# 上海大学

SHANGHAI UNIVERSITY

2022-2023 学年夏季学期

# 课程报告

# 《计算机硬件综合大型作业报告》

小组序号：2-5

项目名称：出租车计价器设计

指导老师：张云华

计算机工程与科学学院

报告日期 2023 年 7 月 3 日

# 目录

# 项目预研报告

## 1 项目意义

在本次夏季实践课程中，我们面临一项具有挑战性的任务，需要进行硬件项目的设计、仿真测试和运行。这个任务要求我们充分运用在大二学年中所学习的与计算机硬件相关的知识，并且利用电路设计软件 Quartus II 来完成相应电路的设计。在这个项目的完成过程中，我们不仅需要回顾课程中所学的知识，还需要积极主动地在互联网上学习新的技术，以提高我们对计算机系统的整体理解。除此以外，在解决实际问题的过程中，我们能够深入实践，加深对知识的理解和掌握。

平时，我们在课堂上的项目主要涉及软件层面的问题，对硬件层面的问题讨论较少，而这次的计算机硬件综合作业则锻炼了我们在这硬件方面的能力。通过这个项目，我们得到了弥补短板的机会，并培养了电路分析与设计的能力，收获颇丰。

## 2 项目设计要求

本次我们小组负责的任务是设计一个出租车计价器。根据生活中的实际情况，我们的项目要求可以总结为以下几点：

- 出租车计价器包含有设置模式和运行模式；
- 在设置模式中，可以进行对起步价格、起步里程和里程单价的设置；
- 在运行模式中，当公里数在起步里程内，价格不变，显示为起步价格，公里数逐渐增加；当公里数超出起步里程以后，按照里程单价，价格逐渐随着公里数增加而变大。此外，在运行模式中包含有暂停功能，暂停后，公里数和价格数都不再变化。清零功能，公里数重置为 0，价格重置为起步价格。
- 使用 6 位数码管和若干个灯作为显示，其中 3 位数码管作为公里数的显示，3 位数码管作为价格的显示（包含一位小数），当公里或者价格超出数码管的显示范围时，使用灯作为数据溢出的显示。

## 3 设计分析

项目的设计阶段是整个过程中的关键一环，需要我们仔细分析项目需求，根据要求选择合适的硬件组件和电路设计方案。这要求我们充分运用课程中学到的知识，考虑到电路的性能、功耗、稳定性等因素。我们需要进行充分的调研和实验，以确保我们的设计能够满足项目的要求，并且具有良好的可靠性和稳定性。

Quartus II 6.0 是一款由 Altera 公司（现为英特尔子公司）开发的集成电路设计软件。Quartus II 是一种用于 FPGA（现场可编程门阵列）和 CPLD（可编程逻辑器件）设计的先进工具，它提供了全面的设计、仿真、优化和布局布线功能。

本次项目我们小组使用 Quartus II 6.0 软件来进行电路的设计，模拟和仿真。

# 实践项目报告

## 1 概要设计

对于本次夏季实训项目，我们要使用 Quartus II 设计一个模拟出租车计价器的电路，由于这是一个项目的设计，任务比较复杂，难以一次性解决，所以我们想到的是将任务分解成多个模块，然后将问题逐个突破。

• 系统包含两种模式，一种是设置模式，另外一种是运行模式

我们可以单独使用一个输入 SETTING_EN，当输入为高电平时，系统进入设置模式；当输入为低电平时，系统进入运行模式。

• 在设置模式中，可以设置起步距离，起步价和超出起步距离后的每公里单价。

在这一个任务中，我们希望系统启动后进入设置模式，虽然实验模拟箱在断电后所有的数据会清除，但是在系统运行时，我们设置好的数据可以保存在系统中，不会随着清除功能时公里数和价格数的重置而丢失，所以需要一个芯片存储输入的数据。在本次项目中使用的是 73LS373 锁存器，这是一种常用的 8 位数据/地址锁存器。

同时，我们希望可以选择设置的对象，从而实现对起步距离，起步价格和每公里单价的依次设置。这里我们使用的是 74LS139 双 2-4 译码器，同时引入 SETTING_SB 和 SETTING_SA 两个开关，当其为 00，01 时，对起步价格进行设定；为 10 时对每公里价格进行设定；为 11 时，对起步距离进行设置。

这样，我们就解决了数据的存储问题。

• 进入运行模式以后，初始时，公里数显示为 0，价格数显示为起步价格，对于每一个时钟脉冲，在起步公里内，公里数增加 0.1 公里，价格数不发生变化；

在这个问题中，由于我们在设计时使用一个 74LS373 数据锁存器来存储起步公里的数值，同时起步公里要包含一位小数，所以起步公里的设置范围为 0 公里到 9.9 公里。

两位十进制数据使用八位二进制数据来表示，如果我们要知道当前的公里数是否超出了起步公里，我们就需要数值比较器来进行数据的比较。这里我们选用了两片 7485 芯片，一片用来比较公里的个位，另外一片用来比较公里的小数位，由于起步公里通常在 10 公里以内，所以这种方法比较合理。

当前公里数如何表示呢，在系统的内部应该如何存储呢？我们使用了 74LS160 芯片表示公里数。一个 74LS160 来表示公里数的一位，从小数位到百位，一共需要 4 个 74LS160 芯片，由于 74LS160 本身的特性，可以实现公里数随时间脉冲逐渐增加。

• 超出起步公里以后，公里仍每次增加 0.1 公里，价格数按照每公里单价的 1/10 进行累加。公里数和价格数显示在数码管上。

在当前公里数超出起步距离以后，价格开始发生变化。由于价格数为 3 位数码管显示，每公里单价在设置时通常保留一位小数，例如 3.2 元/公里，而公里数是每个时钟脉冲增加 0.1 公里的，这时价格数需要每次增加 0.32，如果对价格的设置只有一位小数，后面在利用

累加器时会发生精度的缺失，所以在设置时，实际上我们设置的价格为每运行 0.1 公里的价格，这个数据的范围是 0.00-0.99 元/0.1 公里。价格数要设置好 2 位小数位，在运行时数码管显示十分位，个位和十位。

• 设置溢出灯：当数据超出数码管的显示范围时，溢出灯亮起，每溢出一次亮起一个灯。

举一个例子，当价格数为 98.2 时，假设下一个时钟脉冲到来后，价格数发生溢出，这时价格的百位无法显示，但是我们可以用灯来表示。百位为 1 时，亮起一个灯；下一次溢出，百位为 2，亮起两个灯……只需将百位的输出连接一个芯片，芯片的输出和状态灯相连接，就可以实现溢出灯这一功能。

• 清除功能：公里数重置为 0，价格数重置为初始价格。

使用一个 CLR 输入，连接到各计数器的 CLRN 端，从而实现对公里数的重置。然后公里数会和比较器的输入（起步距离）进行比较，这时选中起步价格作为价格数，从而实现价格数的重置。

• 暂停功能：暂停公里数和价格数的计数。

不妨将脉冲 CLK 和暂停信号 PAUSE 进行与运算然后输入到各累加器的 CLK 端，CLK 有效而 PAUSE 无效时，系统正常工作，当 PAUSE 有效时，各累加器得到的信号无效，即停止累加。

## 2 详细设计

### 2.1 设置模式

**表 2.1 74139 双 2-4 译码器功能表**

| Inputs | | | Outputs | | | |
|---|---|---|---|---|---|---|
| Enable | Select | | | | | |
| G | B | A | Y0 | Y1 | Y2 | Y3 |
| H | X | X | H | H | H | H |
| L | L | L | L | H | H | H |
| L | L | H | H | L | H | H |
| L | H | L | H | H | L | H |
| L | H | H | H | H | H | L |

74LS139 是一种双 2-4 线译码器。其功能表如**表 2.1** 所示。

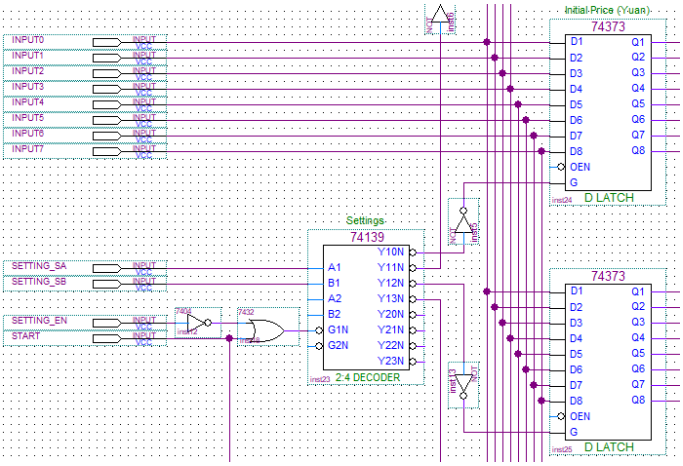



**图 2.1 设置模式选择电路（部分）**

设置模式选择电路如**图 2.1** 所示，74139 的 G1N 连接了一个或门，当 START 为低电平且 SETTING 为高电平时，或门运算的结果为低电平，系统进入到设置模式，这时可根据 SETTING_SB 和 SETTING_SA 的值选择锁存器。

以每 0.1 公里运行价格为例，此时高位 B1 为高电平，低位 A1 为低电平。根据真值表，74LS139 的输出引脚中，只有 Y12N 为低电平，其余引脚均为高电平。输出经过非门连接到 74LS373 数据锁存器的使能端 G 上面。

74373 是一种 8 位数据锁存器，其中 D1-D8 是数据的输入端，Q1-Q8 是数据的输出端。G 是数据锁存控制端，当 G 的值为高电平时，数据的输出同输入端，当 G 的值从高电平变为低电平时，数据输入到锁存器中，OEN 是输出允许端，为低电平时，表示输出允许，所以可以不接线。

前面提到 Y12N 输出的低电平经过非门后变为高电平打入到 G，这时锁存器接受输入的 8 位数据。我们只需要将需要设置的数据以 8 位二进制的形式，通过 input0-input7 开关进行输入即可，当 G 从 1 变为 0 后，数据完成打入。

此外，其他的锁存器的设置类似，只需要改变 SETTING_SB 和 SETTING_SA 的值，之后从开关输入数值。

当 SETTING_SB 和 SETTING_SA 的值为 00，进入对起步价格的设置，由于只能设置 8 位二进制数值，而对于起步价格来说，例如 13.2 元需要 12 位二进制数值来表示。因此在这种情况下，我们输入的 8 位二进制数值打入到锁存器中，我们规定此时锁存器存储的是起步价格的个位和十位。

当 SETTING_SB 和 SETTING_SA 的值为 01 时，这时进入对起步价格的小数位的设置，因为只需一位小数位，所以只需要打入 4 位二进制数，如**图 2.2** 所示。

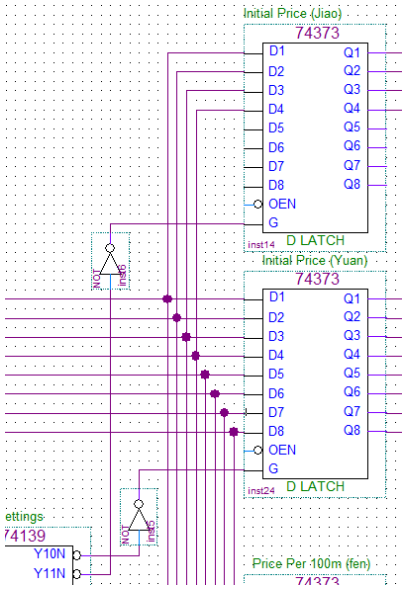



**图 2.2 起步价设置电路**

当 SETTING_SB 和 SETTING_SA 为 10 时，设置每 0.1 公里单价。前面已经详细展开，不再赘述。

当 SETTING_SB 和 SETTING_SA 为 11 时，设置起步公里，Y13N 连接到第 4 片 74373 的数据控制端 G 上面。同理进行设置，设置个位和小数位。

下面给出模拟出租车计价系统数据的设置范围：

| | 起步公里 | 起步价格 | 每 0.1 公里单价 |
|---|---|---|---|
| 范围 | 0-9.9 | 0-99.9 | 0-0.99 |

## 2.2 运行模式（起步公里以内）

在 2.1 中完成了对初始变量的初值的设定，接下来可以模拟运行我们的出租车计价系统。

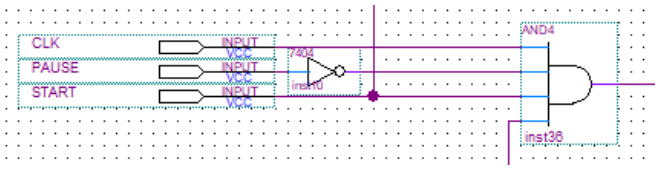



**图 2.3 运行电路输入**

运行电路输入如**图 2.3** 所示，START 置为 1，CLK 接自动时钟脉冲，4 与门的输出和模拟公里数的 74LS160 芯片的 CLK 相连接，如**图 2.4** 所示。

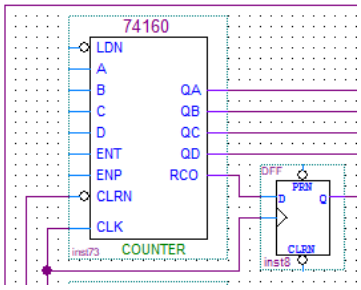



**图 2.4 行驶公里数电路**

值得注意的是，自动时钟脉冲 CLK 只连到代表公里数小数位的 74160 芯片，每有一个时钟脉冲，74160 自动加 1，如**图 2.5** 所示。当计数到 9，此时 RCO 为 1，代表要产生进位。

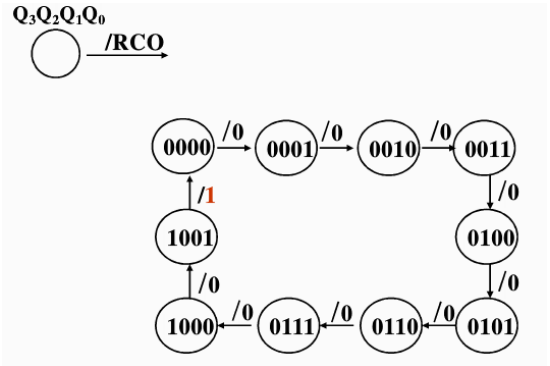



**图 2.5 74160 芯片状态转移图**

进位 RCO 作为 D 触发器的输入。这样在当小数位计数从 9 变为 0 后，D 触发器的输出为 1，而在小数位计数为 9 时，D 触发器的输出为 0，通过增加 D 触发器，解决了进位周期的问题。如**图 2.6** 所示，只需将输出连接到下一个 74LS160 计数器的 CLK 端，即小数位产生了进位，那么个位的数值就加 1。接下来的十位，百位以此类推。

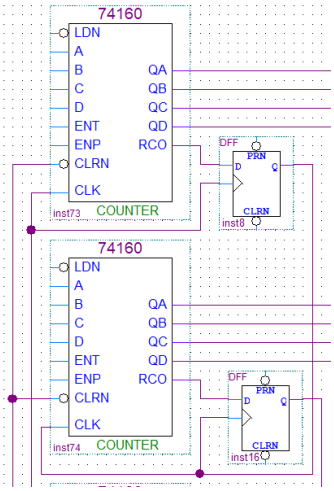



**图 2.6 行驶距离电路（部分）**

在起步里程以内，公里数随着时钟周期不断增加。而价格数不会发生改变。我们如何知道当前公里是否超过了起步里程呢？

在概要设计中我们提到使用 7485 芯片来进行当前里程和起步里程的比较。在起步里程数不会超过 9.9 公路，一共 8 位二进制数，所以我们只需要两片 7485 芯片，其功能表与 Quartus 中的符号分别如**表 2.2** 和**图 2.7** 所示。

**表 2.2 7485 四位数值比较器功能表**

**7485功能表**

| 比较输入 | | | | 级联输入 | | | 输出 | | |
|---|---|---|---|---|---|---|---|---|---|
| $A_3\ B_3$ | $A_2\ B_2$ | $A_1\ B_1$ | $A_0\ B_0$ | $I_{(A>B)}$ | $I_{(A<B)}$ | $I_{(A=B)}$ | $Y_{(A>B)}$ | $Y_{(A<B)}$ | $Y_{(A=B)}$ |
| $A_3>B_3$ | × | × | × | × | × | × | 1 | 0 | 0 |
| $A_3<B_3$ | × | × | × | × | × | × | 0 | 1 | 0 |
| $A_3=B_3$ | $A_2>B_2$ | × | × | × | × | × | 1 | 0 | 0 |
| $A_3=B_3$ | $A_2<B_2$ | × | × | × | × | × | 0 | 1 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1>B_1$ | × | × | × | × | 1 | 0 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1<B_1$ | × | × | × | × | 0 | 1 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0>B_0$ | × | × | × | 1 | 0 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0<B_0$ | × | × | × | 0 | 1 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0=B_0$ | 1 | 0 | 0 | 1 | 0 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0=B_0$ | 0 | 1 | 0 | 0 | 1 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0=B_0$ | × | × | 1 | 0 | 0 | 1 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0=B_0$ | 1 | 1 | 0 | 0 | 0 | 0 |
| $A_3=B_3$ | $A_2=B_2$ | $A_1=B_1$ | $A_0=B_0$ | 0 | 0 | 0 | 1 | 1 | 0 |

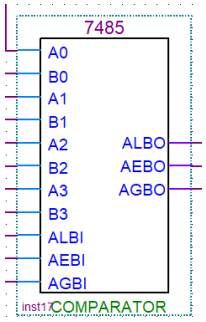



**图 2.7 Quartus 中 7485 芯片的符号**

第一个 7485 芯片的输入端 B3B2B1B0，连接存储起步里程的 74LS373 芯片的低四位，输入端 A3A2A1A0，连接当前公里数计数器的小数位 74LS160 输出 QD，QC，QB，QA。

第二个 7485 芯片的输入端 B3B2B1B0，连接存储起步里程的 74LS373 芯片的高四位，输入端 A3A2A1A0，连接当前公里数计数器的个位 74LS160 输出 QD，QC，QB，QA。

这里我们使用了比较器的串行扩展，低四位比较的结果，会输入到高四位的比较器的输入中，第一片 7485 的输出 ALBO 接到第 2 片 7485 的输入 ALBI 中，其余以此类推。

当高四位的比较结果为相等时，这时就考虑第一片 7485 的输出 ALBO，AEBO，AGBO；否则只关注高四位的比较结果，也就是第二片 7485 的输出 AGBO。

当当前公里数值大于设置的起步里程时，第二片 7485 的输出端 AGBO 输出为 1。

### 2.3 运行模式（起步公里以外）

AGBO 输出为 1，超出起步里程，价格数开始发生变化。对于价格数，我们使用的是自设封装的芯片 BCD accumulator，从十分位到百位一共有 5 位，所以我们使用了 5 片 BCD accumulator。如果只有 3 位的话，按照每 0.1 公里的价格发生变化，很容易想到这时会产生精度丢失的问题。同时还要解决溢出灯的问题。因此一共需要 5 位有效数字。

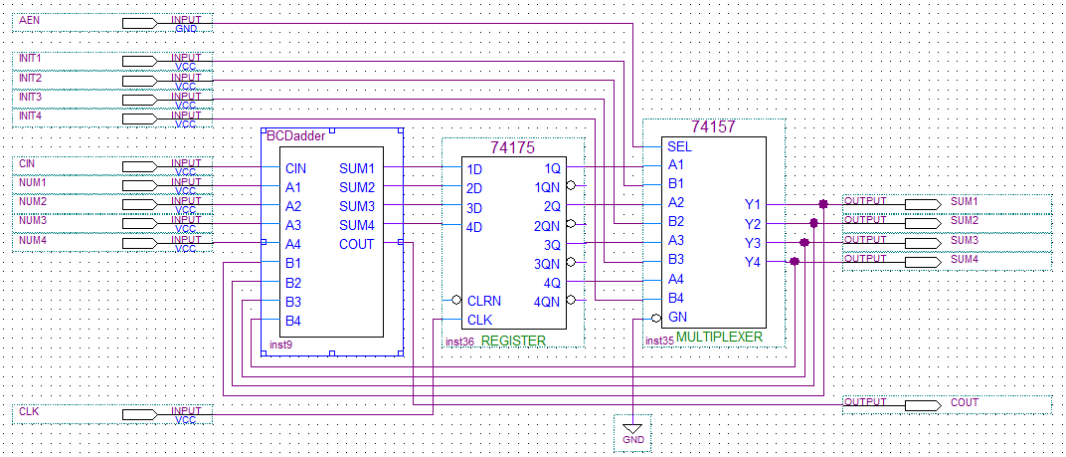



**图 2.8 8421BCD 累加器 BCDaccumulator 电路**

**图 2.8** 是我们设计的 BCD accumulator 芯片，其由一个自设封装的 BCD adder 芯片和一片 4D 触发器 74LS175 芯片和一个四组 2 选 1 数据选择器 74LS157 组成，74LS157 的功能表如**表 2.3** 所示。

**表 2.3 74157 二选一选择器功能表**

| 输入 | | | | 输出 |
|---|---|---|---|---|
| G | S | A | B | Y |
| H | X | X | X | L |
| L | L | L | X | L |
| L | L | H | X | H |
| L | H | X | L | L |
| L | H | X | H | H |

前面提到了当前公里数大于起步距离时，AGBO 输出为 1，AGBO 连接到每一个 BCD accumulator 芯片的 AEN 端，这时 74157 数据选择器的 SEL 端为 1，输出值就是 A4A3A2A1,

当 AEN 为 0 时（未超出起步里程），输出值为 B4B3B2B1，B4B3B2B1 和 INIT4，INIT3，INIT2，INIT1 相连接。而对于 INIT，代表百分位的 BCD accumulator 的初值为 0，代表十分位、个位、十位的 BCD accumulator 的 INIT 直接和 74LS373 锁存器的输出相连接。这样也就实现了在起步里程以内使数码管显示起步价。

接着讨论当 AGBO 为 1 的情况，此时数据选择器输出为 BCD adder 加法器加后的结果，同时，累加结果再次作为加法器的输入。BCD adder 的结构如**图 2.9** 所示。

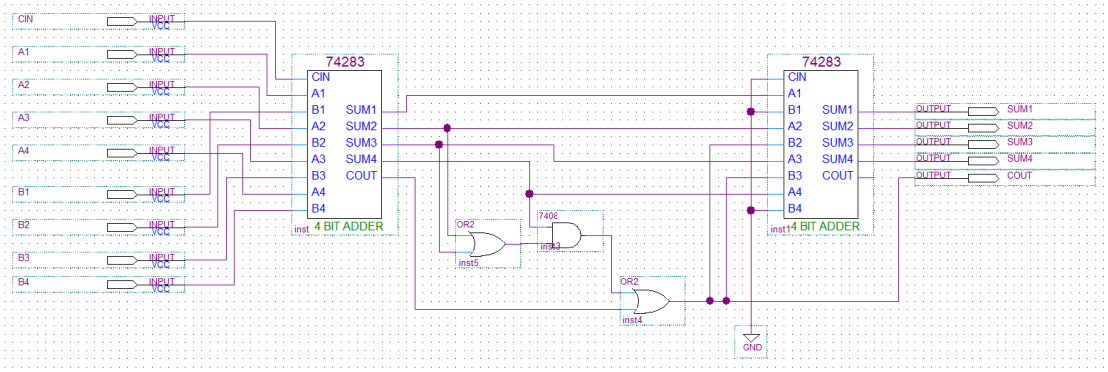



**图 2.9 8421BCD 加法器 BCDadder 电路**

74LS283 是 4 位 2 进制超前进位全加器。它可以实现每一位的进位直接由加数和被加数产生，而不需要等待低位的进位信号，运算速度很快。

在运行过程中，对于每一个时钟脉冲，我们需要将设置模式中存储的每 0.1 公里单价和当前的价格数进行相加。因此对于这一组 BCD accumulator 芯片，其代表百分位的芯片的 NUM4，NUM3，NUM2，NUM1 要和单价的低 4 位相连接，代表十分位的芯片的 NUM4，NUM3，NUM2，NUM1 要和单价的高 4 位相连接，而 NUM 端和 BCD adder 的 A 端连接，作为加数，上次加法后的结果作为被加数输入到 BCD adder 的 B 端。所以代表个位、十位、百位的 BCD accumulator 芯片的 NUM 端都接地，它们不需要加数，只需要进位数。BCD accumulator 的输出端 cout 代表进位数，连接到下一位 BCD accumulator 芯片的 cin 端。

接着讨论 BCD adder 加法器的设计细节。由于我们希望 4 位进行运算后的结果从二进制数转换成 10 进制 BCD 码。所以在相加以后我们还需要一个 74LS283 来对前面计算的结果进行修正。显然最低位和最高位不需要处理，所以 cin 和 B1 接地，同时 B4 也需要接地。

对于 B2 和 B3，我们知道，如果要将 4 位二进制数转化为 BCD 码，大于 9 的数只需要加上 6 即可完成修正。所以要判断这个加完以后的数是否大于 9，大于 9 就需要对 B2 和 B3 进行修改。通过两片 74LS283 芯片间的电路，我们完成了对相加后数值的修正。

经过 BCD adder 处理后的 BCD 码经过 4D 触发器输送到数据选择器。进位送到下一位 BCD accumulator 芯片中。

以上就是价格计数的原理。

## 2.4 溢出状态灯

由于实验箱上面只有 6 个数码管，公里数的显示需要 3 位，价格数的显示需要 3 位。3 位数码管的显示范围只有 0-99.9，通常情况下，价格数的变化速度远比公里数的变化速度要

快，当价格数溢出时，公里数的数值可能会比较小。为了增加数据的显示范围，增加百位的运算，使得百位数值每增加 1 就亮起一个状态灯。由此，可以设计一个转换器，将百位结果转换为灯的显示形式，称之为 LightConverter。

根据其功能，可以写出真值表（**表 2.4**）

**表 2.4 LightConverter 真值表**

| C | B | A | Y4Y3Y2Y1 |
|---|---|---|---|
| **0** | 0 | 0 | 0000 |
| **0** | 0 | 1 | 0001 |
| **0** | 1 | 0 | 0011 |
| **0** | 1 | 1 | 0111 |
| **1** | 0 | 0 | 1111 |

通过该真值表，我们可以通过画卡诺图的方式得出各个输出关于输入变量 C、B、A 的表达式，并且将表达式转化成逻辑电路，如**图 2.10** 所示。

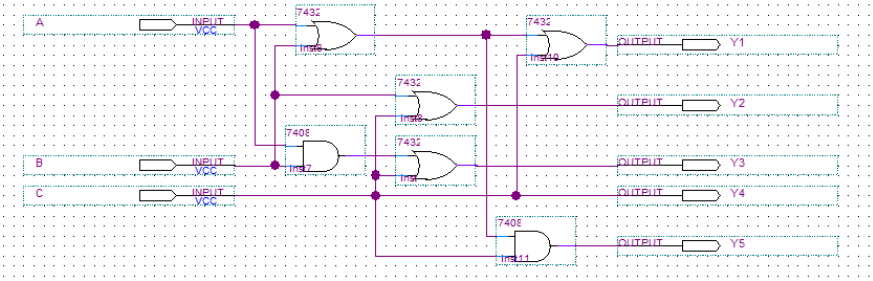



**图 2.10 LightConverter 电路**

由此，公里数和价格百位的四位状态灯显示便可通过 LightConverter 实现，如图 **2.11(a)** 和 **2.11(b)**所示。

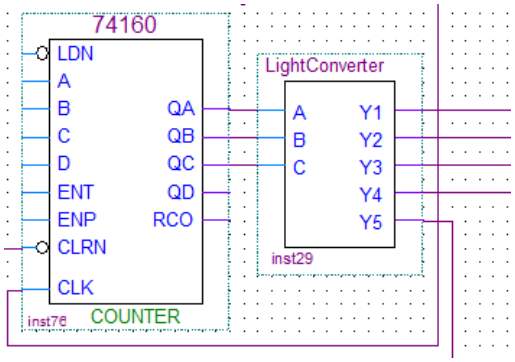



**图 2.11(a) 行驶距离百位显示电路**

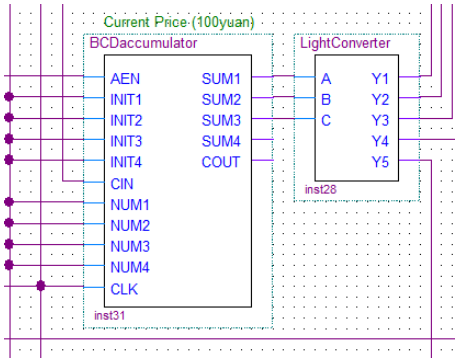



**图 2.11(b) 当前价格百位显示电路**

## 2.5 清除功能

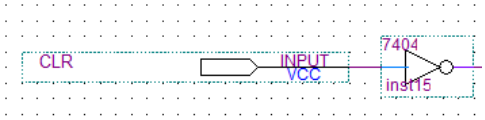



**图 2.12 清除输入 CLR**

如**图 2.12** 所示，增加一个 input 端，实现对系统的重置。重置后，公里数重置为 0，价格数重置为起步价格。根据前面的介绍，我们的 CLR 只需要连接到公里数累加器的 CLR 端，当公里数重置为 0 以后，和起步公里的比较结果自然是小，这样 AGBO 的输出结果为 0，在 BCD accumulator 中，数值选择器会选择我们所打入的初始值，并且更新 output。我们的初始值连接 74LS373 锁存器。通过这种方式，完成对价格数的重置。

### 2.6 暂停功能

如**图 2.13** 所示，增加一个 PAUSE 端，和系统的时钟脉冲信号作与运算，当 PAUSE 无效时，相当于正常的时钟脉冲信号；PAUSE 有效时，脉冲为低电平，停止各个芯片的计数。

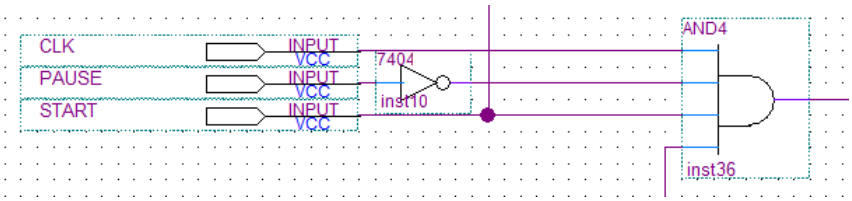



**图 2.13 暂停输入 PAUSE**

### 2.7 溢出锁死

当公里数和价格数的某一个发生溢出后，灯会逐渐亮起，由于灯的数量为最多 4，例如价格数的 4 个灯全部亮起，这时数码管再次溢出，我们希望此时系统自动暂停。所以在 Lightconverter 中，当 CBA 输入为 101 时，Y5 输出为 1。

价格数和公里数各自 Lightconverter 芯片的两个 Y5 进行或非运算，送入**图 2.13** 中的四与门。在溢出灯显示范围内，两个 Y5 均为 0，或非运算结果为 1，与 CLK、PAUSE、START 进行与运算构成时钟信号；当价格数或者公里数灯溢出后，或非运算结果为 0，与运算的结果为 0，时钟信号变为低电平，和暂停时类似，系统停止计数。

## 3 仿真测试

下面是我们使用 Quartus II 的仿真模拟功能对设计好的电路进行的测试。

### 3.1 信号名称

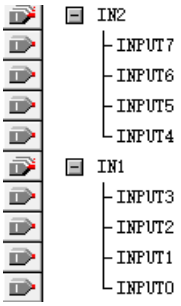



**图 3.1 数据输入**

由上至下分别为 INPUT 7 到 INPUT0，是起步相关数据的输入。

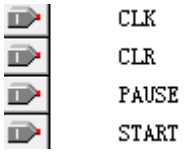



**图 3.2 控制信号**

CLK 控制时钟信号，CLR 能实现公里数和价格的重置，PAUSE 能暂停公里数的计数，START 控制设置模式和运行模式的切换。

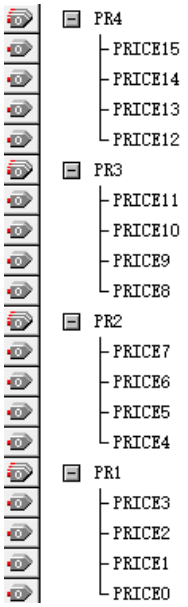



**图 3.3 价格数输出**

由上至下分别为 PRICE15 到 PRICE0，PRICE11 到 PRICE0 是价格的输出，PRICE15 到 PRICE12 是价格溢出的次数。

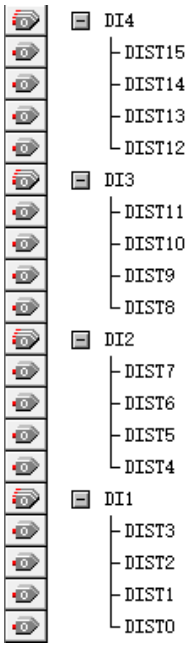



**图 3.4 公里数输出**

由上至下分别为 DIST15 到 DIST0，DIST11 到 DIST0 是价格的输出，DIST15 到 DIST12 是价格溢出的次数。

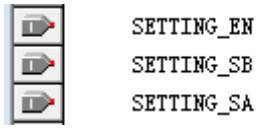



**图 3.5 选择信号**

从上至下分别为 SETTING_EN、SETTING_SB、SETTING_SA。

## 3.2 设置起步价格、起步里程和里程单价

将 START 设置为低电平且将 SETTING_EN 设置为高电平进入设置模式。

下图是数据的输入设置：

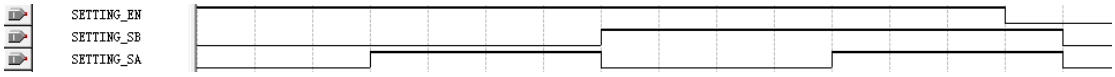



**图 3.6 设置选择信号**


**图 3.7 设置变量初始值**

当 SETTING_SB、SETTING_SA 为 00 时，设置起步价为 14 元；

当 SETTING_SB、SETTING_SA 为 01 时，设置起步价小数位为 0.1；

当 SETTING_SB、SETTING_SA 为 10 时，设置每 0.1 公里单价为 0.86 元；

当 SETTING_SB、SETTING_SA 为 11 时，设置起步公里为 4 公里。

## 3.3 运行计数器

将 START 置为 1，SETTING_EN 置为 0 进入运行模式，在未超过起步公里 4 公里时价格不变，恒为 14.1 元，公里数增长，如**图 3.8**：

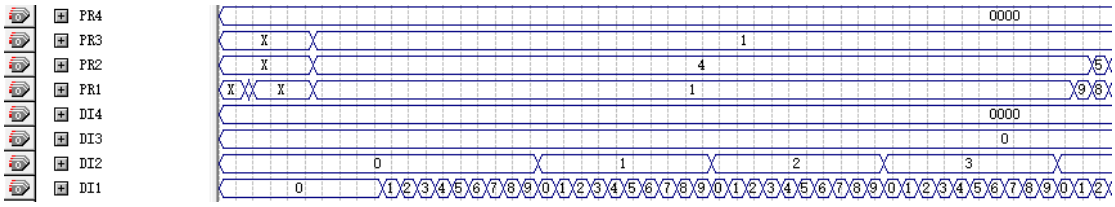



**图 3.8 没有超出起步公里时**

当公里数超过 4 公里后，价格开始按照 0.86 元/0.1 公里增长，如**图 3.9**：

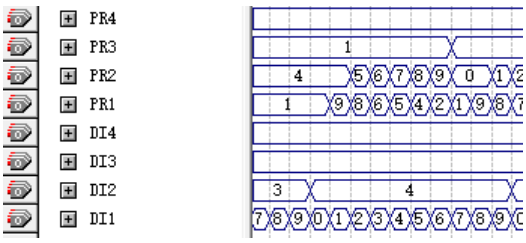



**图 3.9 超出起步公里以后**

当价格超过 99.9 元，即三位数码管显示极限时，出现溢出。PR4（PRICE15-PRICE12）记录着价格溢出的次数，如**图 3.10**。

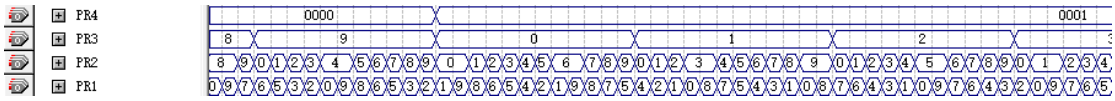



**图 3.10 价格数第一次溢出，亮起一个状态灯**

当 PRICE15-PRICE12 为 1111 即再次溢出时，系统停止计数，如**图 3.11**。

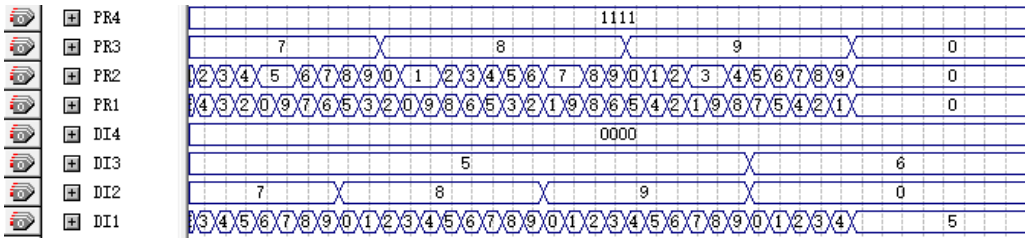



**图 3.11 价格数溢出系统显示范围，进入停机锁死状态**

### 3.4 暂停功能测试

在运行过程中将一段时钟周期的 PAUSE 置为 1，如**图 3.12**：

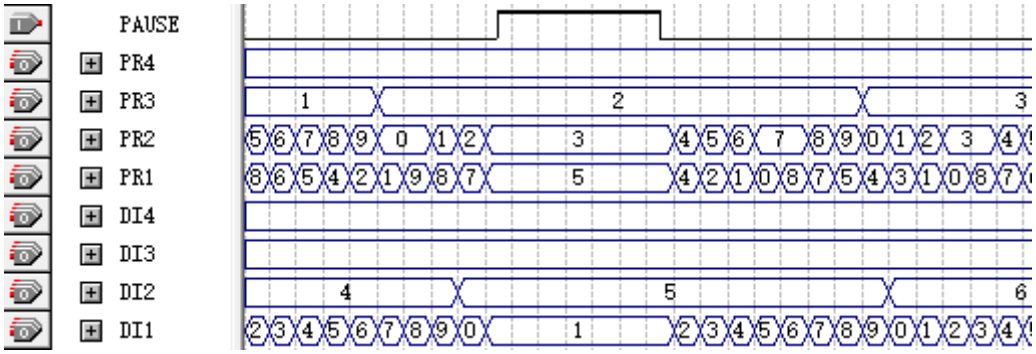



**图 3.12 暂停**

可发现在这段时间内里程和价格都未发生变化，可知暂停功能成功实现。

### 3.5 重置功能测试

将 CLR 调高电平，START 调低电平后，公里数重置为 0，价格也重置为起步价 14.1 元，如**图 3.13** 所示，可知 CLR 重置功能成功实现。

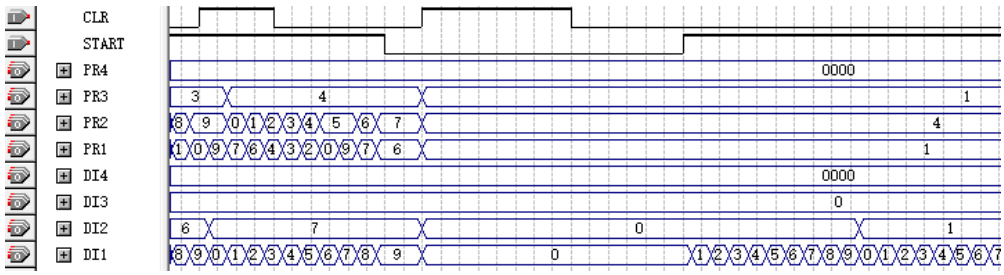



**图 3.13 清除**

## 4 实际调试

下面是在实际机箱上进行的调试。

### 4.1 设置引脚

首先我们选择芯片型号并设置了引脚：


**图 4.1 定义 pins**

## 4.2 接线

根据引脚设置在机箱上的接线如下所示：

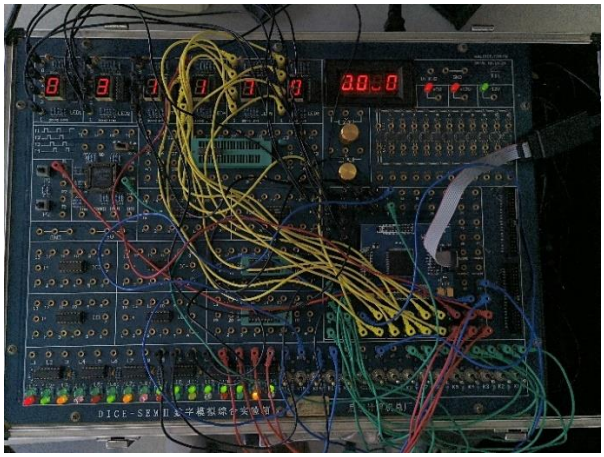


**图 4.2 实验箱接线**

## 4.3 测试与验收

接着我们对各个功能进行了测试，各项功能显示完全，在设置模式时，能够进行起步价格、起步里程和里程单价的设定。

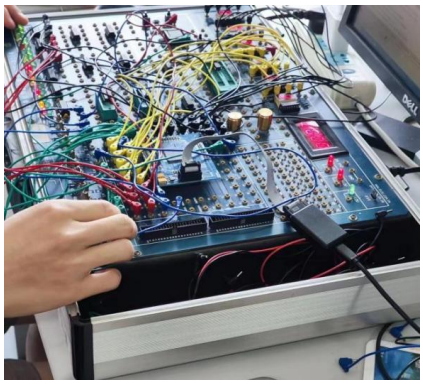


**图 4.3 验收**

在运行模式下，达到起步里程后，价格随着里程数增加，当价格到达 99.9 元时，溢出灯点亮，当 4 个灯全部亮起后，系统停止计数。

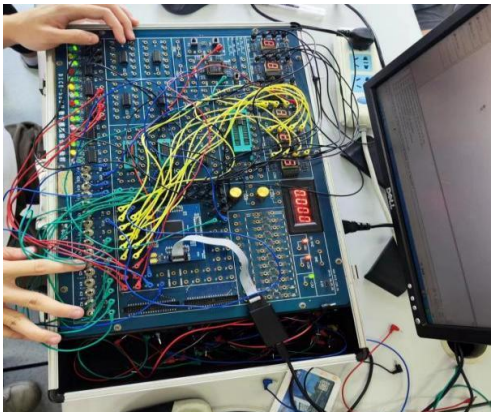
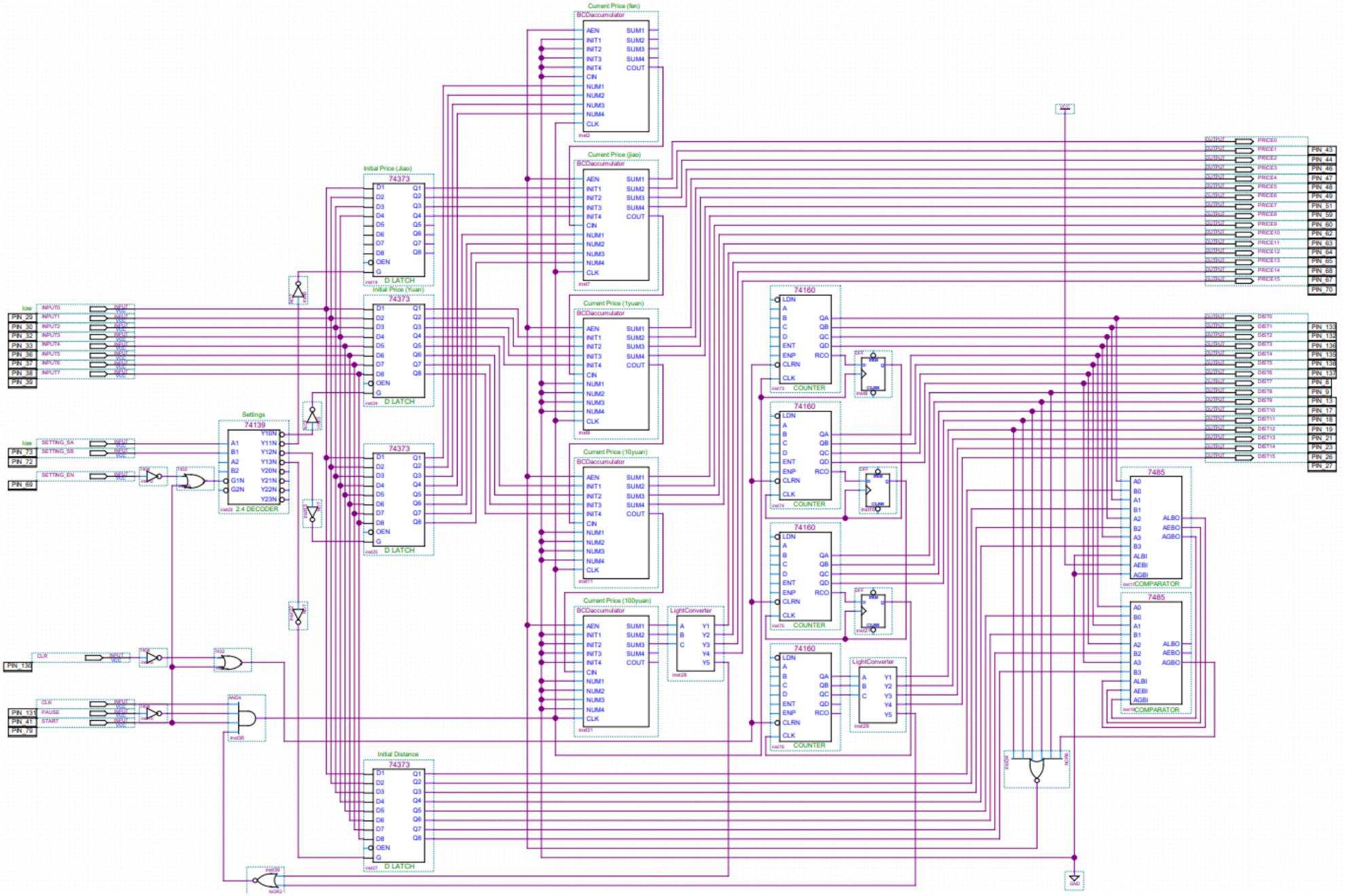


**图 4.4 验收**

调试没有问题，最后我们进行了项目的验收，在张老师的检查下成功通过。


Current Price (fen)
BCDaccumulator
Current Price (jiao)
Current Price (1yuan)
Current Price (10yuan)
Current Price (100yuan)
Initial Price (Jiao)
Initial Price (Yuan)
Initial Distance
74373
D LATCH
Settings
74139
2:4 DECODER
74160
COUNTER
LightConverter
7485
COMPARATOR
low
PIN_29
PIN_30
PIN_32
PIN_33
PIN_36
PIN_37
PIN_38
PIN_39
PIN_73
PIN_72
PIN_69
PIN_130
PIN_131
PIN_41
PIN_79
PIN_43
PIN_44
PIN_46
PIN_47
PIN_48
PIN_49
PIN_51
PIN_59
PIN_60
PIN_62
PIN_63
PIN_64
PIN_65
PIN_68
PIN_67
PIN_70
PIN_133
PIN_132
PIN_136
PIN_135
PIN_138
PIN_137
PIN_8
PIN_9
PIN_13
PIN_17
PIN_18
PIN_19
PIN_21
PIN_23
PIN_26
PIN_27