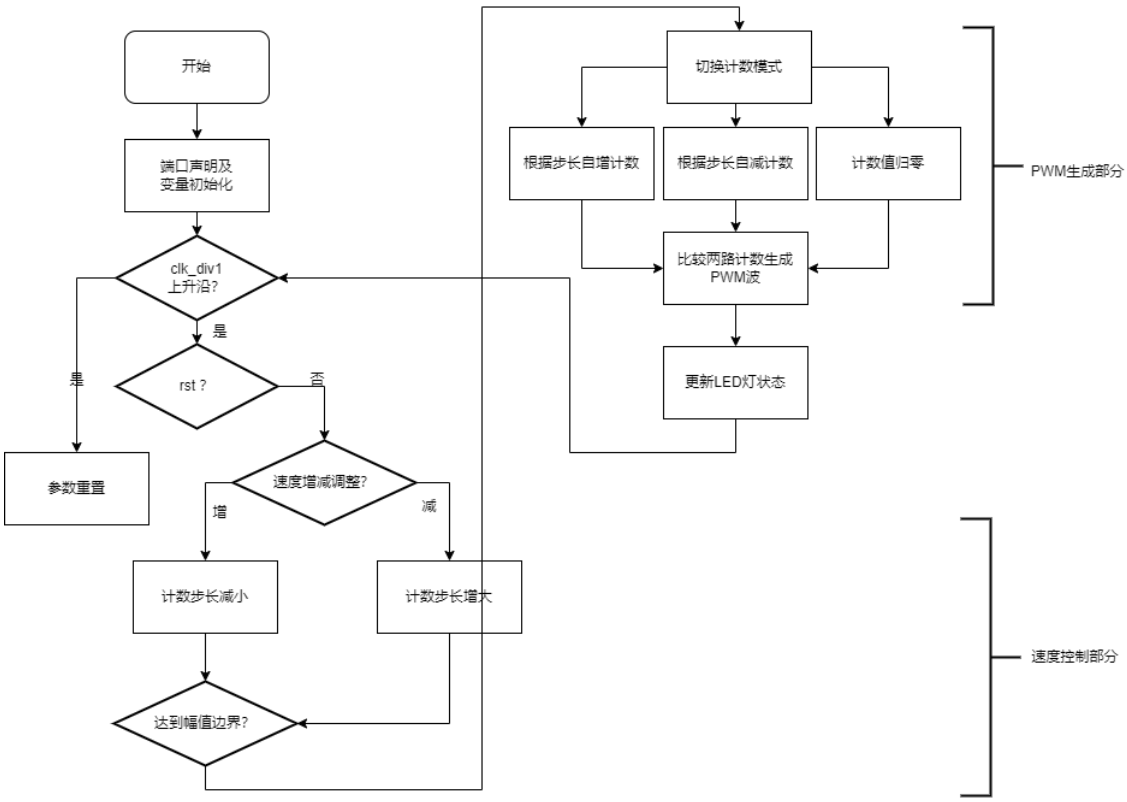

图 2-6 呼吸灯调制程序流程图

随后，我们利用 Netlist Analyzer 生成电路图，如图 2-7 所示。

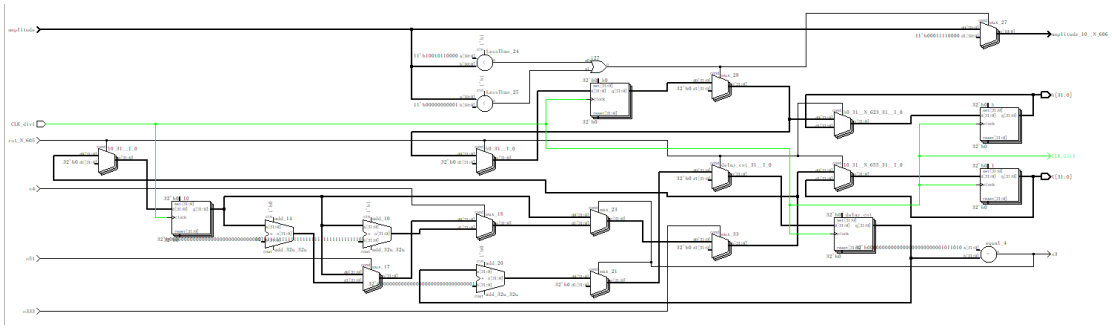

图 2-7 实验二呼吸灯模块电路图

3.3.2 参数调整

呼吸灯是基于人眼视觉效应实现的，为确保实际观测效果较好，对各参数的设置也相对较高，尤其是频率的选择。通过查阅资料，当物体移去时，视神经对物体的印象不会立即消失，而要延续 0.1~0.4 秒的时间[5]。

我们将时钟频率设置成 300Hz，三角波和锯齿波的最大幅值为 1200，三角波变化步长固定为 1，锯齿波变化步长从 1 至 60 等分成 5 等分，作为 5 档速度切换，初始默认为 12。在初始状态下，以三角波周期粗略估计呼吸灯从亮至灭约 8 秒，以锯齿波周期粗略估计呼吸灯闪烁更新周期约为 0.67 秒，示意图如图 2-8。随着速度档上升，这两个时间周期会相应的成比例变化，在本组采用的参数下，理论计算呼吸灯从亮至灭周期约为 1~8 秒，呼吸灯闪烁

更新周期约为 0.13~0.67 秒，理论上，这两组时间参数能较好的观察到呼吸灯的变化。

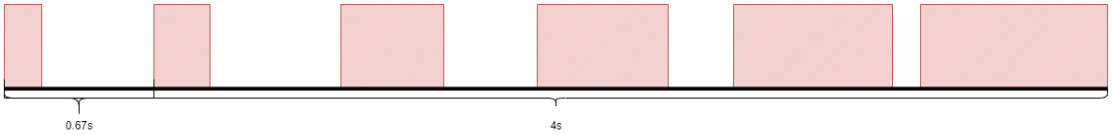

图 2-8 0.67s 半周期连续变化示意图

我们通过实验观测，呼吸灯各项功能良好，在各种速度档下均能连续观测到呼吸灯有较好的显示效果，实验效果基本与理论分析一致。

四、作品使用说明

4.1 整体效果

默认状态下，频率为 300Hz 的时钟用于生成三角波和锯齿波，呼吸灯速度档位于第一档，此时呼吸灯缓缓变亮，缓缓熄灭，周而复始，实现“呼吸”的效果，单个呼吸周期约为 8s。若按下 strength_up 键，速度档加一，呼吸灯“呼吸”速率增大；若按下 strength_down 键，速度档见一，呼吸灯“呼吸”速率减小；若按下 rst 键，呼吸灯速度档回到第一档。整体效果上，能正常实现控制速率以及复位功能，并且每一个速度档位的呼吸灯视觉效果良好，各项功能均已实现。

4.2 管脚定义

表 4-2 列出了实验二对应的 FPGA 芯片 LCMXO2-4000HC-4MG132 管脚分配。

表 2-2 管脚定义

变量	管脚	功能
RCK，SCK，DATA	H3，J2，G3	74HC595
rst	L14	重置
clk	C1	内部时钟
breathingled	N13，M12，P12，M11，P11，N10，N9，P9	呼吸灯显示
strength_down	M14	速度增快
strength_up	N14	速度变慢

五、设计中碰到的问题及解决方案

5.1 呼吸灯视觉效果不佳

在编写好程序后，将程序烧到小脚丫版进行观察，发现无论如何调整速度档位，观测到的现象均是 8 个 led 灯一直点亮，此时无法确定是由于频率参数设置不合理还是程序代码有疏漏造成的这一现象。为此，我们分别采用了通过查阅资料进行理论验算,对上述猜想进行验证。通过查阅资料，结合人眼视觉效应相关理论计算出合适的频率参数，具体分析详见本部分 3.3.2 节。通过修改参数后，呼吸灯效果明显改善，此问题得到解决。

六、FPGA 资源使用

6.1 Design Summary

```
Number of registers:   187 out of  4635 (4%)
   PFU registers:         187 out of  4320 (4%)
   PIO registers:           0 out of   315 (0%)
Number of SLICEs:      236 out of  2160 (11%)
   SLICEs as Logic/ROM:    236 out of  2160 (11%)
   SLICEs as RAM:           0 out of  1620 (0%)
   SLICEs as Carry:       127 out of  2160 (6%)
Number of LUT4s:       472 out of  4320 (11%)
   Number used as logic LUTs:       218
   Number used as distributed RAM:    0
   Number used as ripple logic:     254
   Number used as shift registers:    0
Number of PIO sites used: 16 + 4(JTAG) out of 105 (19%)
Number of block RAMs:  0 out of 10 (0%)
Number of GSRs:       0 out of 1 (0%)
EFB used :       No
JTAG used :      No
Readback used :   No
Oscillator used : No
Startup used :    No
POR :           On
Bandgap :        On
Number of Power Controller:  0 out of 1 (0%)
Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
Number of DCCA:  0 out of 8 (0%)
Number of DCMA:  0 out of 2 (0%)
Number of PLLs:  0 out of 2 (0%)
Number of DQSDLLs:  0 out of 2 (0%)
Number of CLKDIVC:  0 out of 4 (0%)
Number of ECLKSYNCA:  0 out of 4 (0%)
Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

综合上述内容可知，FPGA 寄存器资源占用了 4%，片选部分占用了 11%，逻辑运算单元同样占用了 11%。

6.2 Design Errors/Warnings

```
No errors or warnings present.
```

七、总结

通过实验一的经验积累，我们已经初步掌握了从建立工程，包括工程路径、芯片选型，到源文件输入，编译综合（Synthesis），添加约束、分配管脚，实现（Implementation），以及验证（Verification），包括时序仿真和功能仿真，生成下载的 bitstream 文件，最后下载工程文件到开发板，整个实验的基本流程。本实验作为本次课程的第二个实验，我们进一步了解 PWM 概念及原理并学习基于 PWM 调制技术的脉冲发生器的原理，进而通过 PWM 调节亮度实现呼吸灯。

在实验过程中，我们团队成员针对该实验遇到了问题展开了深入讨论、共同寻找问题的解决方案，最终很好的调整了呼吸灯的各项参数，达到满意的观测效果。同时，为了程序的可读性和可维护性，我们对编写好的代码进行了封装。适当的小组讨论也增强了我们的团队合作与沟通，从小组成员中相互学习到了较好的调试与编程思路。这个实验为我们小组提供了一个很好的学习机会，通过实验进一步巩固了课堂知识，达到学以致用的效果，为后续的实验打下了良好基础。

八、参考文献

[1]硬件描述语言与系统仿真课程实验要求书;
[2]STEP-MXO2 V2 硬件手册;
[3]STEP FPGA Base Board 底板硬件手册;
[4]脉冲宽度调制[EB/OL]. https://baike.baidu.com/item/脉冲宽度调制/10813756
[5]视觉暂留[EB/OL]. https://baike.baidu.com/item/视觉暂留/5125149

实验三　基于数码管独立显示和三色灯的交通指示系统设计

一、实验目的

1、掌握有限状态机的原理、设计方法与步骤；
2、用 VHDL 语言实现十字路口交通灯控制程序。

二、实验内容[1]

该任务主要是数码管独立显示和三色灯的交通指示系统设计，利用小脚丫 FPGA 实验板上的两个三色 LED 灯，模拟实现十字路口的红绿灯控制系统。具体内容如下：

1、可通过按键设置不同控制模式，实现多种模式下的控制方式。如可以设置普通双向对等模式、一向主干道一向次干道的非对等控制模式或者双向长黄闪灯模式等；

2、数码管可显示读秒数据。

三、设计思路

3.1　顶层设计

LAB3_TOP.vhd 是实验 3 的顶层模块，该模块利用 component、port map 和子模块之间构建联系。在本任务中，clk 是输入时钟信号，btn 是输入按键信号组，light1、light2 是模拟的交通指示灯，rck、sck 和 data 是 74HC595 的相关端口。总系统框图如图 3-1 所示。

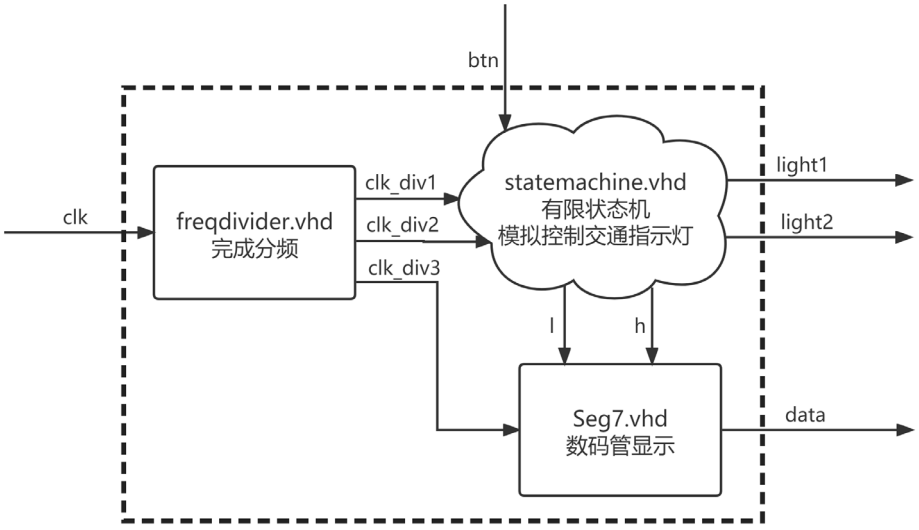

图 3-1 系统总框图

随后，我们利用 Netlist Analyzer 生成电路图，如图 3-2 所示。

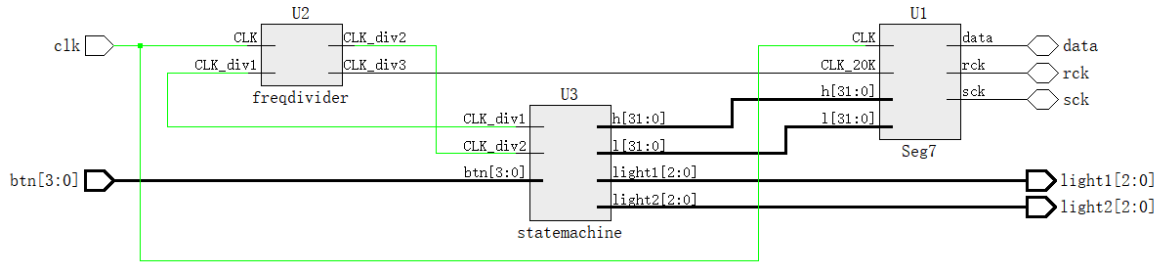

图 3-2 Lattice Diamond 生成的系统电路图

除顶层设计外，还包括 3.2~3.4 共计 3 个子模块。

3.2 分频模块

同样为了程序可读性以及可维护性，我们将系统时钟分频模块封装为 freqdevider 模块。实验中主要涉及针对按键读取的分频 clk_div1、针对状态机的分频 clk_div2、针对 74HC595 的分频 clk_div3。

具体地，由于一次按键按下的动作约为数百毫秒，抖动时间约为 10ms[2]，故采用对按键周期采样的处理方式进行消抖，将频率为 12MHz 的输入时钟信号进行 80000 分频作为按键信号检测频率;将频率为 12MHz 的输入时钟信号分频至 1Hz 作为状态机模块的时钟信号，这样可以简化计时，针对交通指示灯的读秒操作只需在状态机中进行加减 1 秒计时即可完成；将频率为 12MHz 的输入时钟信号进行 800 分频得到 15kHz 的信号作为 74HC595 数据配置与发送的频率。

该部分电路图见图 3-3。

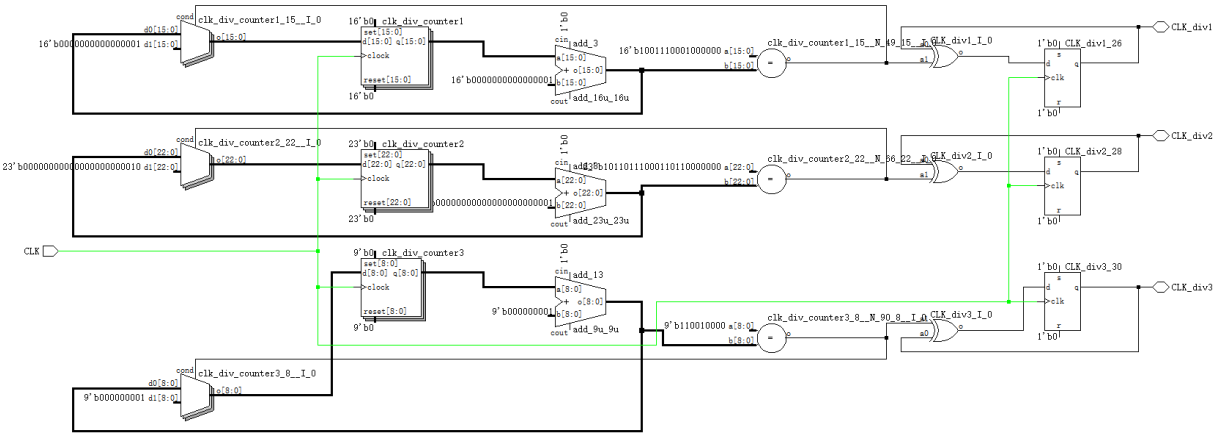

图 3-3 实验三的分频模块电路图

3.3 状态机

在任务要求中，我们需要实现多种模式下的控制方式，包括普通双向对等模式、一向主干道一向次干道的非对等控制模式和双向长黄闪灯模式。在实际操作中，我们选用 STEP FPGA 核心板上的 4 个按键进行模式选择，四个键分别对应：选择普通双向对等模式、选择一向主干道一向次干道的非对等控制模式、选择双向长黄闪灯模式、重置。该模块整体流程图如图 3-4 所示。

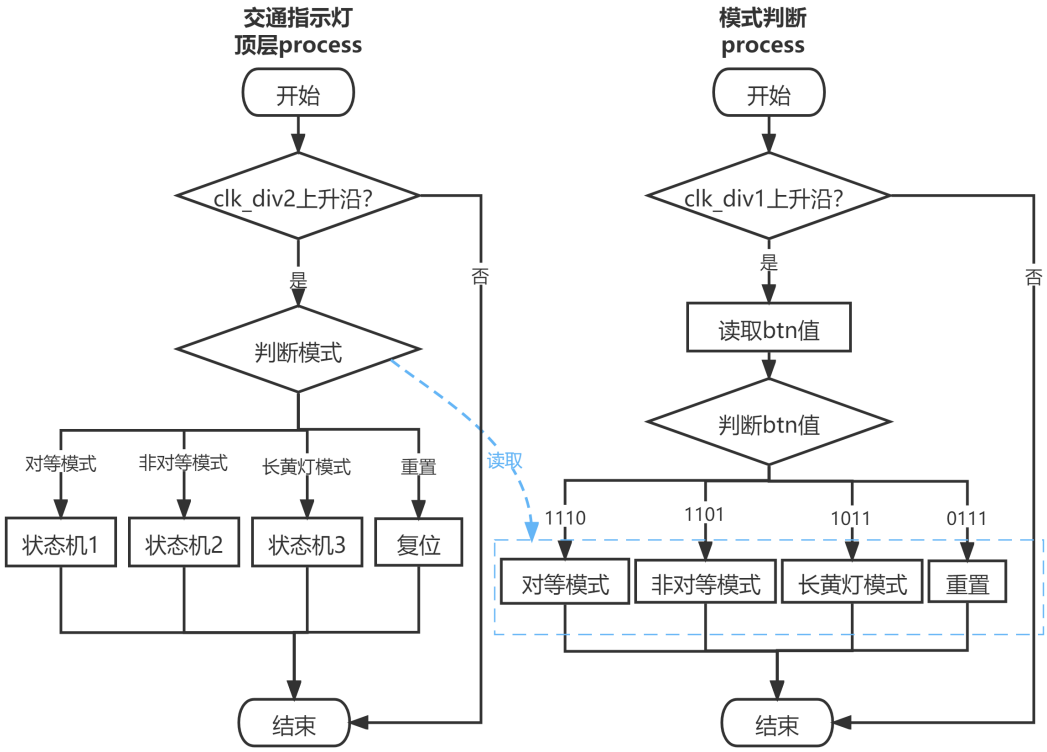

图 3-4 状态机模块流程图

3.3.1 普通双向对等模式

在该模式下，需模拟不区分主干道、次干道的十字交叉路口，各道路红绿灯时间相同，在本实验中都设置为 12 秒，黄灯时间为 2 秒。不妨设 light1 为南北向红绿灯，light2 为东西向红绿灯，数码管完成南北向红绿灯的计时读秒。状态机状态转移图具体图示见图 3-5。

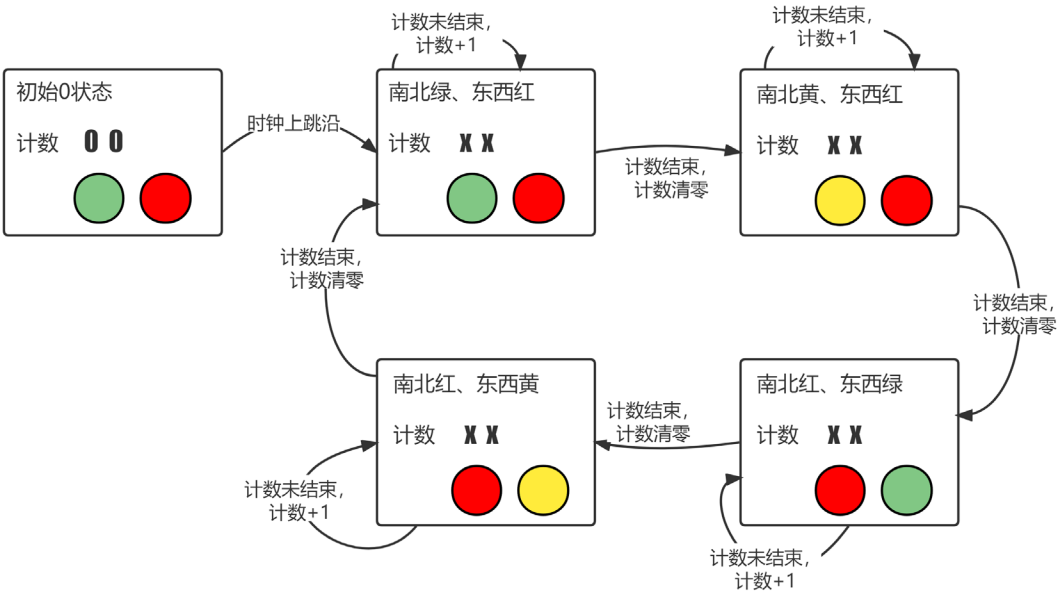

图 3-5 普通双向对等模式状态转移图

3.3.2 一向主干道一向次干道的非对等模式

在该模式下，需要模拟一条主干道和一条支干道的汇合点形成的十字交叉路口，主、支干道的红绿灯闪亮时间不相同。分析可知，该状态机逻辑与 3.3.1 节中提到的是一致的，只需修改红绿灯持续时间即可。在本实验中，设南北向为主干道，将南北向绿灯计数终值赋为 12 秒，红灯计数终值赋为 6s，即可实现非对等的交通指示灯系统。

3.3.3 双向长黄闪灯模式

在该模式下，需要模拟两个方向上的交通灯均处于黄色长闪烁的状态，状态转移图如图 3-6 所示。

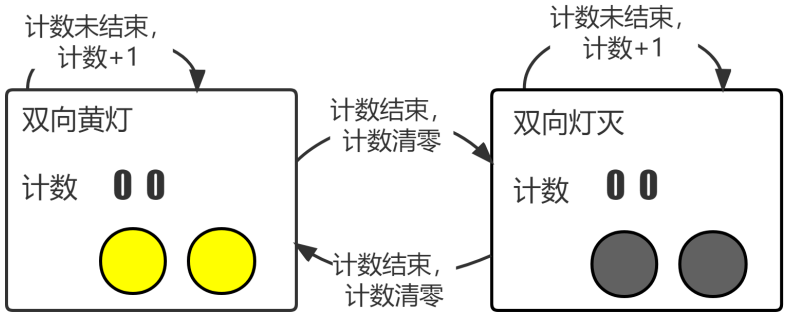

图 3-6 双向长黄闪灯模式状态转移图

3.4 数码管显示

数码管显示部分主要涉及 74HC595 位移缓存器的使用。STEP FPGA 底板上数码管有 6 位，对应就有 6 个使能位，以及每个数码管有 8 位（包括小数点），这是通过两个 595 芯片实现的，硬件连接如图 3-7。

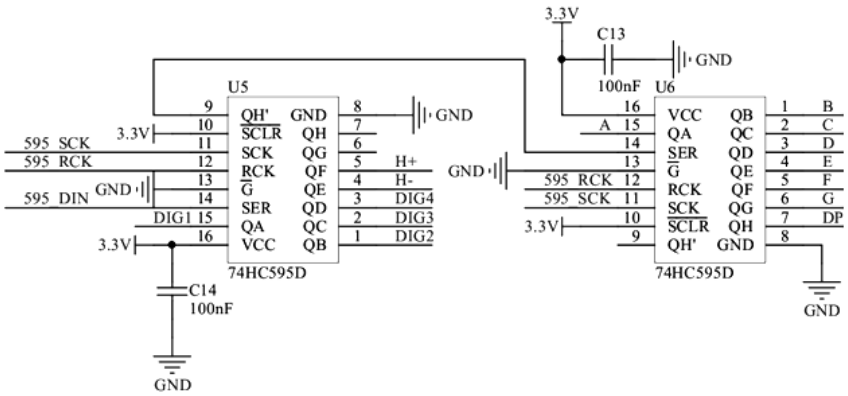

图 3-7 STEP FPGA 底板上数码管硬件连接图[3]

由于交通灯持续时间多为几十秒，因此在本实验中点亮两位数码管即可。

由图 3-7，两个数码管实现了一个 16 位的移位寄存器，第一个芯片的 15 号管脚输入数据，而两个芯片的 12 号和 13 号管脚，分布控制数据的移位和内部数据的显示，因此通过在 15 管脚加入 data，并在 12、13 号管脚输入脉冲，即可控制数码管的点亮与否和显示的内容。

在本实验中，h 作为计数的高位传入该模块，通过数码管字库映射为 8 个 bit 作为 data 的高位，代表数码管高位选通的 8 个 bit 则作为 data 的低位，之后串行注入数据链，并行输出；在下一个周期中，l 分别作为计数的低位传入该模块，通过数码管字库映射为 8 个 bit

作为 data 的低位，代表数码管低位选通的 8 个 bit 则作为 data 的低位，之后串行注入数据链，并行输出。以上两个过程交替，通过人眼视觉暂留效应即可实现两位数码管的同时显示。

四、作品使用说明

4.1 整体效果

默认状态下，进入普通双向对等模式，初始状态为南北向（light1）绿灯，东西向（light2）红灯，数码管从 00 开始计时，数码管显示计时单位为 s；当计时达到 12s 时，南北向（light1）转黄灯，数码管计数清零；当计时达到 2 秒后，南北向（light1）转红灯，东西向（light2）转绿灯，数码管从 00 开始计时；当计时达到 12s 时，东西向（light2）转黄灯，数码管计数清零；当计时达到 2 秒后，南北向（light1）转绿灯，东西向（light2）转红灯。当按下 btn1 时，进入主干道次干道非对等模式，效果与前者差异只在于东西向（light2）绿灯持续时间缩短为 6s。当按下 btn2，南北向（light1）、东西向（light2）进入同时点亮 2 秒、同时熄灭 2s 的循环模式。当按下 btn3 后，回到开机初始状态，数码管计时清零。当按下 btn0，可回到普通双向对等模式。

4.2 管脚定义

表 3-1 管脚定义

变量	管脚	功能
RCK，SCK，DATA	H3，J2，G3	74HC595
btn[3]	N14	重置
btn[2]	M14	进入双向长黄闪灯模式
btn[1]	M13	进入非对等控制模式
btn[0]	L14	进入普通双向对等模式
light1[],light2[]	P2、N2、M2，P4、N3、M3	两个 RGB 三色灯模拟交通灯

五、设计中碰到的问题及解决方案

5.1 模式切换后交通灯显示状态不稳定

我们发现，每次模式切换后交通灯并没有稳定的进入每个状态机的第一个子状态，这一现象在双向长黄闪灯模式下由于子状态较为简单，故还不明显；但是在切换回双向对等模式和双向非对等模式时由于状态机较为复，这种初始状态的不确定性是由于在状态机中未添加默认初始零状态，而这也正是实验三指导文件中所提及的。通过这一子状态的添加，我们成功实现了交通灯系统模式的切换，也使得程序具有稳健性。

六、FPGA 资源使用

6.1 Design Summary

```
Number of registers:   191 out of  4635 (4%)
```

```
    PFU registers:          191 out of  4320 (4%)
    PIO registers:            0 out of   315 (0%)
  Number of SLICEs:       275 out of  2160 (13%)
    SLICEs as Logic/ROM:    275 out of  2160 (13%)
    SLICEs as RAM:            0 out of  1620 (0%)
    SLICEs as Carry:        135 out of  2160 (6%)
  Number of LUT4s:        549 out of  4320 (13%)
    Number used as logic LUTs:        279
    Number used as distributed RAM:     0
    Number used as ripple logic:      270
    Number used as shift registers:     0
  Number of PIO sites used: 14 + 4(JTAG) out of 105 (17%)
  Number of block RAMs:  0 out of 10 (0%)
  Number of GSRs:        0 out of 1 (0%)
  EFB used :       No
  JTAG used :      No
  Readback used :   No
  Oscillator used : No
  Startup used :    No
  POR :            On
  Bandgap :        On
  Number of Power Controller:  0 out of 1 (0%)
  Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
  Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
  Number of DCCA:  0 out of 8 (0%)
  Number of DCMA:  0 out of 2 (0%)
  Number of PLLs:  0 out of 2 (0%)
  Number of DQSDLLs:  0 out of 2 (0%)
  Number of CLKDIVC:  0 out of 4 (0%)
  Number of ECLKSYNCA:  0 out of 4 (0%)
  Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

综合上述内容可知，FPGA 寄存器资源占用了 4%，片选部分占用了 13%，逻辑运算单元同样占用了 13%。

6.2 Design Errors/Warnings

```
No errors or warnings present.
```

七、总结

本次实验很好地考察了我们对于状态机思想的掌握，也帮助我们对课堂理论知识有了更好的理解。总的来说，实验本身并不复杂，但是状态转移中设计状态数较多，且涉及多个模式的控制，这就需要我们对于状态机有较清晰的认识，在整理好时序逻辑、明确状态转移图

进行硬件描述设计，才能较为高效地完成本次任务。

同时，本实验与生活实际相结合，具有较高的趣味性，也让我们深刻地认识到数字电路在生活中的重大意义与广泛应用。

八、参考文献

[1]硬件描述语言与系统仿真课程实验要求书;
[2]实验一案件处理设计.pdf;
[3]STEP FPGA Base Board 底板硬件手册.

实验四　数字时钟设计

一、实验目的

1、了解数字时钟的工作原理；
2、学习和应用模块化编程；
3、了解 BCD 码的原理及应用；
4、对按键消抖，数码管显示，状态机等综合应用；
5、根据要求实现相应数字时钟的设计。

二、实验内容

该任务主要是基于小脚丫二代核心板完成简易数字时钟的设计。具体内容如下：
1、实现三个按键：模式按键、数字加键、数字减键；
2、实现四种模式：正常模式、调时模式、调分模式、调秒模式；
3、实现四个 LED 灯：对应指示数字时钟的四种模式；
4、实现两位数码管：

- ✓ 数字显示：调时模式下，显示时钟数值；调分模式下，显示分钟数值；调秒模式下，显示秒钟数值；正常模式下，按秒分时显示时分秒的数值；
- ✓ DP 显示：显示时钟数值时，左侧数码管 DP 点亮；显示分钟数值时，右侧数码管 DP 点亮；显示秒钟数值时，两个 DP 点都不亮；

三、设计思路

3.1　顶层设计

Clock.vhd 是实验 4 的顶层模块，该模块利用 component、port map 和子模块之间构建联系。在任务书中，clk_in 是时钟、rst_n_in 是重置信号、key_up 是增加时刻按键、key_down 是减小时刻按键、key_set 是模式切换按键、mode_led 是显示当前模式 LED 灯信号，rclk_out、sclk_out 和 ser_out 是 74HC595 的相关端口。总系统框图如图 4-1 所示。

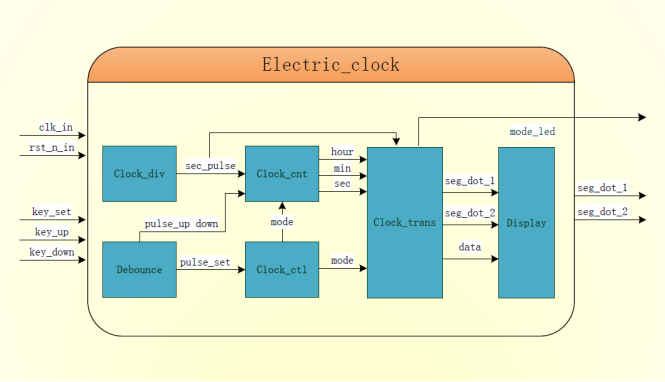

图 4-1　系统总框图

随后，我们利用 Netlist Analyzer 生成电路图，如图 4-2 所示。

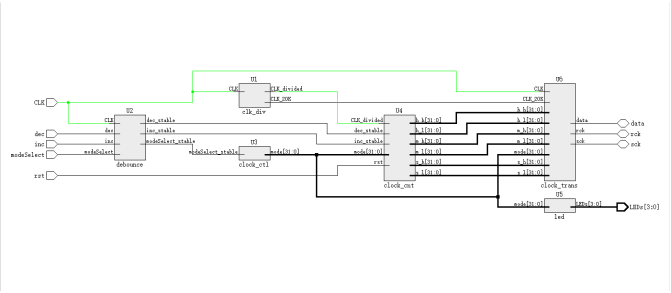

图 4-2 Lattice Diamond 生成的系统电路图

为了增强可读性，我们适当修改了部分变量名称，修改后的名称对应关系如表 4-1 所示。除顶层设计外，还包括 3.2~3.7 共计 6 个子模块。

表 4-1 变量名称替换关系

任务书中原变量名	实际变量名
clk_in	CLK
rst_n_in	rst
key_up	inc
key_down	dec
key_set	modeSelect
mode_led	LEDs
rclk_out、sclk_out、ser_out	RCK、SCK、DATA

3.2 按键去抖

在实际工况中，按键的运作绝非是理想情况下按之则变的，实际上，机械按键在按下和放开的时候往往都存在着短时间的抖动。根据我们在《工程实践与科技创新 II-A》课堂上所授知识，我们可以利用延迟稳定消抖法进行按键去抖操作，根据实验说明书，我们将延时设置为了 10 毫秒。具体图示见图 4-3。

举个例子，当系统探测到某输入信号在时钟上升沿改变时，计数器开始计时。当计数器计时到规定延迟时间，即 10 毫秒时，该模块输出该输入信号对应的稳定信号进行后续操作。为了方便辨识，我们在稳定信号的后面加上“_stable”以示区分。

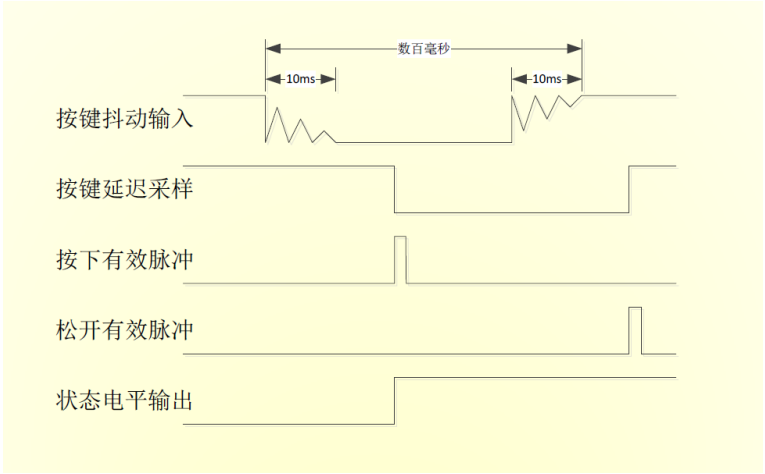

图 4-3 按键去抖图

3.3 模式切换

在任务要求中，我们需要同时实现四种功能，即正常计数模式、调时模式、调分模式、调秒模式。为了在板子上呈现出不同模式对应的内容，我们需要设计模式切换功能。

在实际操作中，我们选用按键计数进行模式切换。每一次按下 mode_select 按键，当前模式序数加 1，考虑到总共有 4 种模式，故而我们选择对按键按下的次数模 4 进行 4 种状态的循环切换。具体图示见图 4-4。

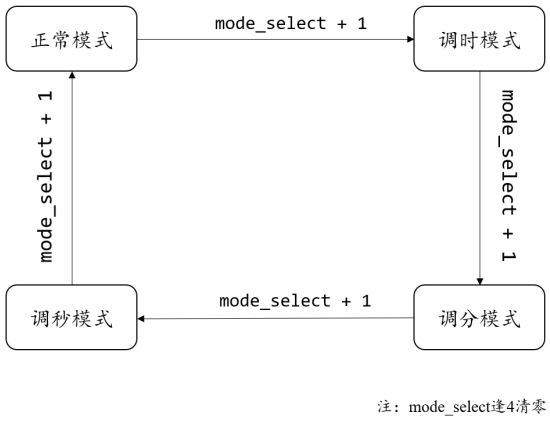

图 4-4 模式切换图

3.4 LED 显示

为了对当前模式进行可视化，我们选用 LED 进行增强显示。具体的模式、模式序号、LED 操作和 LED 的 std_logic_vector 对应内容如图 4-5 所示。

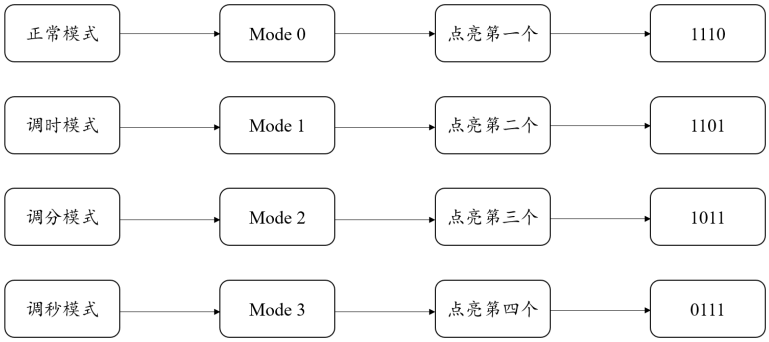

图 4-5 LED 显示对应关系图

3.5 时钟分频

为了方便后续操作，我们可以利用系统时钟自定义不同类型的计数器。纵观整个实验，我们需要为自动计时部分和 74HC595 数据存入及发送部分配置时钟。我们利用的是默认 12 兆赫兹的系统时钟。

对于自动计时部分，我们首先设置的是计数为 6M 次的计数器，分频至 1Hz，但后来我们选择计数为 60M，具体原因详见第 5 部分。

对于 74HC595 部分，我们设置的是计数为 300 次的计数器，分频至 20kHZ，用于等待 74HC595 的数据配置完成后再发送。

需要指出的是，考虑到 74HC595 接收数据是在时钟上升沿瞬态完成的，因而其时钟形态与系统时钟和自动计时部分的时钟形态均不一致，类似脉冲形。因而相关代码书写也略有不同，具体图示见图 4-6。

（1）10赫兹时钟信号波形

（2）74HC595时钟信号波形

图 4-6 分频时钟图

3.6 正常计时、调节时刻

在该部分，我们需要实现四种模式，四种模式又可以大致分为自动模式和手动模式两种。

在自动正常计时模式下，我们利用模 60 计数器，每逢上升沿进 1；同时，需要对秒为 59、分为 59、时为 23、时第二位为 9、分第二位为 9、秒第二位为 9 等需要进位的情况进行特殊化处理。

在手动模式下，我们利用 dec 或者 inc 对对应的时、分、秒进行减、增操作。重点注意 23：59：59、00：00：00 等多种特殊情况。以增、减秒为例，如图 4-7、4-8 所示。

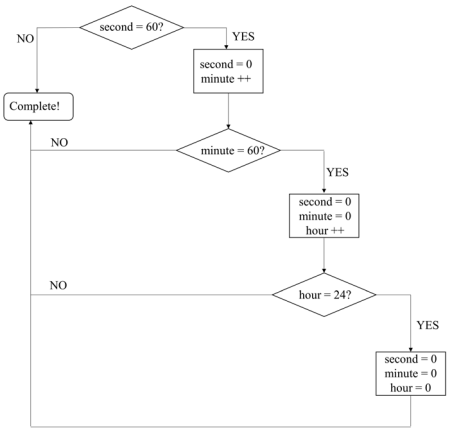

图 4-7 加秒操作流程图

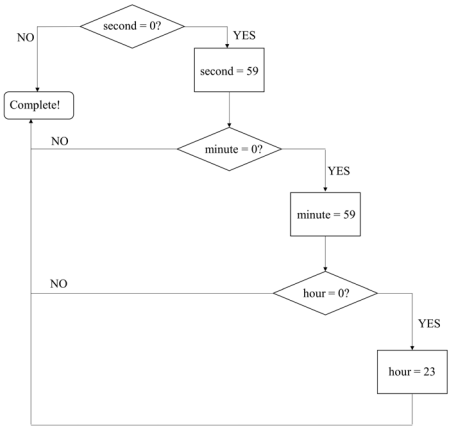

图 4-8 减秒操作流程图

3.7 数码管显示

数码管显示部分主要涉及 74HC595 位移缓存器的使用。通过查阅相关百度百科和 TI 官网 datasheet 可以了解到，74HC595 是串行输入、并行输出。其中，SCK 是输入时钟，RCK 是输出存储器锁存时钟线。SCK 上升沿时，数据寄存器的数据移位；SCK 下降沿时，移位寄存器数据不变。RCK 上升沿时，移位寄存器的数据进入数据存储寄存器；RCK 下降沿时，存储寄存器数据不变。在实验中，我们需要使用 2 块 74HC595，分别用于选通数码管、控制当前数码管显示内容。

74HC595 共有 3 种工作模式，分别为重置、配置数据、传输数据。在配置阶段，一共有 16 位数据，高 8 位中存储的是时间对应的 LED 字型编码，低 8 位存储的是点亮哪两个数码管（时、分或秒）的序号。在传输数据阶段，需要传输这 16 位数据，考虑到 74HC595 串行输出的特性，我们将这 16 位数据从最高位逐个输入，在 SCK 下降沿时放入数据等待，在 SCK 上升沿时数据进入 74HC595。串行输入结束后，立刻产生并行输出 RCK 信号，将信号传入数码管进行显示。具体过程如图 4-9 所示。

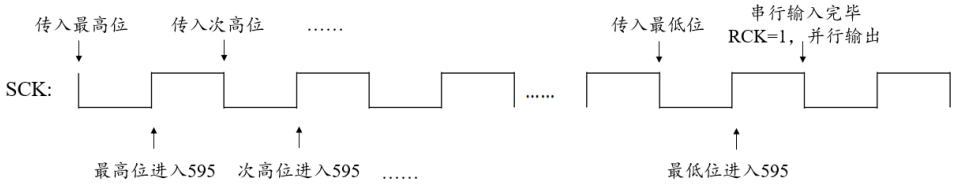

图 4-9 74HC595 串行输入、并行输出图示

四、作品使用说明

4.1 整体效果

默认状态下，时钟从 00：00：00 开始自动计时，第 1 个 LED 亮起。当按下 reset 键后，时钟清零；按下 modeSelect 键后，停止计时，若再按 inc 键，则时钟加 1，若按 dec 键，时钟减 1；再次按下 modeSelect 键，模式变为调分模式；再次按下 modeSelect 键，模式变为调秒模式，两者具体显示效果和调时模式类似。与此同时，调时模式下第二个 LED 亮起，调分模式下第 3 个 LED 亮起，调秒模式下第 4 个 LED 亮起。

4.2 管脚定义

表 4-2 管脚定义

变量	管脚	功能
RCK，SCK，DATA	H3，J2，G3	74HC595
rst	L14	重置
CLK	C1	内部时钟
LEDs	N13，M12，P12，M12	LED
modeSelect	M13	模式选择
dec，inc	N14，M14	减、增时刻

五、设计中碰到的问题及解决方案

5.1 增、减时刻按键不灵敏

究其根源，我们发现，在“正常计时、调节时刻”模块中，我们把自动时钟更新的频率和读取增、减按键的频率均设为了 1Hz，这就要求按下增、减按键必须经过 1Hz 时钟的一个上升沿。为此，我们将该部分时钟调快，从而快速获取增、减按键的信息；与此同时，在该模块中，我们继续定义了 1Hz 的时钟供自动计时模式调用。

六、FPGA 资源使用

6.1 Design Summary

```
Number of registers:    474 out of  4635 (10%)
   PFU registers:          474 out of  4320 (11%)
   PIO registers:            0 out of   315 (0%)
Number of SLICEs:       601 out of  2160 (28%)
   SLICEs as Logic/ROM:    601 out of  2160 (28%)
   SLICEs as RAM:            0 out of  1620 (0%)
   SLICEs as Carry:        348 out of  2160 (16%)
Number of LUT4s:        1201 out of  4320 (28%)
   Number used as logic LUTs:        505
   Number used as distributed RAM:     0
   Number used as ripple logic:      696
   Number used as shift registers:     0
Number of PIO sites used: 12 + 4(JTAG) out of 105 (15%)
Number of block RAMs:  0 out of 10 (0%)
Number of GSRs:        1 out of 1 (100%)
EFB used :        No
JTAG used :       No
Readback used :   No
Oscillator used : No
Startup used :    No
POR :             On
Bandgap :         On
Number of Power Controller:  0 out of 1 (0%)
Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
Number of DCCA:  0 out of 8 (0%)
Number of DCMA:  0 out of 2 (0%)
Number of PLLs:  0 out of 2 (0%)
Number of DQSDLLs:  0 out of 2 (0%)
Number of CLKDIVC:  0 out of 4 (0%)
```

```
Number of ECLKSYNCA:  0 out of 4 (0%)
Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

综合上述内容可知，FPGA 寄存器资源占用了 10%，片选部分占用了 28%，逻辑运算单元同样占用了 28%。

6.2 Design Errors/Warnings

```
No errors or warnings present.
```

七、总结

本次实验是基础实验的终章，是我们首次采用 VHDL 模块化的思想指导程序编写，对后续拓展实验、进阶实验部分有着先导性和全局性的重要意义。

八、参考文献

[1]硬件描述语言与系统仿真课程实验要求书;
[2]STEP-MXO2 V2 硬件手册;
[3]STEP FPGA Base Board 底板硬件手册;
[4]https://baike.baidu.com/item/74HC595/9886491?fr=aladdin.

实验五　基于无源蜂鸣器和矩阵按键电子琴系统设计

一、实验目的

本实验是课程拓展实验的主体，是中阶难度的 VHDL 编程项目。在本实验中，我们需要了解时钟计数分频功能、数组表示、控制标志的设置及蜂鸣器的运用。

二、实验内容

基于小脚丫 FPGA 开发板和按键、蜂鸣器的小钢琴设计：

1、 采用 FPGA+按键+无缘蜂鸣器实现小钢琴功能；

2、 7 个按键，分别对应 7 个音符，按下时蜂鸣器发出对应音调；

3、 设计一个音乐自动播放功能，根据存入的乐谱进行循环播放；

4、 其他个性化功能展示。

三、设计思路

3.1　顶层设计

此部分主要包含如下功能：

- ✓ 模式选择（Mode Switching）：可通过变量 auto1 来控制响单音，还是整曲演奏；
- ✓ 按键扫描（Keyboard Scanning）：扫描按键获取按键数值；
- ✓ 按键操作（Key Operations）：在单音模式下，按键 1~8 分别代表了 8 个不同的音高，按下一个按键即可让蜂鸣器以该音高鸣叫；在演奏模式下，按下 KEY4 则可以开始播放已存入的歌曲；
- ✓ 音乐切换（Music Switching）：切换下一首歌曲（已存入两首）；
- ✓ 音调演奏（Music Play）：将音调与对应的频率联系起来，并实现自动播放。

整体实现思路如图 5-1 所示。

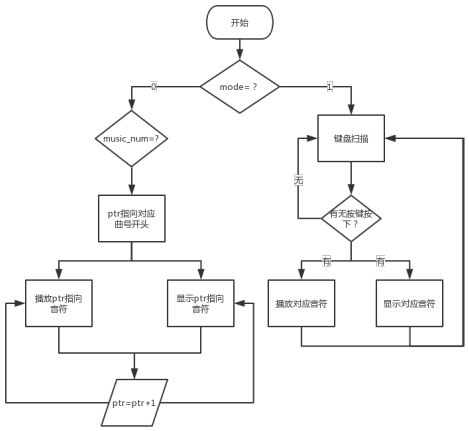

图 5-1 整体实现思路

我们利用 Netlist Analyzer 生成电路图，如图 5-2 所示。

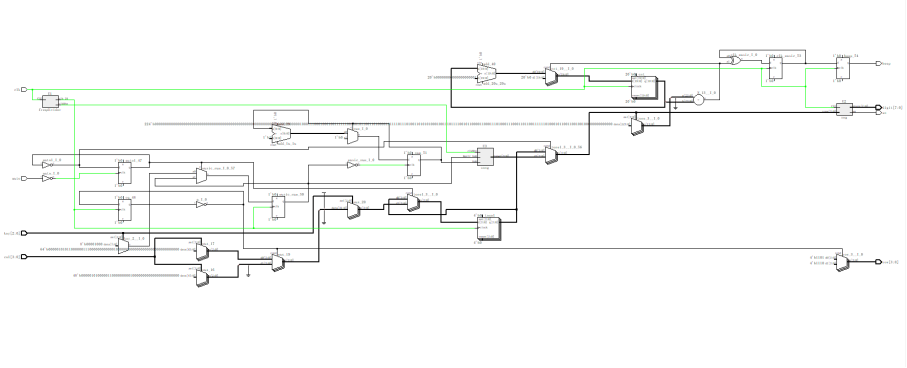

图 5-2 Lattice Diamond 生成的系统电路图

3.2 分频功能

为了程序可读性以及可维护性，我们将系统时钟分频模块单独进行封装，本实验中主要包括脉冲调制信号的分频以及 74HC595 数据存入及发送部分的分频。通过查阅小脚丫技术手册，得知系统时钟频率为 12MHz，我们使用 750k 次的计数器，分频后的时钟频率为 8Hz，输出为 0.125s，用于自动播放音符时单个音符的间隔；使用(20*30000-1)次计数器分频得到频率为 10Hz 的时钟，输出为 0.1s，用于按键操作时读取按键值。

3.3 数码管显示

数码管显示部分主要涉及 74HC595 位移缓存器的使用。通过查阅相关百度百科和 TI 官网 datasheet 可以了解到，74HC595 是串行输入、并行输出。其中，SCK 是输入时钟，RCK 是输出存储器锁存时钟线。SCK 上升沿时，数据寄存器的数据移位；SCK 下降沿时，移位寄存器数据不变。RCK 上升沿时，移位寄存器的数据进入数据存储寄存器；RCK 下降沿时，存储寄存器数据不变。

74HC595 共有 3 种工作模式，分别为重置、配置数据、传输数据。在配置阶段，一共有 16 位数据，高 8 位中存储的是时间对应的 LED 字型编码，低 8 位存储的是点亮哪两个数码管（时、分或秒）的序号。在传输数据阶段，需要传输这 16 位数据，考虑到 74HC595 串行输出的特性，我们将这 16 位数据从最高位逐个输入，在 SCK 下降沿时放入数据等待，在 SCK 上升沿时数据进入 74HC595。串行输入结束后，立刻产生并行输出 RCK 信号，将信号传入数码管进行显示。具体过程如图 5-3 所示。

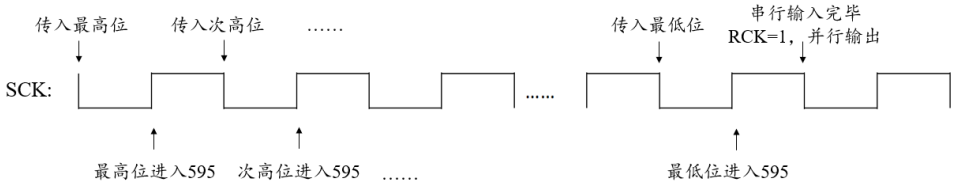

图 5-3 74HC595 串行输入、并行输出图示

3.4 已存入歌曲编码

根据钢琴各键对应音高的频率表，我们预先将音调所对应的频率编制好，以方便之后编曲时调用，如图 5-4 所示。我们主要选取的是中央“C”开始的小字一组及其之后 8 度。

钢琴上各音的名称和频率

颜色	白	黑	白	黑	白	白	黑	白	黑	白	黑	白
音名										A^2	$\#A^2$	B^2
频率										27.5	29.1	30.9
音名	C^1	$\#C^1$	D^1	$\#D^1$	E^1	F^1	$\#F^1$	G^1	$\#G^1$	A^1	$\#A^1$	B^1
频率	32.7	34.6	36.7	38.9	41.2	43.7	46.2	49.0	51.9	55.0	58.3	61.7
音名	C	#C	D	#D	E	F	#F	G	#G	A	#A	B
频率	65.4	69.3	73.4	77.8	82.4	87.3	92.5	98.0	103.8	110.0	116.5	123.5
音名	c	#c	d	#d	e	f	#f	g	#G	a	#a	b
频率	130.8	138.6	146.8	155.6	164.8	174.6	185.0	196.0	207.7	220.0	233.1	246.9
音名	c^1	$\#c^1$	d^1	$\#d^1$	e^1	f^1	$\#f^1$	g^1	$\#g^1$	$\mathbf{a^1}$	$\#a^1$	b^1
频率	261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	**440.0**	466.2	493.9
音名	c^2	$\#c^2$	d^2	$\#d^2$	e^2	f^2	$\#f^2$	g^2	$\#g^2$	a^2	$\#a^2$	b^2
频率	523.3	554.4	587.3	622.3	659.3	698.5	740.0	784.0	830.6	880.0	932.3	987.8
音名	c^3	$\#c^3$	d^3	$\#d^3$	e^3	f^3	$\#f^3$	g^3	$\#g^3$	a^3	$\#a^3$	b^3
频率	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
音名	c^4	$\#c^4$	d^4	$\#d^4$	e^4	f^4	$\#f^4$	g^4	$\#g^4$	a^4	$\#a^4$	b^4
频率	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
音名	c^5											
频率	4186											

图 5-4 88 键钢琴各键对应频率

然后通过查阅一些简单歌曲的简谱，我们就可以实现对歌曲的编码与播放。相关曲目五线谱如图 5-5、5-6 所示，系组员手动打谱。本次实验我们选择了《小星星》和《天空之城》。

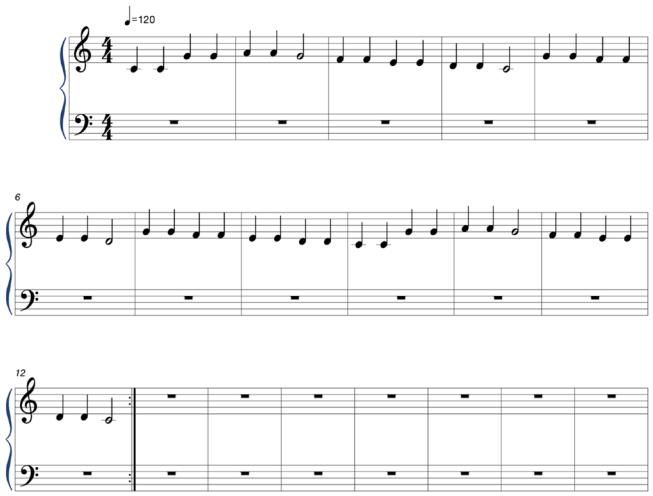

图 5-5 《小星星》五线谱

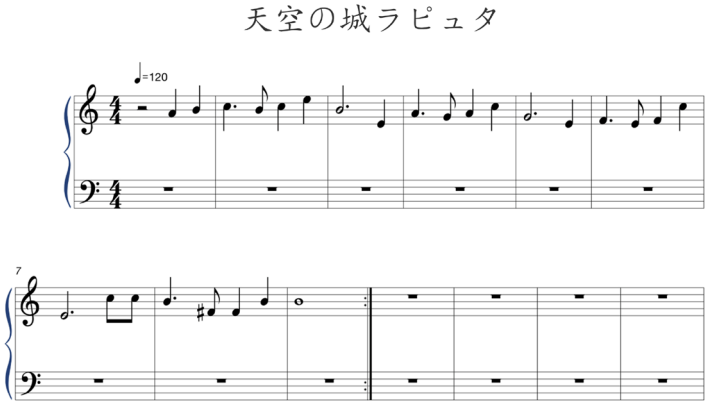

图 5-6 《天空之城》五线谱

四、作品使用说明

4.1 整体效果

在按键演奏模式下，每按下一个按键，蜂鸣器可以正确的鸣叫出正确的音调，七段显示器同时显示相对应的音调，本模式支持 8 个音；按下 auto 可以实现音乐的自动播放，在自动播放模式下，默认播放《小星星》，按下 key2 键后可以更换播放曲目为《天空之城》。

4.2 管脚定义

表 5-1 管脚定义

变量	管脚	功能
clk	C1	内部时钟
auto	L14	切换模式
key[0]	M13	按下发出小字一组“do”
key[1]	M14	按下发出小字二组“do”
key[2]	N14	切换自动播放曲目
row	N8, P8, N7, P7	矩阵按键-行
col	L3, N5, P6, N6	矩阵按键-列
beep	P13	蜂鸣器

五、设计中碰到的问题及解决方案

5.1 自动播放

一开始尝试过很多办法，比如用一个大数组去存放音乐的音调，但是都失败了，最后还是用最原始的办法，针对时钟循环定义每一次的音调，最终得以解决。

六、FPGA 资源使用

6.1 Design Summary

```
Number of registers:    108 out of  4635 (2%)
   PFU registers:          108 out of  4320 (3%)
   PIO registers:            0 out of   315 (0%)
Number of SLICEs:       150 out of  2160 (7%)
   SLICEs as Logic/ROM:    150 out of  2160 (7%)
   SLICEs as RAM:            0 out of  1620 (0%)
   SLICEs as Carry:         50 out of  2160 (2%)
Number of LUT4s:        298 out of  4320 (7%)
   Number used as logic LUTs:        198
   Number used as distributed RAM:     0
   Number used as ripple logic:      100
```

```
   Number used as shift registers:     0
Number of PIO sites used: 23 + 4(JTAG) out of 105 (26%)
Number of block RAMs:  0 out of 10 (0%)
Number of GSRs:        1 out of 1 (100%)
EFB used :         No
JTAG used :        No
Readback used :    No
Oscillator used :  No
Startup used :     No
POR :              On
Bandgap :          On
Number of Power Controller:  0 out of 1 (0%)
Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
Number of DCCA:  0 out of 8 (0%)
Number of DCMA:  0 out of 2 (0%)
Number of PLLs:  0 out of 2 (0%)
Number of DQSDLLs:  0 out of 2 (0%)
Number of CLKDIVC:  0 out of 4 (0%)
Number of ECLKSYNCA:  0 out of 4 (0%)
Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

综合上述内容可知，FPGA 寄存器资源占用了 2%，片选部分占用了 7%，逻辑运算单元同样占用了 7%。

6.2 Design Errors/Warnings

```
No errors or warnings present.
```

七、总结

本次实验是本学期《硬件描述语言与系统仿真》课程实验部分较为有趣的一次实验，在硬件工程的基础上添入了音乐艺术，在编制歌曲编码时感受到了巴赫十二平均律和朱载堉律学的魅力。

八、参考文献

[1] PCF8591 Datasheet;
[2] 硬件描述语言与系统仿真课程实验要求书;
[3] 基于小脚丫 FPGA 开发板和按键、蜂鸣器的小钢琴设计.

实验八　基于小脚丫 FPGA 的直流电压测量装置

一、实验目的

学习模数转换器 ADC 的相关知识，串行（I2C 接口）ADC 芯片 PCF8591 的驱动设计，同时学习二进制数转换 BCD 码的设计方法和 LCD 显示驱动方法。

- ✓ 学习模数转换器 ADC 的相关知识；
- ✓ 学习串行（I2C 接口）ADC 芯片 PCF8591 的驱动设计；
- ✓ 学习二进制数转换 BCD 码的设计方法；
- ✓ 学习 LCD 显示驱动方法；
- ✓ 完成简易电压表设计实现。

二、实验内容

基于 STEP-MXO2 V2 FPGA 核心板和 STEP BaseBoard V2.0 底板，通过底板上的串行模数转换器 ADC 芯片测量可调电位计输出电压，并将电压信息显示在核心板或底板的数码管或 LCD 屏上。通过 FPGA 编程驱动串行 ADC 芯片，得到数字量化的电压信息，将量化的数字信息转换成 BCD 码形式，同时驱动独立数码管或 LCD 将电压值显示出来。系统连接如图 8-1 所示。

具体而言，0~3.3V 的直流电压加在串行 ADC 的模拟输入端，串行 ADC 将直流电压转换为 8 位的数字量，0~3.3V 的直流模拟电压得到 0~255 的数字量；小脚丫 FPGA 通过内部产生的 I2C 时序将 ADC 转换的数据读取到 FPGA 内部的寄存器，并将串行的二进制数据转换成 8 位并行的数据。转换后数据的以三种方式显示，一是通过点亮 8 个 LED 显示电压的相对强度，二是通过底板上连接的 4 个 7 位数码管以二进制的形式（0~255）或直流电压的方式（0~3.300）显示，三是通过扩展板上连接的 LCD 显示屏，在 LCD 上显示电压值。

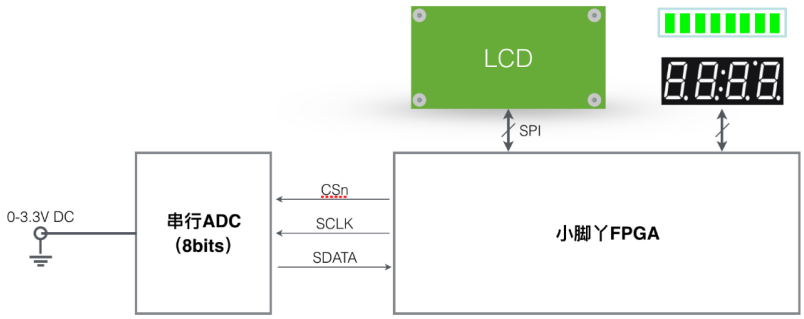

图 8-1 测量装置模块示意图

三、设计思路

3.1　顶层设计

在实验 4 模块化设计的成功经验之上，本次实验我们依然采用总——分的结构。具体层次架构如图 8-2 所示。

- ✓ Voltage 将各模块汇总，利用各模块实现具体功能。
- ✓ ADC_I2C：驱动 I2C 总线与 PCF8591 芯片通信，获取二进制电压值；
- ✓ LED_display：8 个 LED 灯以“亮-灭”的形式显示二进制电压值；
- ✓ bin_2_BCD：二进制数值转换为 BCD 码；
- ✓ seg_display：利用 74HC595 芯片在数码管上显示模拟电压值；
- ✓ LCD_ram：LCD_ram 模块根据当前的 bcd 码值与当前需要写入 LCD 显示帧存的行地址，返回一行 128 个像素的点亮情况，与显示数字或图片点阵中的该行相对应;
- ✓ LCD_display：根据 BCD 码值，驱动 LCD 显示相应的数值。

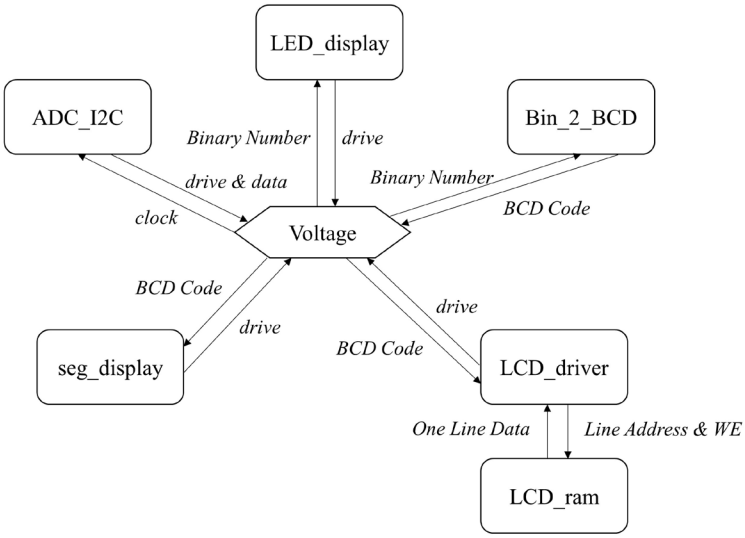

图 8-2 整体架构图

3.2 I2C 总线与 PCF8591 AD/DA 芯片的通信

该部分主要的代码实现部分为 I2C_ADC.vhd。阅读《直流电压测量装置设计指导》第 3.1 节可知，PCF8591 芯片在进行单端 AD 采样时，其采样值与模拟电压满足如图 8-3 所示的关系。由于在本板卡中 Vref=3.3V，根据采样得到的 8 比特二进制整数（十进制即 0~255）可以由此换算得到模拟电压值，并将其显示在数码管或 LCD 屏上。

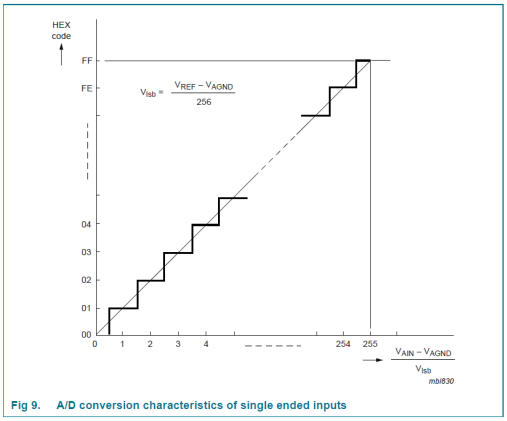

图 8-3 PCF8591 芯片单端 AD 采样时的采样值与模拟电压的关系

为了使用 I2C 总线与 PCF8591 通信，需要满足 I2C 总线的通信协议，这一部分在参考文献《PCF8591 Datasheet》中第 9.2 节，由图 8-4 可知，当在 I2C 总线空闲时，数据线和时钟线都处于高态。当时钟为高态，数据线从高态到低态作为起始条件（S）；当时钟为高态，数据线从低态到高态作为终止条件（P）。

Both data and clock lines remain HIGH when the bus is not busy.

A HIGH-to-LOW transition of the data line while the clock is HIGH is defined as the START condition - S.

A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the STOP condition - P (see Figure 12).

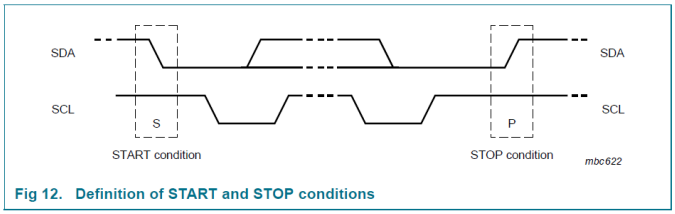

Fig 12. Definition of START and STOP conditions

图 8-4 通信开始和停止条件图

一个典型的 I2C 总线时序图如图 8-5 所示。

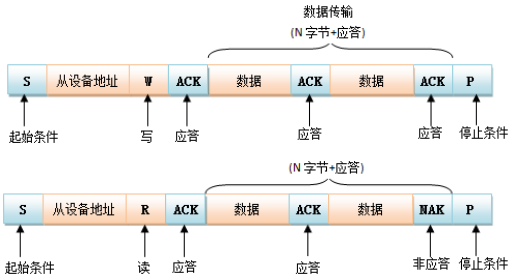

图 8-5 主设备往从、从从设备中读写数据图示

PCF8591 内部维护一个 8 比特的 Control Byte 寄存器来存储功能配置数据，其中每一个比特都对应一些功能设置，在板子上电后，寄存器自动初始化为全 0，因此需要配置该数据来实现特定要求。

根据以上内容，本设计中我们需要两次通信过程。第 1 次为配置数据，具体为：开始–写寻址–读响应–写配置数据–读响应–结束；第 2 次为读 ADC 数据，具体为：开始–读寻址–读响应–[读 ADC 数据–写响应–]循环读-结束。读出的 ADC 数据为一个 8 比特整数，经过处理后使用数码管，LCD 显示屏等器件进行显示。状态转换图如图 8-6 所示。

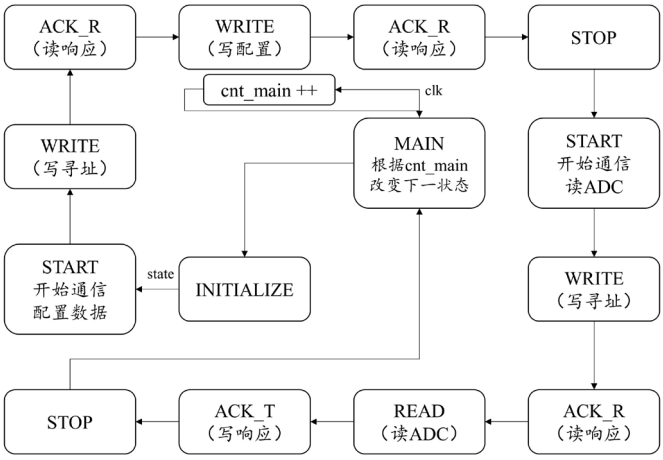

图 8-6 ADC 状态转移图

3.3 LED 显示二进制值

LED 显示对应二进制值的操作较为简单，我们构造一个 OUT 型的 STD_LOGIC_VECTOR (7 DOWNTO 0)，和 ADC 二进制数值相连（全部取反），就能够达到显示效果。

3.4 二进制转 BCD 码

BCD 码中 4 位只有十个数，但是在 4 位二进制中 4 位有 0~15 共十六个数，因此需要经过转换算法，将二进制数转换成 BCD 码后才能在数码管上显示。我们可以选用加三移位法。

具体流程如图 8-7 所示。首先，左移要转换的二进制码 1 位；其次，左移之后，BCD 码分别置于百位、十位、个位；然后，如果移位后所在的 BCD 码列大于或等于 5，则对该值加 3；最后，继续左移的过程直至全部移位完成，从而得到 BCD 码。

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1
Shift 1			1	1 1 1 1	1 1 1
Shift 2			1 1	1 1 1 1	1 1
Shift 3			1 1 1	1 1 1 1	1
Add 3			1 0 1 0	1 1 1 1	1
Shift 4		1	0 1 0 1	1 1 1 1	
Add 3		1	1 0 0 0	1 1 1 1	
Shift 5		1 1	0 0 0 1	1 1 1	
Shift 6		1 1 0	0 0 1 1	1 1	
Add 3		1 0 0 1	0 0 1 1	1 1	
Shift 7	1	0 0 1 0	0 1 1 1	1	
Add 3	1	0 0 1 0	1 0 1 0	1	
Shift 8	1 0	0 1 0 1	0 1 0 1		
BCD	2	5	5		

图 8-7 二进制数转换 BCD 码步骤示意图

3.5 数码管显示 BCD 码

数码管显示部分主要涉及 74HC595 位移缓存器的使用。通过查阅相关百度百科和 TI 官网 datasheet 可以了解到，74HC595 是串行输入、并行输出。其中，SCK 是输入时钟，RCK 是输出存储器锁存时钟线。SCK 上升沿时，数据寄存器的数据移位；SCK 下降沿时，移位寄存器数据不变。RCK 上升沿时，移位寄存器的数据进入数据存储寄存器；RCK 下降沿时，存储寄存器数据不变。

74HC595 共有 3 种工作模式，分别为重置、配置数据、传输数据。在配置阶段，一共有 16 位数据。在传输数据阶段，需要传输这 16 位数据，考虑到 74HC595 串行输出的特性，我们将这 16 位数据从最高位逐个输入，在 SCK 下降沿时放入数据等待，在 SCK 上升沿时数据进入 74HC595。串行输入结束后，立刻产生并行输出 RCK 信号，将信号传入数码管进行显示。具体过程如图 8-8 所示。

此外，当数码管显示为 0~3.3 时，需要点亮小数点；当显示二进制数值时，则不需要。

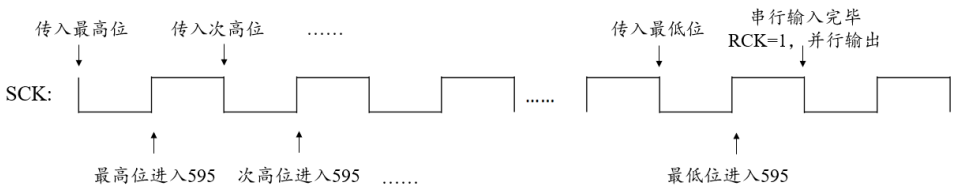

图 8-8 74HC595 串行输入、并行输出图示

3.6 LCD 显示屏存储

LCD_ram 存储了 din 对应的 LCD 显示。din 是 20 位的 BCD 码，我们首先将其分成 5 段，每段 4 位对应一个十进制位，所以该十进制数有 1 个小数点和 5 位有效位数，对应 display4_num、display_dot、display3_num、display2_num、display1_num 和 display0_num 这 6 个长度位 4 的数组；随后，利用规模为 16*16 的数字点阵二维数组 num 表征所有可能字符，即 0~9 和小数点的数字点阵表示，将 num 和前述的 display_num 系列数组构建对应关系。最后，考虑到 display_num 不同，最终在 LCD 显示屏上显示的位置也不一样，故而我们利用反向扫描的方式构造 ram 二维数组存储所有可能。具体显示对应位置如图 8-9 所示，数字点阵举例如图 8-10 所示。

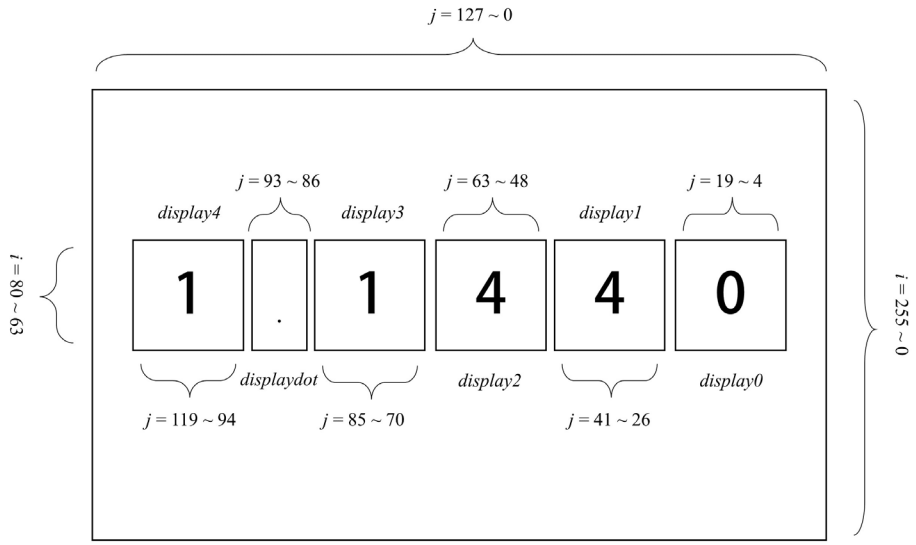

图 8-9 LCD 屏幕显示对应位置图示

```
num4(0)  <= "0000000001111000";
num4(1)  <= "0000000011111000";
num4(2)  <= "0000000110111000";
num4(3)  <= "0000001110111000";
num4(4)  <= "0000011100111000";
num4(5)  <= "0000111000111000";
num4(6)  <= "0001110000111000";
num4(7)  <= "0011111111111111";
num4(8)  <= "0111111111111111";
num4(9)  <= "1111111111111111";
num4(10) <= "0000000000111000";
num4(11) <= "0000000000111000";
num4(12) <= "0000000000111000";
num4(13) <= "0000000000111000";
num4(14) <= "0000000000111000";
num4(15) <= "0000000000111000";
```

图 8-10 数字“4”的 16*16 数字点阵表示

3.7 LCD 显示屏显示 BCD 码

该部分使用 SPI 总线与 ST7735S 通信，LCD_driver 通过地址信号的扫描，获取图像显存中对应 LCD 屏幕中一行的像素状态值，将 LCD_ram 中存储的图像行像素数据依次写入

LCD 显示帧存中，从而使得 LCD 显示屏显示相对应的图像。

LCD_driver 模块的状态转移图如图 8-11 所示。显然，一共有 6 种工况。具体内容如下：

- ✓ IDLE：无工作。可用于重启；
- ✓ MAIN：主状态。可用于调配其他状态；
- ✓ INIT：初始化。通过 LCD 输入特定指令启动并设置显示范围和格式；
- ✓ SCAN：扫描态。给 LCD_ram 传输使能与地址信号，得到每一行的数据，并定位目标写入数据的位置，将像素状态转化成 2 位 8 比特的颜色数据，传递给写状态；
- ✓ WRITE：写状态。以顺序方式将 8 比特数据按总线形式驱动对应端口；
- ✓ DELAY：延时态。通过计数清零的方式实现延时。

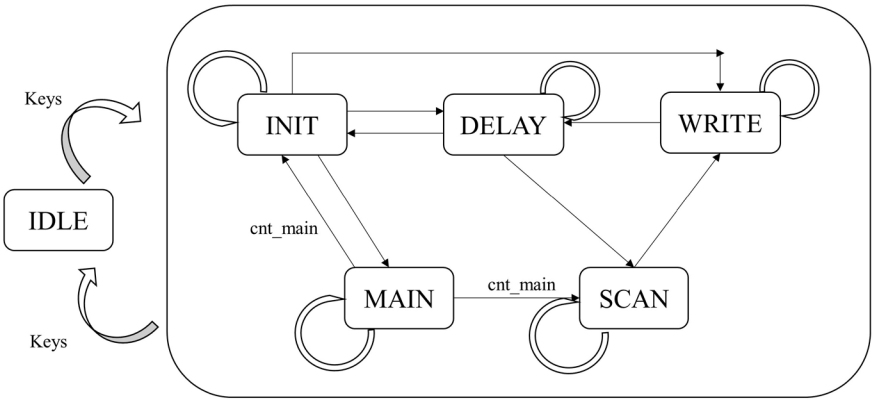

图 8-11 LCD_driver 模块的状态转移图

该模块启动时，会首先进入主状态，并会自动调用初始化状态，在初始化状态中会进行复位延时，并执行写入初始化指令、延时等操作，与写状态 WRITE 和延时状态 DELAY 切换频繁，这两个状态可以看作是通过状态机的架构实现了函数的功能。执行过一次之后再次返回主状态时会切换到扫描状态 SCAN，进入 SCAN—WRITE—DELAY 循环，重复不断地将 ram 里的数据以总线形式输入到 LCD 里的显存中。

3.8 系统总括

最后，我们利用 Netlist Analyzer 生成电路图，如图 8-12 所示。

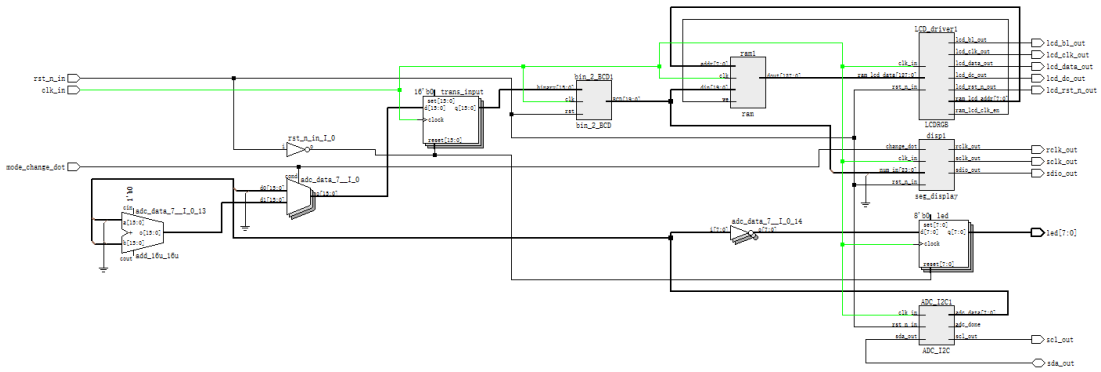

图 8-12 Lattice Diamond 生成的系统电路图

四、作品使用说明

4.1 整体效果

旋转实验底板左上方的 A/D、D/A 旋钮，数码管会显示当前 ADC 采样值；按下 mode_change_dot 按钮后，数码管显示二进制值，此时，8 个 LED 灯珠也对应显示该二进制值。与此同时，LCD 显示屏上显示内容一直和数码管显示内容保持一致。

4.2 管脚定义

表 8-1 管脚定义

变量	管脚	功能
clk_in	C1	内部时钟
mode_change_dot	L14	改变显示内容（255/3.3）
rst_n_in	M13	重置
lcd_bl_out lcd_dc_out lcd_rst_n_out lcd_clk_out lcd_data_out	J13 J14 K12 K13 K14	LCD 相关指定管脚
led[0]~led[7]	N13~P9	LED 显示二进制值
scl_out sdio_out rclk_out sclk_out sda_out	C8 G3 H3 J2 B8	74HC595 相关指定管脚

五、设计中碰到的问题及解决方案

5.1 LCD 显示屏数字是纵向、偶尔显示不全

其实，在最初开展实验时，我们发现数字不仅是纵向的而且是反过来的。参考往届学长报告和代码，我们发现 LCD 的扫描方式是从右向左的，在代码书写中，均要“反向”，即原先的行数 *i*，要改成 255-*i*；原先的列数 *j*，要改成 127-*j*，所以反过来是意料之中的。至于纵向，则是代码书写错误，行和列在 LCD_ram 的两层 loop 中搞混了。

对于显示不全问题，参考往届学长报告和代码后，我们发现，在 scan 过程中，横、纵坐标扫描的范围和初始化中设定的 *x*、*y* 数据是对应的，这样就能够解决显示不全的问题。

六、FPGA 资源使用

6.1 Design Summary

```
Number of registers:    487 out of  4635 (11%)
```

```
   PFU registers:          487 out of  4320 (11%)
   PIO registers:            0 out of   315 (0%)
Number of SLICEs:       996 out of  2160 (46%)
   SLICEs as Logic/ROM:    996 out of  2160 (46%)
   SLICEs as RAM:            0 out of  1620 (0%)
   SLICEs as Carry:         79 out of  2160 (4%)
Number of LUT4s:        1987 out of  4320 (46%)
   Number used as logic LUTs:        1829
   Number used as distributed RAM:      0
   Number used as ripple logic:       158
   Number used as shift registers:      0
Number of PIO sites used: 21 + 4(JTAG) out of 105 (24%)
Number of block RAMs:  0 out of 10 (0%)
Number of GSRs:        1 out of 1 (100%)
EFB used :         No
JTAG used :        No
Readback used :    No
Oscillator used :  No
Startup used :     No
POR :              On
Bandgap :          On
Number of Power Controller:  0 out of 1 (0%)
Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
Number of DCCA:  0 out of 8 (0%)
Number of DCMA:  0 out of 2 (0%)
Number of PLLs:  0 out of 2 (0%)
Number of DQSDLLs:  0 out of 2 (0%)
Number of CLKDIVC:  0 out of 4 (0%)
Number of ECLKSYNCA:  0 out of 4 (0%)
Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

综上所述，FPGA 寄存器资源占用了 11%，片选和逻辑运算单元占用了近半，体现出本实验的难度与工程量。

6.2 Design Errors/Warnings

```
No errors or warnings present.
```

七、总结

本次实验是本学期《硬件描述语言与系统仿真》课程实验部分的终章，代码工程量大、知识点覆盖全面，全方位考察了学生 VHDL 语言的掌握情况和熟练程度，对后续高阶硬件类课程的深入学习，电子设计类学科竞赛的提前锻炼有着重要意义。

八、参考文献

[1] PCF8591 Datasheet;
[2] ST7735S_V1.1_20111121datasheet;
[3]直流电压测量装置设计指导;
[4]数字系统仿真 VHDL 设计—第 049 组设计报告-2016 级组长姚春晖.