**一、设计要求及指标**

1.1设计目标

本项目主要设计一个出租车自动计价系统，计价器可以通过数码管显示当前行车的总里程、总时间以及需要支付的车费，当计价器跳表时，会有声音提示提醒乘客。此外，此计价器还可对起步价、里程单价及计时单价等进行调整。此计价器在23点到次日6点之间行车收取额外的费用，对远程行车（设定为12千米）收取额外的费用。

1.2需求分析

考虑到我们设计一个出租车计价器，那么有下面的需求：

* 基础功能：
  + 复位、启动：司机拨下空车提示牌，启动并复位计价器数据，开始新一轮计价。
  + 里程计价：当车速高于3.6km/h时按照里程计价，白天每公里3元，夜晚每公里4元。
  + 计时计价：当车速低于3.6km/h时按照计时计价，白天每分钟3元，夜晚每公里4元。
  + 起步价：当行驶距离在2km之内或行驶等待时长在10分钟之内，则保持起步价，白天为10元，夜晚加收2元。
  + 等待超时收费：在行驶途中等待时间超过十分钟后，每分钟等待时间白天3元/分钟，夜晚4元/分钟。
  + 停止计价： 司机将空车提示牌拨上，结束行程，计价器停止计价。
  + 显示当前收费：数码管显示当前应收车费。
  + 起步价、里程单价、计时单价可调：可以根据需求调整起步价，里程单价，计时单价。
* 扩展功能：
  + 用户体验
    - 显示总里程：显示当前已行驶总里程。
    - 显示行驶总计时时间：显示上车后已行驶总时间。
    - 跳表（里程、计时）的提示：跳表时发出“滴”的提示。
  + 计价
    - 长途附加费：行车路程超出12km，则里程计价和计时计价均增加1元。
    - 夜间附加费：若在夜间23时至次日6时行车，则里程计价和计时计价均增加1元，若同时又为长途，则里程计价和计时计价均增加2元。

1.3设计技术指标

* 系统输入
  + 启动信号key\_start：使能计价器复位并开始计价。
  + 确定按键key\_conform:设置出租车价格的时候和设置当前时间的时候确认。
  + 设置价格按键key\_price:使能价格设置模块，对出租车价格进行设置。
  + 位选按键key\_select：设置当前时间和出租车价格的时候选择设置的位。
  + 设置当前时间按键key\_time:使能当前时间设置模块，对出租车当前时间进行设置。
  + 设置赋值按键key\_set:在设置当前时间和出租车价格的时候按下，在位选按键key\_select的配合下，该位数值加1。
  + 车轮脉冲wheel：表示出租车的行驶车轮脉冲。
  + 时间脉冲clk：用于驱动出租车计价器，设计频率为10Hz。仿真的时候为了节约时间，频率设为10KHz。
* 系统输出
  + 行驶总里程数码管段选编码，位选：outpdistance,seldistance。
  + 行驶总时间数码管段选编码，位选：outptime,seltime。
  + 行驶总价数码管段选编码，位选:outprice,selprice。
  + 蜂鸣器跳表提示：bee\_pulse
* 系统参量
  + 高低速车轮脉冲阈值：interval\_threshold
  + 里程跳表距离：run\_unit=1km
  + 里程单价： day\_mileage\_price =3元
  + 计时单价： day\_timing\_price=3元/分钟
  + 起步价：day\_starting\_price=10元
  + 车轮周长：wheel\_circumference=（0.63×π）m
* 基础功能
  + 复位、启动
    - 检测启动信号按键key\_start，启动计价器，复位所有计价数据，开始新一轮计价。
  + 高低速度（里程、计时）判定
    - 计算相邻wheel脉冲之间的间隔wheel\_interval。
    - 当相邻wheel脉冲之间的间隔Wheel\_interval达到interval\_threhold，则判定出租车处于低速行驶，否则判定为高速行驶。
  + 里程计价脉冲生成
    - 当出租车处于高速状态下使能，按照里程计价标准计价。
    - 累加wheel脉冲到累加值wheel\_count。
    - 当累加值wheel\_count 大于(里程跳表距离run\_unit /车轮周长wheel\_circumference) 时，即产生里程计价脉冲run\_pulse，并且复位wheel\_count。
  + 里程计价
    - 每一个里程计价脉冲run\_pulse会产生一个里程单价day\_mileage\_price，总价total\_price 为里程单价run\_price的累加。
  + 计时计价脉冲生成
    - 低速状态下使能。
    - 当计时时间wait\_time 大于计时跳表时间wait\_unit ，则产生计时计价脉冲wait\_pulse，并复位wait\_time。
  + 计时计价
    - 每一个计时计价脉冲wait\_pulse产生一个计时单价wait\_priceshiy，总价total\_price 为计时单价wait\_price的累加。
  + 起步价
    - 当总价total\_price小于起步价start\_price, 支付费用pay就是start\_price；否则，支付费用pay为total\_price。
    - 启动计价后，先以起步价内单价计价；当总价total\_price超过起步价start\_price后，以起步价外单价计价。
  + 停止计价
    - 检测到信号start第二次脉冲，则暂停所有计价。
  + 显示当前收费
    - 通过5位7段LED显示支付费用pay。
* 扩展功能
  + 用户体验
    - 显示总里程：通过7段数码管显示当前里程run\_total。
    - 显示总计时时间：通过7段数码管显示运行时间time\_total，精确到分钟。
    - 跳表（里程、计时）的提示：计价器跳表时，会发出“滴”的一声，提醒乘客。当检测到一个里程计价脉冲run\_pulse或者计时计价脉冲wait\_pulse时，就会触发蜂鸣器发出提醒。
  + 计价
    - 长途附加费：当run\_total大于12km时，计时计价和里程计价均在原有基础上加1。
    - 夜间附加费： 当time在23-6时之间，里程计价和计时计价均增加1,若同时满足run\_total大于12km，则里程计价和计时计价均增加2。

二、系统总体设计

**2.1 状态机分析**

**2.1.1 状态迁移图**

****

图2.01 系统的状态迁移图

如图2.01 整个系统的状态迁移图，系统分为四个状态，状态描述如下表2.01所示。

表2.01 系统状态描述

|  |  |
| --- | --- |
| 状态 | 状态描述 |
| 停止计价状态 | 出租车处于空车状态，没有接客计价，空车牌处于立起状态，此时，启动信号为低电平，系统处于停止计价状态，同时，可以进行初始时间以及初始价格的设置 |
| 启动复位状态 | 出租车接到乘客，将空车牌按下，启动信号为高电平，系统处于启动状态，发出复位信号，使计价器价格复位，同时发出启动状态信号，使能计价模块 |
| 起步价内计价状态 | 接收到启动状态发出的启动状态信号，系统进入起步价内计价状态，当里程小于2km且低速计时时间小于10min时，一直处于起步价内计价 |
| 起步价外计价状态 | 当里程数大于2km或在2km内低速计时时间超过10min时，系统处于起步价外计价状态，价格变动，蜂鸣器发出警报 |

**2.1.2主要状态迁移描述**

* **状态表**

下表2.02为系统状态表，表明系统状态迁移情况。通过启动信号st、2km里程判断信号qibu和2km内低速计时10min判断信号tenin控制状态，判断当前处于哪一种计价模式，并输出复位信号load和计价模式状态值state\_pricing，使能对应的计价模块或复位。

表2.02 系统状态表(Sn+1/Zn)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 现态Sn  输入/ Xn | 停止计价  Stop | | 启动复位  Start\_reset | 起步价内计价  Within\_starting\_price | 起步价外计价  Beyond\_starting\_price |
| 000 | | 停止计价/04 |  | 停止计价/02 |  |
| 001 | |  |  |  | 停止计价/03 |
| 010 | |  |  |  | 停止计价/03 |
| 011 | |  |  |  | 停止计价/03 |
| 100 | | 启动复位/04 | 起步价内计价/11 | 起步价内计价/02 |  |
| 101 | |  |  | 起步价外计价  /02 | 起步价外计价/03 |
| 110 | |  |  | 起步价外计价  /02 | 起步价外计价  /03 |
| 111 | |  |  | 起步价外计价  /02 | 起步价外计价  /03 |

**注：**

1. 横线处表明此状态迁移存在逻辑错误，不可达。
2. 输入Xn 的3位变量顺序分别为启动信号st 、2km里程判断信号qibu和2km内低速计时10min判断信号tenin。

输出Zn的2位变量顺序分别为复位信号load和计价状态信号state\_pricing，其中，state\_pricing为整数类型。

* **迁移描述**

1. 当系统的启动信号start = 0，系统进入停止计价Stop状态，输出停滞状态state\_start=0，显示最终总价，当系统启动，启动信号start=1，系统进入启动复位Start\_reset状态；若start一直为0，则保持stop状态。
2. 当处于启动复位Start\_reset状态时，输出复位信号load=1，计价状态state\_pricing=1 ，启动状态信号state\_start=1然后下一个时钟上升沿，进入起步价内计价状态Within\_starting\_price。
3. 当处于起步价内计价Within\_starting\_price状态时，输出复位信号load=0，计价状态state\_pricing=2，使能起步价内计价模块。当里程小于2km，且低速计时时间小于10min，即qibu=0，tenin=0时，仍为起步价内计价状态；当接收到停止信号start=0，进入停止计价stop状态；否则，进入起步价外计价Beyond\_starting\_price状态。
4. 当处于起步价外计价Beyond\_starting\_price状态时，输出复位信号load=0，计价状态state\_pricing=3，使能起步价外计价模块。当接收到停止信号start=0，进入停止计价stop状态；否则，仍为起步价外计价Beyond\_starting\_price状态。

**2.2 系统总体描述**

**2.2.1系统图(原理图)**

****

图2.02 系统原理图

**2.2.2基本工作原理**

系统工作于总线时钟频率下，通过按键key\_time、key\_price进入初始时间和初始价格设置模块，按键key\_select、key\_set分别进行选位和加值操作，设置完成后，通过key\_confirm按键确认设置。按键key\_start即空车牌，也就是计价器的启动按键，当牌子按下，进入计价模式，通过测定车轮的脉冲输入wheel，判断计价状态，由中心控制模块进行状态控制，实现实时计价，并显示在数码管上。

**2.2.3总输入、输出的描述及具体参数值**

* **输入信号**
  + clk：系统时钟信号，数据类型为std\_logic，时钟频率为10Hz，在实际仿真时，由于仿真时间过长，仿真采用时钟为100ns；
  + key\_start：计价启动按键，数据类型为std\_logic，按下按键时，输出为1，复位上一次计价，开始新一轮计价；关上按键时，输出为0，结束计价；
  + key\_time：初始时间设置按键，数据类型为std\_logic，按下按键进入初试时间设置模式，一般只需要同步设置一次，用来判断当前是否为夜间行驶；
  + key\_price：初始价格设置按键，数据类型为std\_logic，按下进入价格设置模式，按1次，可设置白天起步价，再按1次，设置夜晚起步价，再按1次，设置白天里程单价，再按1次，设置白天计时单价，再按1次，设置白天长途单价，按confirm按键退出设置；
  + key\_select：设置位选按键，数据类型为std\_logic，在时间、价格设置时，通过按下按键，选择要设置的位；
  + key\_set：数值设置按键，数据类型为std\_logic，在时间、价格设置时，通过按下按键，改变当前位的数值；
  + key\_confirm：设置确认按键，数据类型为std\_logic，确认设置，退出时间、价格设置模式；
  + wheel：车轮脉冲信号，数据类型为std\_logic，判断汽车行进状态。
* **输出信号**
  + selprice[2..0]：价格显示数码管位选输出，数据类型为std\_logic\_vector(2 downto 0)；
  + outpprice[7..0]：价格显示数码管段选输出，数据类型为std\_logic\_vector(7 downto 0)，输出要求接8位共阴数码管，当段码子输入为1时，点亮对应的段LED，显示对应价格位；
  + seltime[2..0]：时间显示数码管位选输出，数据类型为std\_logic\_vector(2 downto 0)；
  + outptime [7..0]：时间显示数码管段选输出，数据类型为std\_logic\_vector(7 downto 0)，输出要求接8位共阴数码管，当段码子输入为1时，点亮对应的段LED，显示对应时间位；
  + seldistance[2..0]：里程显示数码管位选输出，数据类型为std\_logic\_vector(2 downto 0)；
  + outpdistance [7..0]：里程显示数码管段选输出，数据类型为std\_logic\_vector(7 downto 0)，输出要求接8位共阴数码管，当段码子输入为1时，点亮对应的段LED，显示对应里程位；
  + bee\_pulse:跳表信号bee\_pulse,当价格发生变化之后，产生输出。

**2.3 各模块的描述**

**2.3.1按键消抖模块**

****

图2.03 remove\_jitter模块

* **基本功能**

实现按键key\_time、key\_price、key\_select、key\_set、key\_confirm的防抖功能，消除毛刺信号，输出稳定的电平信号。

* **输入信号及具体参数值**

clk：系统时钟信号，数据类型为std\_logic，时钟频率为10Hz；实际仿真采用100ns；

key\_in： 按键输入信号，变量类型为std\_logic。由于按键key\_time、key\_price、key\_select、key\_set、key\_confirm每次只有其中一个被按下，因此，只设一个输入。

* **输出信号及具体参数值**

key\_out: 消抖后的按键输出信号，变量类型为std\_logic。进行消抖后，按下按键，输出高电平1，若按键未被按下，输出低电平0。

**2.3.2 高低速判断模块**



图2.04 hl模块

* **基本功能**

输入时间脉冲clk和车轮脉冲wheel后，通过二者进行高低速判断，一个wheel脉冲表示车辆行经2m，一个clk脉冲表示0.1s，在超过20个clk脉冲中没有出现wheel，即速度小于1m/s，定为低速运行，进行计时计价；在20个clk脉冲中出现了wheel脉冲，即速度大于1m/s，定为高速运行，进行里程计价。

* **输入信号及具体参数值**

ce：使能信号，数据类型为std\_logic，输入1有效，通过按键key\_start输入；

clk：系统时钟信号，数据类型为std\_logic。频率为10Hz；

load：复位信号，数据类型为std\_logic，输入1有效，通过状态机输入；

din：车轮脉冲信号，数据类型为std\_logic，一个车轮脉冲表示出租车行进2m。

**输出信号及具体参数值**

ceout：数据类型为std\_logic，高低速状态输出标志，输出1表示高速状态，输出0表示低速状态。

**2.3.3 夜间行车判断模块**

通过设置与真实时间同步的初始时间，而后实现同步计数，达到时钟的作用，来判断当前行程是否为夜间行车。

* **初始时间设置模块**



图2.05 setInit模块

* **基本功能**

通过按键key\_time、key\_select、key\_set、key\_confirm设置初始时间，与真实时间同步。按下按键key\_time进入时间设置模式，通过按键key\_select选择要设置的位，按键key\_set进行加值操作，设置完成后，按下按键key\_confirm退出设置模式。

* **输入信号及具体参数值**

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

time\_state : 按键key\_time消抖后的信号，进入/退出时间设置模式，数据类型为std\_logic；

sel , set : 按键key\_select和key\_set消抖后的信号，选位信号和加值信号，数据类型为std\_logic；

confirm : 按键key\_confirm消抖后的信号，确认信号，按下确认当前时间，位选标志位复位，退出设置模式，数据类型为 std\_logic。

* **输出信号及具体参数值**

hour\_s, hour\_f , min\_s , min\_f : 输出时间的4位数据，数据类型为integer。

* **初始时间转化模块**



图2.06 settime模块



图2.07 10位乘法器模块

* **基本功能**

将setInit中输出的4位时间信号，转化为总分钟的形式，进而输入到lingcheng模块。

* **输入信号及具体参数值**

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

hour\_s, hour\_f , min\_s , min\_f : setInit模块输出的时间的4位数据，数据类型为integer。

* **输出信号及具体参数值**

settime：当前时间以00:00为基准，转化为分钟的形式输出，数据类型为std\_logic\_vector(10 downto 0)。

* **凌晨行车判断模块**

****

图2.08 lingcheng模块

* **基本功能**

通过计时器实现一个同步时钟，通过与360分钟（6h）和1380分钟（23h）比较，判定当前是否为夜间行车，若当前分钟数在小于360和大于1380的区间内，判定为夜间行车，启用夜间计价模式，否则，为白天行车，采取普通计价模式。

* **输入信号及具体参数值**

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

settime：当前时间的分钟的形式输入，数据类型为std\_logic\_vector(10 downto 0)。

timesel：设置夜间行车模式时间使能按键，由按键key\_time输；

* **输出信号及具体参数值**

ceout : 夜间行车判断信号，数据类型 std\_logic。当为夜间23h至次日6h间行车，为夜间行车，输出1，若为白天行车，输出0。

**2.3.4 价格调控模块**

* **五种价格设置模块**

****

图xx or5price模块引脚图

* **基本功能**

通过按键key\_price、key\_select、key\_set、key\_confirm设置五种初始价格，分别为白天起步价、夜晚起步价、白天里程单价、白天计时单价、白天长途单价。按键key\_price进入设置模式并通过按键次数选择需要设置的价格，按键key\_select选择要设置的位，按键key\_set进行加值操作，设置完成后，按键key\_confirm退出设置模式。

* **输入信号及具体参数值**

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

price\_state：按键key\_price消抖后的信号，进入/选择价格设置，数据类型为std\_logic，当按一次时，设置白天起步价；再按一次，设置夜晚起步价；再按一次，设置白天里程价；再按一次，设置白天计时价；再按一次，设置白天长途单价，以此循环，直至接收到confirm信号；

sel , set : 按键key\_select和key\_set消抖后的信号，选位信号和加值信号，数据类型为std\_logic；

confirm : 按键key\_confirm消抖后的信号，确认信号，按下确认当前价格，位选标志位复位，退出设置模式，数据类型为 std\_logic。

* **输出信号及具体参数值**

day\_starting\_price\_j：设置的白天起步价角位价格，数据类型为integer range 0 to 9 ；

day\_starting\_price\_y ：设置的白天起步价元位价格，数据类型为integer range 0 to 9 ；

day\_starting\_price\_t ：设置的白天起步价十位价格，数据类型为integer range 0 to 9 ；

day\_starting\_price\_h ：设置的白天起步价百位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_j：设置的夜间起步价角位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_y：设置的夜间起步价元位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_t：设置的夜间起步价十位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_j：设置的夜间起步价百位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_j：设置的白天里程单价角位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_y：设置的白天里程单价元位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_t：设置的白天里程单价十位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_h：设置的白天里程单价百位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_j：设置的白天计时单价角位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_y：设置的白天计时单价元位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_t：设置的白天计时单价十位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_h：设置的白天计时单价百位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_j：设置的白天长途单价角位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_y：设置的白天长途单价元位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_t：设置的白天长途单价十位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_h：设置的白天长途单价百位价格，数据类型为integer range 0 to 9 ；

pricesel : 输出当前设置的价格的标志位，至显示价格的数码管上，数据类型为integer，当设置某一种价格时，将其显示到价格数码管上。

* **价格转化模块**

****

图xx setprice模块引脚图

* **基本功能**

将set\_price模块设置好的每个价格的各位数值输入进setprice模块，调用10位乘法器，将每种价格转化为整数形式，为了便于计算，将价格设置为0-9999的整数，四位整数分别表示价格的百位、十位、元位和角位，在显示时再将各位分开。由此，输出五种价格的整数数值。

* **输入信号及具体参数值**

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

day\_starting\_price\_j：设置的白天起步价角位价格，数据类型为integer range 0 to 9 ；

day\_starting\_price\_y ：设置的白天起步价元位价格，数据类型为integer range 0 to 9 ；

day\_starting\_price\_t ：设置的白天起步价十位价格，数据类型为integer range 0 to 9 ；

day\_starting\_price\_h ：设置的白天起步价百位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_j：设置的夜间起步价角位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_y：设置的夜间起步价元位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_t：设置的夜间起步价十位价格，数据类型为integer range 0 to 9 ；

night\_starting\_price\_j：设置的夜间起步价百位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_j：设置的白天里程单价角位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_y：设置的白天里程单价元位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_t：设置的白天里程单价十位价格，数据类型为integer range 0 to 9 ；

day\_mileage\_price\_h：设置的白天里程单价百位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_j：设置的白天计时单价角位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_y：设置的白天计时单价元位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_t：设置的白天计时单价十位价格，数据类型为integer range 0 to 9 ；

day\_timing\_price\_h：设置的白天计时单价百位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_j：设置的白天长途单价角位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_y：设置的白天长途单价元位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_t：设置的白天长途单价十位价格，数据类型为integer range 0 to 9 ；

day\_long\_distance\_price\_h：设置的白天长途单价百位价格，数据类型为integer range 0 to 9 。

* **输出信号及具体参数值**

day\_starting\_price：白天起步价，数据类型为integer range 0 to 9999，以整数形式输出价格；

night\_starting\_price：夜晚起步价，数据类型为integer range 0 to 9999，以整数形式输出价格；

day\_mileage\_price：白天里程单价，数据类型为integer range 0 to 9999，以整数形式输出价格；

day\_timing\_price：白天计时单价，数据类型为integer range 0 to 9999，以整数形式输出价格；

day\_long\_distance\_price：白天长途单价，数据类型为integer range 0 to 9999，以整数形式输出价格。

**2.3.5 里程计价脉冲生成模块**

****

图xx licheng模块引脚图

* **基本功能**

当接收到起步计价信号时，里程计价脉冲生成模块licheng使能，每收到1个车轮脉冲din，即出租车行驶2m,根据车轮脉冲din的数目，没行进1km，输出里程计价公里脉冲dp；在出租车行驶达到或超过起步里程2km之后，起步电平qibu由0变为1；在出租车达到或行驶超过远程里程12km之后，远程电平yuancheng由0变为1；

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

load:复位信号，输入1有效，通过状态机切换状态输入;

din:里程脉冲信号，输入1表示出租车行驶过一个里程，用于统计出租车的行驶里程；

cein：高低速输入信号，cein输入1表示出租车处于高速状态，计价里程开始统计；cein输入0表示处于低速状态，计价里程不统计;

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

* **输出信号及具体参数值**

dp：计价里程脉冲，一个脉冲表示出租车高速状态下行进1km，用于里程计价；

qibu:起步状态信号，出租车处于起步状态内输出0，处于起步状态外输出1；

yuancheng:远程状态电平信号，出租车处于远程状态内输出0，处于远程状态外输出1；

**2.3.6 计时计价脉冲生成模块**

****

图xx delay模块引脚图

* **基本功能**

统计出租车路程中低速行驶的时间，用于计时计价，输出等待分钟脉冲和十分钟tenout电平输出，作为计时计价的标志；

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

load:复位信号，输入1有效，通过状态机切换状态输入;

cein：高低速输入信号，cein输入1表示出租车处于高速状态，计价里程开始统计；cein输入0表示处于低速状态，计价里程不统计;

clk :系统时钟信号，数据类型为std\_logic，频率为10Hz；

* **输出信号及具体参数值**

ceout：低速行驶分钟脉冲，一个ceout脉冲表示出租车低速行驶了1分钟；

tenout:低速行驶分钟电平：当tenout被置1的时候，表示出租车已经低速行驶10分钟了，要开始及时计价了；

**2.3.7 起步价内计价模块**

****

图xx starting\_price模块引脚图

* **基本功能**

若night深夜行车状态电平输入1，总起步价输出夜间行车起步价；若night深夜行车状态电平输入0，总起步价输出白天行车起步价。该模块受状态机输出state\_pricing信号输入跳转。day\_starting\_price输入白天起步价，白天起步价可以根据政策和市场修改；night\_starting\_price输入夜晚起步价，夜晚起步价可以根据政策和市场修改；

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

load:复位信号，输入1有效，通过状态机切换状态输入;

day\_starting\_price：输入白天起步价 ，白天起步价可以根据政策和市场修改；

night\_starting\_price输入夜晚起步价，夜晚起步价可以根据政策和市场修改；

state\_pricing：状态机跳转信号，输入2起步价内计价模块开始工作，输入3，起步价内计价模块复位；

night: 深夜行车状态电平输入1，总起步价输出夜间行车起步价；若night深夜行车状态电平输入0，总起步价输出白天行车起步价

* **输出信号及具体参数值**

total:起步价价格；

**2.3.8 起步价外计价模块**

****

图xx unit\_price模块引脚图

* **基本功能**

在起步价外，接受到一个计价里程dp脉冲输入，里程价格开始计价；接受到一个等待分钟minin脉冲输入，计时价格开始计价；若night深夜行车状态电平输入1，按照夜间各种价格收费；若night深夜行车状态电平输入0，按照白天各种价格收费；各种收费价格都是可以根据政策或者市场被价格设置模块修改的；最后输出里程外计价的总的价格；

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

load:复位信号，输入1有效，通过状态机切换状态输入;

state\_pricing：状态机跳转信号，输入3起步价外计价模块开始工作；

night: 深夜行车状态电平输入1，价格按照夜间行车收取；若night深夜行车状态电平输入0，价格按照白天收取；

yuancheng: 远程行车状态电平输入1，价格按照远程行车收取；若远程行车状态电平输入0，价格按照白远程行车内收取；

day\_starting\_price：输入白天起步价 ，白天起步价可以根据政策和市场修改；

night\_starting\_price输入夜晚起步价，夜晚起步价可以根据政策和市场修改；

day\_mileage\_price :输入白天里程计价，白天里程计价可以根据政策和市场修改；

day\_timing\_price : 输入白天计时计价，白天计时计价可以根据政策和市场修改；

day\_long\_distance\_price : 输入白天长途计价，白天长途计价可以根据政策和市场修改；

* **输出信号及具体参数值**

total\_price：输出起步价外车费；

**2.3.9 总里程脉冲生成统计模块**

* **总里程脉冲生成模块**



图xx total\_run模块引脚图

* **基本功能**

当复位使能开始行驶后，此模块一直统计车轮脉冲din，一个车轮脉冲din表示的是出租车行进2m，在统计满500个din脉冲后，即到达一公里，输出一个里程脉冲，输入到脉冲统计模块统计里程脉冲，即可得到行驶的总里程。

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

clk：系统时钟信号，数据类型为std\_logic。频率为10Hz；

load：复位信号，输入1有效，通过状态机切换状态输入；

din：车轮脉冲信号，每输入一个车轮脉冲信号，表示出租车前行2m。

* **输出信号及具体参数值**

dp\_total：统计车轮脉冲后输出的里程脉冲，输出到脉冲统计模块统计行驶的总里程。

* **总里程脉冲统计模块**



图xx countplus模块引脚图

countplus模块用于统计里程脉冲，用于显示车辆行驶总距离，统计等待时间脉冲用于显示车辆行驶总时间。

* **基本功能**

统计total\_run模块输出的出租车行驶里程脉冲，通过total输出给分位模块，用于数码管显示。

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

load:复位信号，输入1有效，通过状态机切换状态输入;

din:里程脉冲信号，输入1表示出租车行驶过一个里程，用于统计出租车的行驶里程；

* **输出信号及具体参数值**

total：出租车的行驶总里程，输出到分位模块进行分位，便于数码管显示；

**2.3.10 总计时脉冲生成统计模块**

* **总计时脉冲生成模块**



图xx drivetime模块引脚图

* **基本功能**

统计系统时钟clk脉冲个数，一个clk脉冲周期是0.1s,在统计600个clk时钟脉冲后，即达到1分钟，输出一个行驶时间分钟脉冲dout，dout输出给countplus模块统计行驶的总时间，用于数码管显示。

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入；

load:复位信号，输入1有效，通过状态机切换状态输入；

din:时钟脉冲信号，系统时钟clk通过该端口输入，输入一个脉冲表示出租车已经行驶了0.1s。

* **输出信号及具体参数值**

dout:行驶分钟脉冲，每输出一个dout行驶分钟脉冲，表示车辆又行驶了一分钟。输出给统计脉冲模块行驶分钟脉冲总数，得到统计行驶总时间，用于数码管显示行驶时间；

* **总计时脉冲统计模块**



图7xx countplus模块引脚图

* **基本功能**

countplus模块用于统计里程脉冲，用于显示车辆行驶总距离，统计等待时间脉冲用于显示车辆行驶总时间。输入信号及具体参数值

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

load：复位信号，输入1有效，通过状态机切换状态输入；

din：行驶时间分钟脉冲信号，从模块drivetime的dout端口输入，输入1表示出租车行驶一分钟，用于统计出租车的行驶时间；

* **输出信号及具体参数值**

total：出租车的行驶总时间，精确到分钟，输出到分位模块进行分位，便于数码管显示；

**2.3.11 分位模块**

* **价格和里程分位模块**

****

图xx separateprice模块引脚图

因为价格和里程都是十进制的，所以共用一个分位模块。

* **基本功能**

从里程内计价模块，里程外计价模块输入要分位的总行驶价格；从countplus模块输入总行驶里程。该模块可以分位的范围0-9999，分位后，输入总数的四位分别以独立的四位输出，输出到对应的数码管模块数显示；将输入的四位行驶里程分别以独立的四位输出，便于数码管显示。

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

clk：系统时钟信号，数据类型为std\_logic。频率为10Hz；

load：复位信号，输入1有效，通过状态机切换状态输入；

total:要拆成四位便于数码管显示的价格；

* **输出信号及具体参数值**

ge,shi,bai,qian:价格的个位，十位，百位，千位的数值；

* **时间分位模块**



图xx separatetime模块引脚图

* **基本功能**

从drivetime模块输入要分位的总行驶价格；该模块可以分位的范围00:00至23:59，分位后，输入总数的四位分别以独立的四位输出，输出到对应的数码管模块数显示；将输入的四位行驶时间分别以独立的四位输出，便于数码管显示。

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

clk：系统时钟信号，数据类型为std\_logic。频率为10Hz；

load：复位信号，输入1有效，通过状态机切换状态输入；

totaltime:要拆成四位便于数码管显示的时间；

* **输出信号及具体参数值**

ffen,sfen,fh,sh：时间的第一个分数，第二个分数，第一个小时数，第二个小时数；

**2.3.12 选择模块**

* **价格选择模块**



图xx or2total模块引脚图

* **基本功能**

ce使能信号由key\_start输入，当ce输入为1的时候，使能模块。输入信号total1从起步价外计价模块输入；输入信号total2从起步价内计价模块输入；输入信号price从设置价格模块选择模块or5price输入。如果处于设置状态，pricesel输入1，此时total选择输出price信号，当处于正常的显示状态的时候，即pricesel输入0的时候，此时total根据输入情况选择输出价格。在起步价内计价的状态下，total输出total2价格；在起步价外计价状态下，total输出total1的价格。

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

pricesel：设置价格标志信号，数据类型是std\_logic，输入1表示在设置价格状态，输入0表示正常计价状态，该信号通过set\_price模块的pricesel输入；

total1:起步价外计价模块输入，输入起步价外的出租车车费；

total2:起步价内计价模块输入，输入起步价内的出租车车费；

price:设置价格状态设置的价格。根据选择位不同，可以是白天起步价，夜晚起步价，白天里程价，夜晚里程价，白天等待价，夜晚等待计价，白天远程计价，夜晚远程计价，此处通过选择模块在数码管上显示便于修改价格；

* **输出信号及具体参数值**

total:经过不同条件筛选输出的价格，在pricesel输入1的状态下，total输出的是设置的价格；在pricesel输入0的状态下，输入total1，那么total便输出total,若输入total2，那么total便输出total。

* **时间选择模块**

****

图xx or2time模块引脚图

* **基本功能**

辅助显示总的行驶时间和设置夜间模块的时间。当timesel为1的时候，即设置时间模式，输出要正在设置的时间；当timesel为0的时候，输出正常要显示驾驶时间；

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

timesel：时间设置状态，输入1表示正在进行按键设置模式，输入0表示正常显示时间模式；

ffen,sfen,fh,sh：正常出租车行驶时间的经过分位模块输出的四个位；

vffen,vsfen,vfh,vsh：设置出租车夜间时间的经过分位模块输出的四个位；

* **输出信号及具体参数值**

rffen,rsfen,rfh,rsh：输出经过选择的时间的四个位；

* **设置价格选择模块**

****

图xx or5price模块引脚图

* **基本功能**

设置价格的时候选出价格，输出端口输出5个输入价格有输入信号的价格；

* **输入信号及具体参数值**

ce：使能信号，输入1有效，通过按键key\_start输入；

pricesel：设置价格标志信号，数据类型是std\_logic，输入1表示在设置价格状态，输入0表示正常计价状态，该信号通过set\_price模块的pricesel输入；

day\_starting\_price：输入白天起步价 ，白天起步价可以根据政策和市场修改；

night\_starting\_price输入夜晚起步价，夜晚起步价可以根据政策和市场修改；

day\_mileage\_price :输入白天里程计价，白天里程计价可以根据政策和市场修改；

day\_timing\_price : 输入白天计时计价，白天计时计价可以根据政策和市场修改；

day\_long\_distance\_price : 输入白天长途计价，白天长途计价可以根据政策和市场修改；

**输出信号及具体参数值**

price：经过选择输出的价格；

**2.3.13 显示模块**

* **价格显示模块**



图xx displayprice模块引脚图

* **基本功能**

4个数码管显示999.9元以内的车费，车费精确到角。输出车费价格的位选和段选。

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

clk:系统时钟信号，数据类型为std\_logic。频率为10Hz;

load:复位信号，输入1有效，通过状态机切换状态输入;

a:输入待显示的价格的千位。数据类型是std\_logic\_vector(3 downto 0)；

b:输入待显示的价格的个位。数据类型是std\_logic\_vector(3 downto 0)；

c:输入待显示的价格的十位。数据类型是std\_logic\_vector(3 downto 0)；

d:输入待显示的价格的百位。数据类型是std\_logic\_vector(3 downto 0)；

* **输出信号及具体参数值**

sel:数码管位选信号，将具体要显示的信号传给某个数码管显示；

outp:数码管显示段选，待显示的信号按照共阴极数码管显示的编码显示；

* **时间显示模块**



图xx displaytime模块引脚图

* **基本功能**

4个数码管以“xx:xx”格式显示出租车运行的时间，时间精确到分钟。输出行驶时间的位选和段选。

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

clk:系统时钟信号，数据类型为std\_logic。频率为10Hz;

load:复位信号，输入1有效，通过状态机切换状态输入;

ffen:输入待显示的时间的第一个分钟位置，数据类型是std\_logic\_vector(3 downto 0)；

sfen:输入待显示的时间的第二个分钟位置，数据类型是std\_logic\_vector(3 downto 0)；

fh:输入待显示的时间的第一个小时位置。数据类型是std\_logic\_vector(3 downto 0)；

sh:输入待显示的时间的第二个小时位置。数据类型是std\_logic\_vector(3 downto 0)；

* **输出信号及具体参数值**

sel:数码管位选信号，将具体要显示的信号传给某个数码管显示；

outp:数码管显示段选，待显示的信号按照共阴极数码管显示的编码显示；第二个数码管显示带小数点的数值，第三个数码管倒着显示，和第二个配合显示时间的格式；

* **里程显示模块**



图xx displaydistance模块引脚图

* **基本功能**

4个数码管以“xxxx”格式显示出租车实际行驶的距离，里程距离精确到公里。在接收到里程分离模块输出的四个位参数后，输出行驶距离的的位选和数码管显示段选。

* **输入信号及具体参数值**

ce:使能信号，输入1有效，通过按键key\_start输入;

clk:系统时钟信号，数据类型为std\_logic。频率为10Hz;

load:复位信号，输入1有效，通过状态机切换状态输入;

ge,shi,bai,qian:里程分离模块分离的总里程的个位，十位，百位，千位的实际数值。

* **输出信号及具体参数值**

sel:数码管位选信号，将具体要显示的信号传给某个数码管显示；

outp:数码管显示段选，待路程的四路要显示的信号按照共阴极数码管显示的编码显示。四位数码管显示的值都是不带小数点的；

**2.3.14 蜂鸣器跳表提醒模块**



图xx bee模块引脚图

* **基本功能**

输出bee\_pulse驱动蜂鸣器，在计价器计价要跳表的时候，输出一个脉冲，驱动蜂鸣器发出响声，提醒乘客。无论是起步价，里程计价，计时计价都能准确做出响应。

* **输入信号及具体参数值**

ce：使能信号，数据类型是std\_logic，输入1有效，通过按键key\_start输入；

qibu：起步信号，数据类型是std\_logic，输入1表示已经是起步价外里程了，输入0表示还在起步价内，qibu电平由licheng模块输入；

run\_pulse：里程脉冲信号，数据类型是std\_logic，当输入一个脉冲表示出租车行进1km，计价模块即将进行里程计价，run\_pulse由licheng模块输入；

wait\_pulse：等待时间脉冲，数据类型是std\_logic，当输入一个脉冲表示出租车已经等待一分钟，计价模块即将进行等待时间计价，wait\_pulse由licheng模块输入。

* **输出信号及具体参数值**

bee\_pulse：蜂鸣器脉冲，数据类型是std\_logic，当输出一个脉冲的时候，提醒乘客出租车计价器即将跳表。

三、模块实现与仿真

3.1按键消抖模块

3.1.1关键VHDL源代码及解说

--按键消抖

**if** **(** clk'**event** **and** clk **=** '1'**)**

**then**

**if** **(** key\_in **=** '1' **)** --检测有按键按下

**then**

**if** **(** count **=** N **)**

**then**

count **<=** count **;**-- 计数器值为N时保持状态

**else**

count **<=** count **+** 1**;** -- 否则计数器+1

**end** **if;**

**代码解说：**计数器型消抖电路是设置一个控制计数器，当clk在上升沿时，如果按键开关key\_in=1，计数器加1，key\_in=0时，计数器清零。

**if** **(** count **=** N **-** 1**)** --当持续大于2个时钟周期时，输出单脉冲

**then**

key\_out **<=** '0'**;**

**else**

key\_out **<=** '1'**;**

**end** **if;**

**else**

count **<=** 0**;** --没有按键按下，计数器清0

**end** **if;**

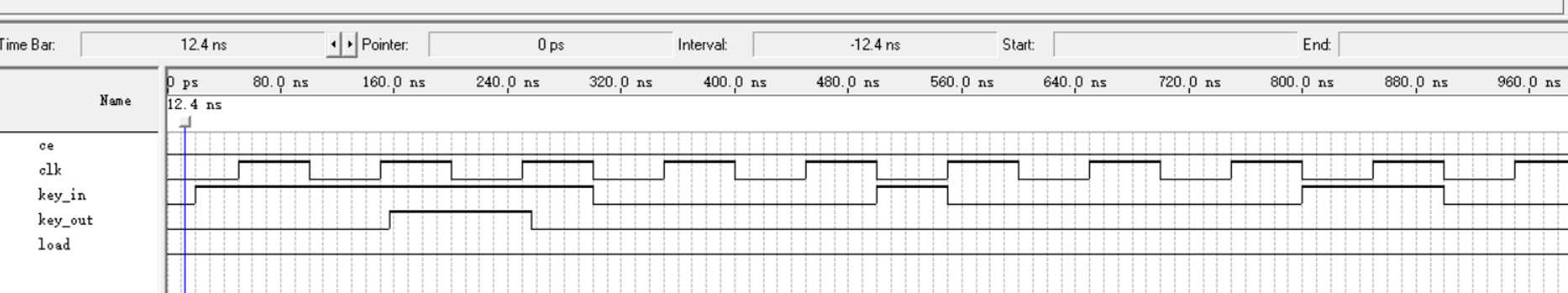
**end** **if;**

**代码解说：**按键key\_in持续时间大于2个clk时钟周期时，计数器输出一个单脉冲，否则没有脉冲输出。

**3.1.2仿真输入信号的设计与分析**

clk输入系统时钟，为了仿真节约时间，设置频率为10KHz。Key\_in模块按键输入和毛刺信号。超过三个clk时钟周期的认为是按键输入，小于三个clk时钟周期的是毛刺信号。

**3.1.3 仿真结果时序图、简要分析**



图xx 按键消抖模块仿真结果

开始key\_in输入一个按键信号，经过消抖得到一个key\_out输出，接着key\_in输入两个毛刺信号，都被消除掉了，仿真结果正确。

**3.2控制模块**

**3.2.1关键VHDL源代码及解说**

**A、状态机定义代码**

**entity** state\_controller **is**

**port(**

st **:** **in** std\_logic**;**

--qibu脉冲

qibu **:** **in** std\_logic**;**

clk **:** **in** std\_logic**;**

tenin **:** **in** std\_logic**;**

load **:** **out** std\_logic**;**

--输出计价状态

state\_pricing **:** **out** integer **range** 1 **to** 4

**);**

**end** **entity** state\_controller**;**

**architecture** control **of** state\_controller **is**

**type** FSM\_ST **is** **(** start\_reset **,** stop **,** within\_starting\_price **,** beyond\_starting\_price **);**

--start\_reset ：开始并复位；stop：停止计价 ； within\_starting\_price：起步价内 ；beyond\_starting\_price：起步价外

**signal** current\_state **,** next\_state **:** FSM\_ST**;**

**代码解说：**定义了状态机的四个状态，分别是：开始复位start\_reset、停止计价stop、起步价内计价within\_starting\_price和起步价外计价beyond\_starting\_price。

**B、状态机核心模块：**

REG **:** **process(**clk **,**st**)** --主控时序进程

**begin**

**if** st **=** '0' **then**

current\_state **<=** stop **;**

**elsif** clk**=**'1' **and** clk'**event** **then**

current\_state **<=** next\_state**;**

**end** **if;**

**end** **process** REG**;**--

**代码解说：**当计价启动信号st=0时，汽车处于停止计价stop状态，每当一个时钟上升沿到来，改变状态机的状态。

COM **:** **process(** current\_state **,** qibu **,** st **,** tenin **)** --主控组合进程

**begin**

**case** current\_state **is**

**when** start\_reset **=>**

load **<=** '1'**;**

state\_pricing **<=** 1**;**

next\_state **<=** within\_starting\_price**;**

**代码解说：** 当是开始复位start\_reset状态时，输出复位信号load=1，使上一次计价清零，计价状态state\_pricing=1，当下一个时钟上升沿到来，进入下一状态起步价内计价within\_starting\_price。

**when** within\_starting\_price **=>**

load **<=** '0'**;**

state\_pricing **<=** 2**;**

**if** st **=** '0' **then**

next\_state **<=** stop**;**

**end** **if;**

**if** st **=** '1' **and** qibu **=** '1' **then**

next\_state **<=** beyond\_starting\_price**;**

**end** **if;**

**if** st **=** '1' **and** qibu **=** '0' **and** tenin **=** '1' **then**

next\_state **<=** beyond\_starting\_price**;**

**end** **if;**

**if** st **=** '1' **and** qibu **=** '0' **and** tenin **=** '0' **then**

next\_state **<=** within\_starting\_price**;**

**end** **if;**

**代码解说：**当是起步价内计价within\_starting\_price时，复位信号load=0，输出计价状态state\_pricing=2，使能起步价内计价模块starting\_price，开始计价，并显示到计价数码管displayprice上。如果此时，启动信号st=0，即到达目的地停车，下一状态为停止计价stop状态；若接收到2km起步价外信号qibu且汽车仍处于启动状态，下一状态转为起步价外计价beyond\_starting\_price；若仍在2km内，但计时计价时间超过十分钟，即tenin=1，下一状态为起步价外计价beyond\_starting\_price；只有当里程小于2km且低速计时时间小于10min时，仍为起步价内起价within\_starting\_price。

**when** beyond\_starting\_price **=>**

load **<=** '0'**;**

state\_pricing **<=** 3**;**

**if** st **=** '0' **then**

next\_state **<=** stop**;**

**end** **if;**

**if** st **=** '1' **and** qibu **=** '1' **then**

next\_state **<=** beyond\_starting\_price**;**

**end** **if;**

**代码解说：**当是起步价外计价beyond\_starting\_price时，复位信号load=0，输出计价状态state\_pricing=3，使能起步价外计价模块unit\_price，开始计价，并显示到计价数码管displayprice上。如果此时，启动信号st=0，即到达目的地停车，下一状态为停止计价stop状态；若仍接收到2km起步价外信号qibu且汽车仍处于启动状态，下一状态仍为起步价外计价beyond\_starting\_price。

**when** stop **=>**

load **<=** '0'**;**

state\_pricing **<=** 4**;**

**if** st **=** '1' **then**

next\_state **<=** start\_reset **;**

**end** **if;**

**when** **others** **=>**

load **<=** '0'**;**

state\_pricing **<=** 4**;**

next\_state **<=** stop**;**

**end** **case;**

**end** **process** COM**;**

**代码解说：**当其他情况时，均跳转至停止计价stop状态，复位信号load=0，输出计价状态state\_pricing=4，不再使能计价模块，显示最终价格。当再次收到启动信号st=1，重新进入开始复位start\_reset状态。

**3.2.2仿真输入信号的设计与分析**

**3.2.3 仿真结果时序图、简要分析**

**3.3高低速判断模块**

**3.3.1关键VHDL源代码及解说**

**关键VHDL源代码：**

**if** clk**=**'1' and clk'**event** **then**

**if** time **<** 21 **then**

time**:=**time**+**1**;**

**else**

time**:=**21**;**

**end** **if;**

**end** **if;**

**if** din**=**'1' **then**

time**:=**0**;**

**end** **if;**

**if** time**>**20 **then**

ceout**<=**'0'**;**

**else**

ceout**<=**'1'**;**

**end** **if;**

**代码解说：**检测到clk上升沿后，通过变量time来判断出租车的速度是否达到1m/s。系统的一个车轮脉冲输入din表示出租车已经行进了2/m。clk的频率是10Hz,在两个车轮脉冲din之间，如果time>20，那么出租车的行驶速度就小于1m/s；如果time<=20,那么出租车的行驶速度就大于1m/s。

**关键VHDL源代码：**

**if** load**=**'0' **then**

**.....**

**else**

time**:=**0**;**

ceout**<=**'0'**;**

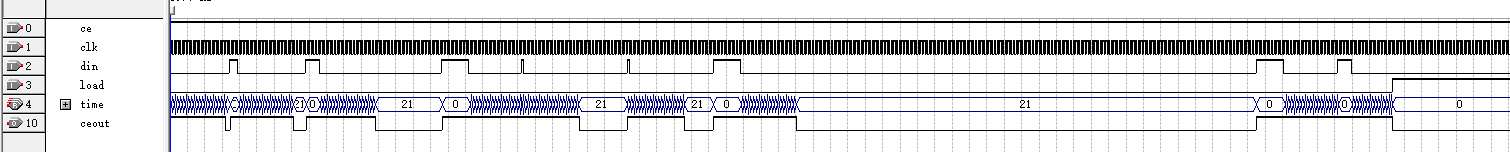
**end** **if;**

**代码解说：**在复位信号load到来之后，对变量和输出进行复位。

**3.3.2仿真输入信号的设计与分析**

clk时钟信号频率是10Hz。为节约时间，仿真的时候设置为10KHz,一共仿真50us。车轮脉冲din输入随机信号脉冲，模拟行车的不停变换速度的状况。在最后输入load电平信号，模拟复位的情况。

**3.3.3 仿真结果时序图、简要分析**



图xx

ceout输出为1，表示现在是高速行车状态，此时应该是里程计价，当ceout输出为0,此时是低速行车状态，应该是按照计时计价。当load复位电平信号到来的时候，变量time和输出ceout全部置0。该模块仿真结果达到了预期的效果。

**3.4 夜间行车模块**

**3.4.1关键VHDL源代码及解说**

**A、初始时间设置模块**

**关键VHDL源代码：**

**if** **rising\_edge(** clk **)**

**then**

**if** time\_state **=** '1' --设置按键按下

**then**

**if** time\_sel **=** '0' **then**

time\_sel **:=** '1'**;**

**elsif** time\_sel **=** '1' **then**

time\_sel **:=** '0'**;**

**end** **if;**

**end** **if;**

**代码解说：**按下key\_time进入设置模式，再次按下退出。

**if** time\_sel **=** '1' **then**

**if** **(**sel **=** '1'**)** **and** **(**bit\_sel **<** 3**)** --位选

**then**

bit\_sel **:=** bit\_sel **+** 1**;**

**elsif** **(**sel **=** '1' **)and** **(**bit\_sel **=** 3**)**

**then**

bit\_sel **:=** 0**;**

**else** bit\_sel **:=** bit\_sel**;**

**end** **if;**

**代码解说：**在设置模式中，按下选位键key\_select,选择要设置的位，总共有四位，表示几时几分。

**if** set **=** '1' --+1

**then**

**case** **(**bit\_sel**)** **is** -- 四位分别加值

**when** 0 **=>** --00:00第四位

**if** d **<** 9

**then**

d **:=** d **+** 1**;**

**elsif** d **=** 9

**then**

d **:=** 0**;**

**end** **if;**

**代码解说：**设置分钟的个位，从0加到9后，再次回到0循环设置。

**when** 1 **=>** --00:00第三位

**if** c **<**5

**then**

c **:=** c **+** 1**;**

**elsif** c **=** 5

**then**

c **:=** 0**;**

**end** **if;**

**代码解说：**设置分钟的十位，从0加到5后，再次回到0循环设置。

**when** 2 **=>** --00:00二位

**if** **((**a **<** 2**)** **and** **(**b **<** 9**))**

**then**

b **:=** b **+** 1**;**

**elsif** **((**a **<** 2**)** **and** **(**b **=** 9**))**

**then**

b **:=** 0**;**

**end** **if;**

**if** **((**a **=** 2**)** **and** **(**b **<**3**))**

**then**

b **:=** b **+** 1**;**

**elsif** **((**a **=** 2**)** **and** **(**b **=**3**))**

**then**

b **:=** 0**;**

**end** **if;**

**代码解说：**设置时钟的个位，当时钟的十位为0或1时，个位可从0加到9后，再次回到0循环设置；当时钟的十位为2时，个位可从0加到3，再回到0循环设置。

**when** 3 **=>** --00:00第一位

**if** a **<** 2

**then**

a **:=** a **+**1**;**

**elsif** **(**a **=** 2**)**

**then**

a **:=** 0 **;**

**end** **if;**

**when** **others** **=>**

**null;**

**end** **case;**

**代码解说：**设置时钟的十位，从0-2循环设置。

hour\_s **<=** a**;**

hour\_f **<=** b**;**

min\_s **<=** c**;**

min\_f **<=** d**;**

**if** **(** confirm **=** '1' **)** --退出设置模式，位选复位，时间不变

**then**

bit\_sel **:=** 0**;**

time\_sel **:=** '0'**;**

**end** **if;**

**代码解说：**输出四个时间位的数值，按下key\_confirm退出设置模式，位选复位。

**B、初始时间转化模块**

**关键VHDL源代码：**

u25\_2 **:** mult10b

**port** **map** **(**q **=>** mul1 **,** d1 **=>** c **,** d2 **=>** ten**);**

u25\_3 **:** mult10b

**port** **map** **(**q **=>** mul2 **,** d1 **=>** b **,** d2 **=>** sixty**);**

u25\_4 **:** mult10b

**port** **map** **(**q **=>** mul3 **,** d1 **=>** a **,** d2 **=>** six\_hundreds **);**

--累加得出分钟数

**process(**clk **,** d **,** mul1 **,** mul2 **,** mul3**)**

**begin**

**if** clk'**event** **and** clk**=**'1' **then**

transition **<=** **(**d **+** mul1 **+** mul2 **+** mul3**)** **;**

settime**(**10 **downto** 0**)** **<=** transition**(**10 **downto** 0**);**

**end** **if;**

**end** **process;**

**代码解说：**调用10位乘法器，将时间四位数转化为分钟表示，分钟十位乘十，时钟个位乘六十，时钟十位乘六百后，再将三个结果与分钟个位相加，得出总分钟数。

**C、凌晨行车判断模块**

**关键VHDL源代码：**

**if** clk'**event** **and** clk**=**'1' **then**

**if** timesel**=**'1' **then**

mins**:=**settime**;**

**end** **if;**

**if** time**<** 60 **then**

--计数加一

time**:=**time **+** 1**;**

**else**

time**:=**0**;**

mins**:=**mins**+**1**;**

**end** **if;**

**代码解说：**若处于设置时间状态，设置了初始时间，则让分钟数mins等于初始时间settime，当秒数累加到60，分钟数+1，秒数计数器清零。

--6h

**if** mins **<** 360 **then**

ceout**<=**'1'**;**

--23h

**elsif** mins **>** 1380 **and** mins **<** 1441 **then**

ceout**<=**'1'**;**

--输出为0，让后面的计数器先不要计数

**代码解说：**当分钟数mins小于360，大于1380且小于1441时，即此时运行时间在凌晨23h至次日6h之间，为夜间行车，输出为1。

**elsif** mins **>** 1440 **then**

time**:=**0**;**

mins**:=(others=>**'0'**);**

--输出1，起步价结束，后面的计数器计数开始

**else**

--6h-23h

ceout**<=**'0'**;**

**end** **if;**

**代码解说：**当分钟数mins在360至1380之间，即此时运行时间在白天，输出为0，当mins达到1441时，清零，为第二天计时。

**3.4.2仿真输入信号的设计与分析**

**A、初始时间设置模块**

**B、初始时间转化模块**

**C、凌晨行车判断模块**

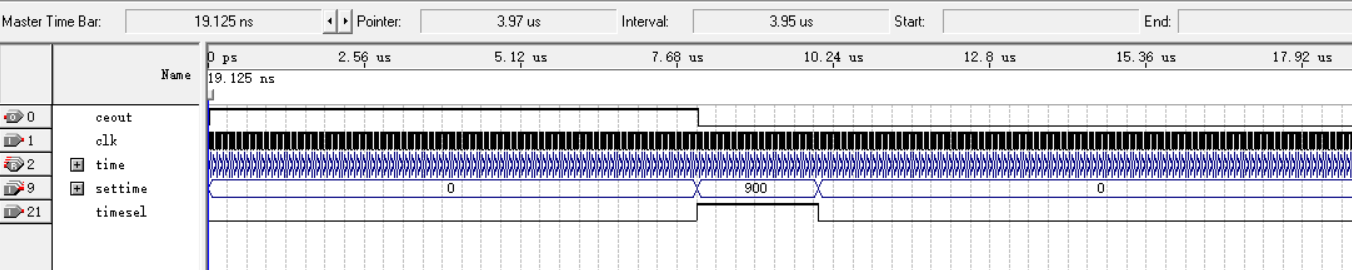
系统时钟clk设为10KHz;系统开机默认是从0:00开始计时的，设一个timesel高电平，设置settime为900，即下午三定的时间，看ceout输出。

**3.4.3 仿真结果时序图、简要分析**

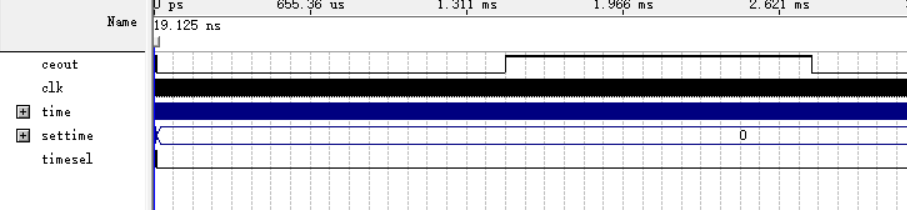
**A、初始时间设置模块**

**B、初始时间转化模块**

**C、凌晨行车判断模块**



图xx 凌晨行车判断模块仿真结果图1



图xx 凌晨行车判断模块仿真结果图2

一开始ceout输出1,表示处于凌晨的状态，后来设置初始时间，设置到了下午三点，ceout输出为0，表示白天行车。后面从下午三点开始计时，计时记到晚上11点，ceout跳转高电平，表明夜间行车状态。仿真结果正确。

**3.5价格调控模块**

**3.5.1关键VHDL源代码及解说**

**A、五种价格设置模块**

**if** **rising\_edge(** clk **)** **then**

**if** price\_state **=** '1' **and** price\_sel **<**5 **then**

price\_sel **:=** price\_sel **+** 1**;**

pricesel**<=**price\_sel**;**

**elsif** price\_state **=** '1' **and** price\_sel **=**5 **then**

price\_sel **:=** 1**;**

pricesel**<=**price\_sel**;**

**end** **if;**

**case** price\_sel **is**

**when** 1 **=>**

bit\_sel **:=** 0**;**

**if** **(**sel **=** '1'**)** **and** **(**bit\_sel **<** 3**)** --位选

**then**

bit\_sel **:=** bit\_sel **+** 1**;**

**elsif** **(**sel **=** '1' **)and** **(**bit\_sel **=** 3**)**

**then**

bit\_sel **:=** 0**;**

**else** bit\_sel **:=** bit\_sel**;**

**end** **if;**

**if** set **=** '1' --+1

**then**

**case** bit\_sel **is**

**when** 0 **=>**

**if(** day\_starting\_price\_jiao **<** 9 **)** **then**

day\_starting\_price\_jiao **:=** day\_starting\_price\_jiao **+**1**;**

**elsif** **(**day\_starting\_price\_jiao **=** 9**)** **then**

day\_starting\_price\_jiao **:=** 0**;**

**end** **if;**

**when** 1 **=>**

**if(** day\_starting\_price\_yuan **<** 9 **)** **then**

day\_starting\_price\_yuan **:=** day\_starting\_price\_yuan **+**1**;**

**elsif** **(**day\_starting\_price\_yuan **=** 9**)** **then**

day\_starting\_price\_yuan **:=** 0**;**

**end** **if;**

**when** 2 **=>**

**if(** day\_starting\_price\_ten **<** 9 **)** **then**

day\_starting\_price\_ten **:=** day\_starting\_price\_ten **+**1**;**

**elsif** **(**day\_starting\_price\_ten **=** 9**)** **then**

day\_starting\_price\_ten **:=** 0**;**

**end** **if;**

**when** 3 **=>**

**if(** day\_starting\_price\_hundred **<** 9 **)** **then**

day\_starting\_price\_hundred **:=** day\_starting\_price\_hundred **+**1**;**

**elsif** **(**day\_starting\_price\_hundred **=** 9**)** **then**

day\_starting\_price\_hundred **:=** 0**;**

**end** **if;**

**when** **others** **=>**

**null;**

**end** **case;**

**end** **if;**

**when** **others** **=>**

bit\_sel **:=** 0**;**

price\_sel **:=** 0**;**

pricesel**<=**0**;**

day\_starting\_price\_j **<=** day\_starting\_price\_jiao **;**

day\_starting\_price\_y **<=** day\_starting\_price\_yuan **;**

day\_starting\_price\_t **<=** day\_starting\_price\_ten **;**

day\_starting\_price\_h **<=** day\_starting\_price\_hundred**;**

**if** confirm **=** '1' **then**

price\_sel **:=** 0**;**

pricesel**<=**0**;**

**end** **if;**

**代码解说：**-

**B、价格转化模块**

u28\_2**:** mult10b

**port** **map** **(**q **=>** mul11 **,** d1 **=>** day\_starting\_price\_a **,** d2 **=>** thousand **);**

u28\_3**:** mult10b

**port** **map** **(**q **=>** mul12 **,** d1 **=>** day\_starting\_price\_b **,** d2 **=>** hundred**);**

u28\_4**:** mult10b

**port** **map** **(**q **=>** mul13 **,** d1 **=>** day\_starting\_price\_c **,** d2 **=>** ten**);**

**if** clk'**event** **and** clk**=**'1' **then**

**if** price\_state**=**'1' **then**

transition1 **<=** **(**day\_starting\_price\_d **+** mul11 **+** mul12 **+** mul13**)** **;**

day\_starting\_price **<=** conv\_integer **(**transition1 **);**

**代码解说：**-

**3.5.2仿真输入信号的设计与分析**

**A、五种价格设置模块**

**B、价格转化模块**

**3.5.3 仿真结果时序图、简要分析**

**A、五种价格设置模块**

**B、价格转化模块**

**3.6 里程计价脉冲生成模块**

**3.6.1关键VHDL源代码及解说**

**关键VHDL源代码：**

**if** flag**=**0 **then**--起步状态下里程计数不受到高低速状态的影响

**if** din**=**'1' **then**

--不足500m

**if** run **<** 250 **then**

run**:=**run**+**1**;**

dp**<=**'0'**;**

**else**

--正好500m

**if** run**=**250 **then**

twofive**:=**twofive**+**1**;**

**end** **if;**

**if** run**<**500 **then**--

run**:=**run**+**1**;**

dp**<=**'0'**;**

**else**

twofive**:=**twofive**+**1**;**

dp**<=**'1'**;**

twenty**:=**twenty**+**1**;**

run**:=**"0000000000000"**;**

**end** **if;**

**end** **if;**

**end** **if;**

**if** twenty **<** 12 **then**

yuancheng**<=**'0'**;**

**else**

yuancheng**<=**'1'**;**

--13

twenty**:=**"1101"**;**

**end** **if;**

--判断是否到达起步里程

**if** twofive **<** 4 **then**

qibu**<=**'0'**;**

**else**

qibu**<=**'1'**;**

flag**<=**1**;**

dp**<=**'1'**;**

--5

twofive**:=**"101"**;**

**end** **if;**

**end** **if;**

**代码解说：**引入变量flag是为了表明里程计价的状态，flag=0,即此时是起步价状态，里程计价不受到hl模块输入的高低速输入电平影响。引入变量run统计车轮脉冲din,当检测到一个车轮脉冲din，run自加1；当检测到250个车轮脉冲，即车辆已经行驶了500m,用于协助输出起步电平的变量twofive自加1；当检测到500个车轮脉冲，即车辆已经行驶了1km, 输出一个里程脉冲dp, 用于协助输出远程电平变量twenty自加1,变量run清0 。当变量twofive<4时，出租车每行驶500m, 变量twofive自加1，此时qibu端口一直输出低电平，显示出租车处于起步价内的状态。在行驶达到2km，开始转变为起步外计价状态后，模块需要输出一个起步电平，表明已经处于起步价外状态，为了稳定，给变量twofive赋值为5，此时qibu端口一直输出高电平。起步价外计价的状态，因为高低速的影响，所以在起步价外计价的时候，高速的时候，需要统计里程脉冲用于里程计价，低速的时候由另外的等待时间模块统计等待时间脉冲用于等待时间计价，此时统计车轮脉冲暂停，给flag赋值1，进入起步里程外的里程统计代码。

**关键VHDL源代码：**

**if** cein**=**'1' and flag**=**1 **then**--高速状态下进行里程计价

**if** din**=**'1' **then**

--不足500m

**if** run **<** 250 **then**

run**:=**run**+**1**;**

dp**<=**'0'**;**

**else**

--正好500m

**if** run**=**250 **then**

twofive**:=**twofive**+**1**;**

**end** **if;**

**if** run**<**500 **then**

run**:=**run**+**1**;**

dp**<=**'0'**;**

**else**

twofive**:=**twofive**+**1**;**

dp**<=**'1'**;**

twenty**:=**twenty**+**1**;**

run**:=**"0000000000000"**;**

**end** **if;**

**end** **if;**

**end** **if;**

**if** twenty **<** 12 **then**

yuancheng**<=**'0'**;**

**else**

yuancheng**<=**'1'**;**

--13

twenty**:=**"1101"**;**

**end** **if;**

--判断是否到达起步里程

**if** twofive **<** 4 **then**

qibu**<=**'0'**;**

**else**

qibu**<=**'1'**;**

flag**<=**1**;**

--5

twofive**:=**"101"**;**

**end** **if;**

**end** **if;**

**代码解说：**在输入cein为1的时候，即高速状态下，并且满足flag等于1才执行高速状态下的里程计价。在执行完起步价内计价代码后，变量twofive一直被赋值为5，qibu端口一直输出高电平，表明目前处于起步价外计价状态。此时在高速状态下，统计出租车每行进一公里，输出一个里程计价脉冲dp到起步价外计价模块用于里程计价，变量twenty自加1，此时yuancheng端口输出为0，表明此时尚未到达远程行驶的状态。当变量twenty达到12，即已经行驶了12km，达到远程行驶的状态，需要一直输出远程行驶状态电平yuancheng为1，给变量twenty赋值为13。

**关键VHDL源代码：**

**if** load**=**'0' **then**

**......**

**else**

qibu**<=**'0'**;**

yuancheng**<=**'0'**;**

run**:=**"0000000000000"**;**

twofive**:=(others=>**'0'**);**

twenty**:=(others=>**'0'**);**

dp**<=**'0'**;**

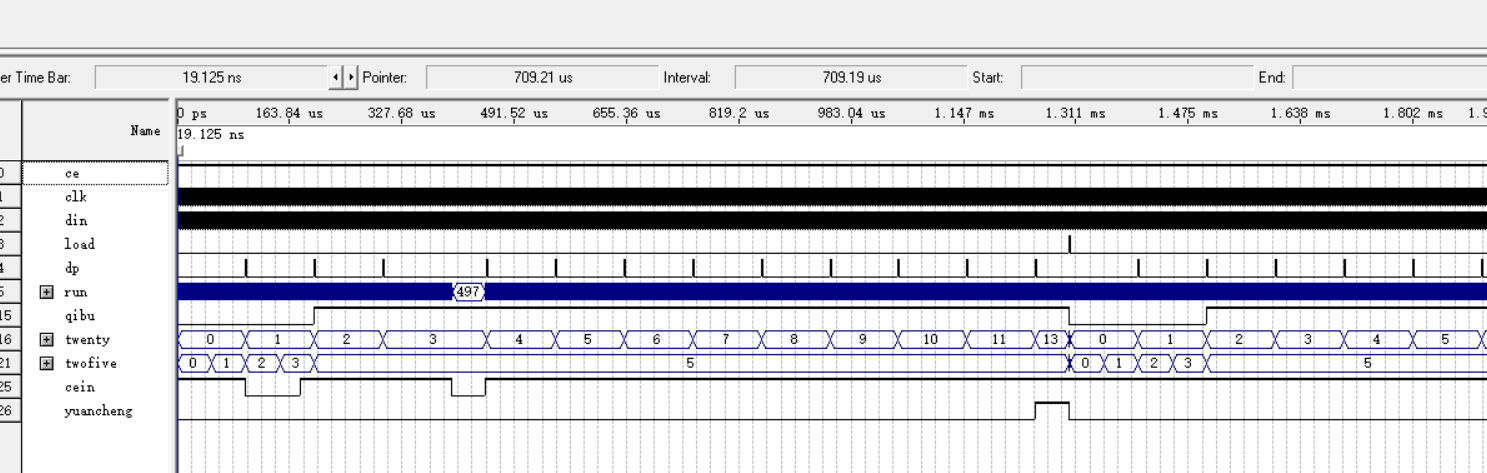
flag**<=**1**;**

**end** **if;**

**代码解说：**在起步信号load到来之后，所有的变量和输出全部置0，准备下一次的里程统计。

**3.6.2仿真输入信号的设计与分析**

**3.6.3 仿真结果时序图、简要分析**



图xx licheng模块仿真结果

**3.7 计时计价脉冲生成模块**

**3.7.1关键VHDL源代码及解说**

**if** load**=**'0' **then**

**if** ce**=**'1' **then**

**if** clk**=**'1' **and** clk'**event** **then**

**if** cein**=**'0' **then**--计时计价

**if** time **<** 600 **then**

time**:=**time**+**1**;**

ceout**<=**'0'**;**

**else**

time**:=**0**;**

ten**:=**ten**+**1**;**

ceout**<=**'1'**;**

**end** **if;**

**if** ten **<** 10 **then**

tenout**<=**'0'**;**

**else**

tenout**<=**'1'**;**

ten**:=**11**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **if;**

**else**

time**:=**0**;**

ten**:=**0**;**

ceout**<=**'0'**;**

tenout**<=**'0'**;**

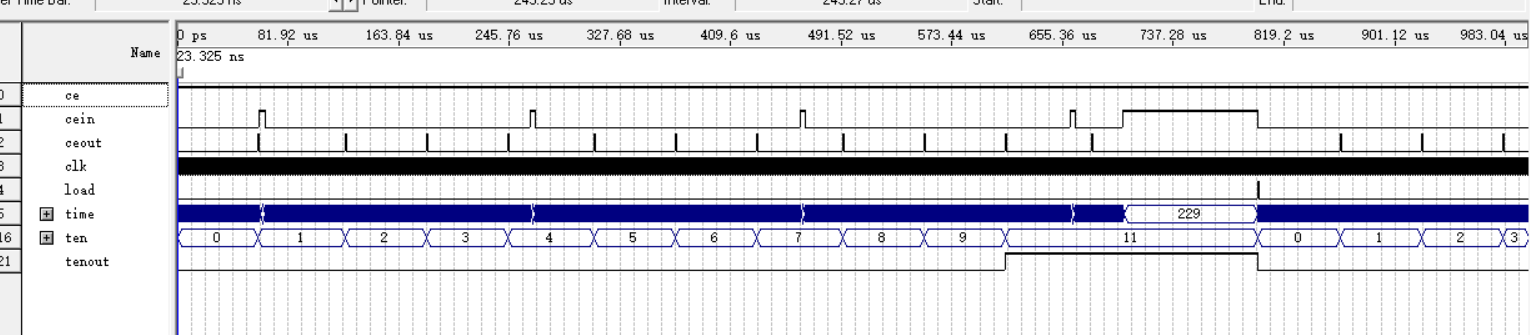
**end** **if;**

**代码解说：**-

**3.7.2仿真输入信号的设计与分析**

仿真时间1ms,为了节约时间，系统时钟信号clk周期设置为10KHz。cein输入几个高电平模拟出租车高速行驶，这个时候等待计时模块停止统计clk脉冲。load信号开始输入0，后来输入1，模拟复位操作。

**3.7.3 仿真结果时序图、简要分析**



图xx delay模块仿真结果

cein每输入一段高电平,变量times停止统计clk脉冲，暂时保留现在的值。当clk被统计达到600个，即等待时间达到1分钟，输出一个分钟脉冲ceout。ten变量统计输出的分钟脉冲，每输出一个分钟脉冲ceout，ten自加1，当ten达到10，输出tenout高电平，此时计价器低速行驶，开始收费。load输入复位信号脉冲后，所有变量置0，开始新一轮等待时间脉冲生成。

**3.8 起步价内计价模块**

**3.8.1关键VHDL源代码及解说**

**if** state\_pricing **=** 1 **and** load **=** '1' **then** --总价复位为0

total\_price **<=** 0**;**

**end** **if;**

**if** state\_pricing **=** 2 **then** --起步价内计价

**if** night **=** '0' **then** --白天起步价

total\_price **<=** day\_starting\_price**;**

**elsif** night **=** '1' **then** --夜间起步价

total\_price **<=** night\_starting\_price**;**

**end** **if;**

**end** **if;**

**if** state\_pricing **=** 3 **then**

total\_price **<=** 0**;**

**end** **if;**

**代码解说：**-

**3.8.2仿真输入信号的设计与分析**

**3.8.3 仿真结果时序图、简要分析**

**3.9 起步价外计价模块**

**3.9.1关键VHDL源代码及解说**

**if** clk**=**'1' **and** clk'**event** **then**

night\_mileage\_price **:=** day\_mileage\_price **+** 10**;**

--夜间计时单价

night\_timing\_price **:=** day\_timing\_price **+** 10**;**

--夜间长途单价

night\_long\_distance\_price **:=** day\_long\_distance\_price **+** 10**;**

**if** state\_pricing **=** 1 **and** load **=** '1' **then** --总价复位为0

total\_price **<=** 0**;**

mileage\_price\_sum **<=** 0 **;**

timing\_price\_sum **<=**0 **;**

**end** **if;**

**if** state\_pricing **=** 3 **then**

**if** night **=** '1' **then**

**if** dp **=** '1' **then**

**if** yuancheng **=** '1' **then**

mileage\_price\_sum **<=** mileage\_price\_sum **+** night\_long\_distance\_price **;** --夜间远程计价

**elsif** yuancheng **=** '0' **then**

mileage\_price\_sum **<=** mileage\_price\_sum **+** night\_mileage\_price **;** --夜间普通计价

**end** **if;**

**end** **if;**

**if** minin **=** '1' **then**

timing\_price\_sum **<=** timing\_price\_sum **+** night\_timing\_price**;**--夜晚计时计价

**end** **if;**

total\_price **<=** mileage\_price\_sum **+** timing\_price\_sum **+** night\_starting\_price**;**

**elsif** night **=** '0' **then**

**if** dp **=** '1' **then**

**if** yuancheng**=**'1' **then**

mileage\_price\_sum **<=** mileage\_price\_sum **+** day\_long\_distance\_price **;**--白天远程

**elsif** yuancheng **=** '0' **then**

mileage\_price\_sum **<=** mileage\_price\_sum **+** day\_mileage\_price **;**--白天普通

**end** **if;**

**end** **if;**

**if** minin **=** '1' **then**

timing\_price\_sum **<=** timing\_price\_sum **+** day\_timing\_price**;** --白天计时计价

**end** **if;**

total\_price **<=** mileage\_price\_sum **+** timing\_price\_sum **+** day\_starting\_price**;**

**end** **if;**

**end** **if;**

**end** **if;**

**代码解说：**-

**3.9.2仿真输入信号的设计与分析**

**3.9.3 仿真结果时序图、简要分析**

**3.10 总里程脉冲生成统计模块**

**3.10.1关键VHDL源代码及解说**

**A、总里程脉冲生成模块**

**if** load**=**'0' **then**

--当ce为1时，执行下面程序

**if** ce**=**'1' **then**

--当clk为上升沿时执行下面程序

**if** clk'**event** **and** clk**=**'1' **then**

**if** din**=**'1' **then**

--不足500m

**if** run **<** 250 **then**

run**:=**run**+**1**;**

dp\_total**<=**'0'**;**

**else**

--正好500m

**if** run**=**250 **then**

twofive**:=**twofive**+**1**;**

**end** **if;**

**if** run**<**500 **then**

run**:=**run**+**1**;**

dp\_total**<=**'0'**;**

**else**

twofive**:=**twofive**+**1**;**

dp\_total**<=**'1'**;**

twenty**:=**twenty**+**1**;**

run**:=**"0000000000000"**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **if;**

**else**

run**:=**"0000000000000"**;**

twofive**:=(others=>**'0'**);**

twenty**:=(others=>**'0'**);**

dp\_total**<=**'0'**;**

**end** **if;**

**代码解说：**-

**B、总里程脉冲统计模块**

**if** load**=**'0' **then**

**if** ce**=**'1' **then**

**if** din**=**'1' **and** din'**event** **then**

comb**:=**comb**+**1**;**

total**<=**comb**;**

**end** **if;**

**end** **if;**

**else**

comb**:=**0**;**

total**<=**0**;**

**end** **if;**

**代码解说：**-

**3.10.2仿真输入信号的设计与分析**

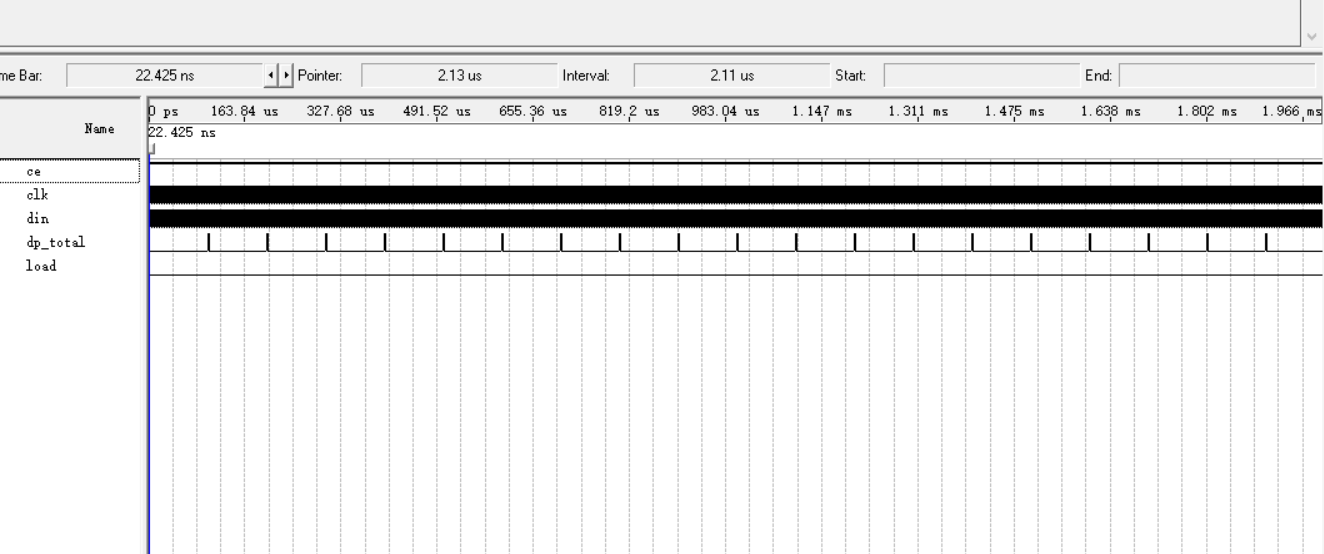
**A、总里程脉冲生成模块**

设置时钟信号clk，din,观察行驶总里程脉冲输出。

**B、总里程脉冲统计模块**

**3.10.3 仿真结果时序图、简要分析**

**A、总里程脉冲生成模块**



图xx 总里程脉冲生成模块仿真图

每行驶一公里，便输出一个里程脉冲，仿真结果正确。

**B、总里程脉冲统计模块**

**3.11 总计时脉冲生成统计模块**

**3.11.1关键VHDL源代码及解说**

**A、总计时脉冲生成模块**

**if** load**=**'0' **then**

**if** ce**=**'1' **then**

**if** clk**=**'1' **and** clk'**event** **then**

**if** cein**=**'0' **then**--计时计价

**if** time **<** 600 **then**

time**:=**time**+**1**;**

ceout**<=**'0'**;**

**else**

time**:=**0**;**

ten**:=**ten**+**1**;**

ceout**<=**'1'**;**

**end** **if;**

**if** ten **<** 10 **then**

tenout**<=**'0'**;**

**else**

tenout**<=**'1'**;**

ten**:=**11**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **if;**

**else**

time**:=**0**;**

ten**:=**0**;**

ceout**<=**'0'**;**

tenout**<=**'0'**;**

**end** **if;**

**代码解说：**-

**B、总计时脉冲统计模块**

**if** load**=**'0' **then**

**if** ce**=**'1' **then**

**if** din**=**'1' **and** din'**event** **then**

comb**:=**comb**+**1**;**

total**<=**comb**;**

**end** **if;**

**end** **if;**

**else**

comb**:=**0**;**

total**<=**0**;**

**end** **if;**

**代码解说：**-

**3.11.2仿真输入信号的设计与分析**

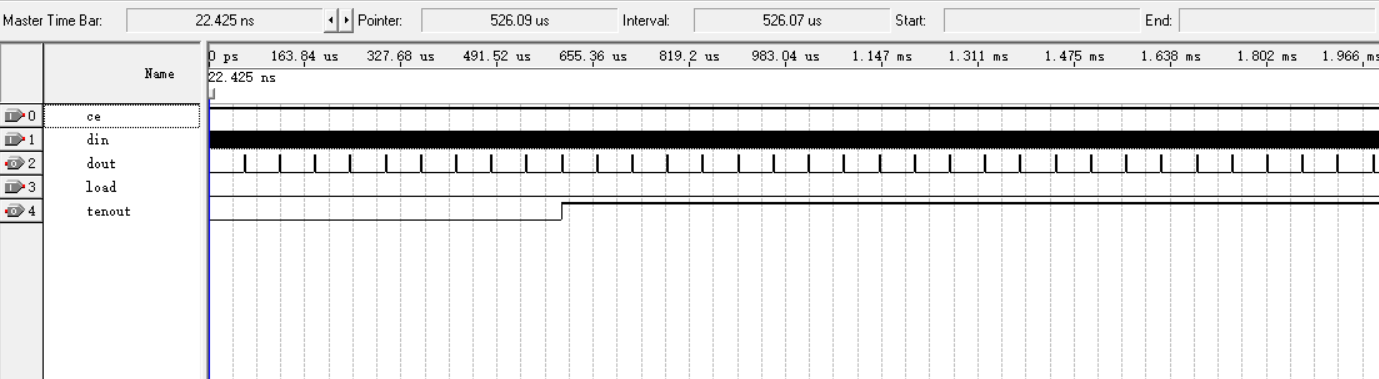
**A、总计时脉冲生成模块**

设置时间脉冲din，统计行驶分钟脉冲dout；

**B、总计时脉冲统计模块**

**3.11.3 仿真结果时序图、简要分析**

**A、总计时脉冲生成模块**



图xx 总计时脉冲生成模块

行驶一分钟输出一个分钟脉冲，输出给脉冲统计模块用于显示出租车的总的行驶时间。仿真结果正确。

**B、总计时脉冲统计模块**

**3.12 分位模块**

价格和里程因为都是十进制，所以可以共用一个分位模块，便于数码管显示；时间因为有第二个小时位，第二个分钟位最大只能是5，所以需要单独的一个时间分位模块。

**3.12.1关键VHDL源代码及解说**

**A、价格和里程分位模块**

**关键VHDL源代码：**

**if** clk'**event** **and** clk**=**'1' **then**

**if** comb **<** total **then**

**if(**comba**=**9 **and** combb**=**9 **and** combc**=**9**)then**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**"0000"**;**

combd**:=**combd**+**1**;**

comb**:=**comb**+**1**;**

**elsif(**comba**=**9 **and** combb**=**9**)then**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**combc**+**1**;**

comb**:=**comb**+**1**;**

**elsif(**comba**=**9 **and** combb**=**9 **and** combc**=**9 **and** combd**=**9**)** **then**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**"0000"**;**

combd**:=**"0000"**;**

combe**:=**combe**+**1**;**

comb**:=**comb**+**1**;**

**elsif(**comba**=**9**)** **then**

comba**:=**"0000"**;**

combb**:=**combb**+**1**;**

comb**:=**comb**+**1**;**

**else**

comba**:=**comba**+**1**;**

comb**:=**comb**+**1**;**

**end** **if;**

**else**

shi**<=**combb**;**

ge**<=**comba**;**

bai**<=**combc**;**

qian**<=**combd**;**

wan**<=**combe**;**

comb**:=**total**;**

**end** **if;**

**end** **if;**

**代码解说：**此处clk上升沿检测（此处clk为系统时钟10Hz的脉冲），变量comb从0开始自加1，在满足comb < total的情况下，低位都是9的情况下，就会像最高位进1，当comb =total的时候，把变量comba从ge输出，将变量combb从shi输出，将combc从bai输出，将变量combd从qian输出，将combe从wan输出。

**关键VHDL源代码：**

**if** load**=**'0'**then**

**……**

**else**

--数值清零

comb**:=**0**;**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**"0000"**;**

combd**:=**"0000"**;**

combe**:=**"0000"**;**

shi**<=**"0000"**;**

ge**<=**"0000"**;**

bai**<=**"0000"**;**

qian**<=**"0000"**;**

wan**<=**"0000"**;**

**end** **if;**

**代码解说：**load=0的时候进行输入路程或者价格分位的操作，当load=1的时候，进行复位，将输出和变量全部置0。

**B、时间分位模块**

**关键VHDL源代码：**

**if** clk'**event** **and** clk**=**'1' **then**

**if** comb **<** totaltime **then**

**if(**comba**=**9 **and** combb**=**5 **and** combc**=**9**)then**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**"0000"**;**

combd**:=**combd**+**1**;**

comb**:=**comb**+**1**;**

**elsif(**comba**=**9 **and** combb**=**5**)then**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**combc**+**1**;**

comb**:=**comb**+**1**;**

**elsif(**comba**=**9 **and** combb**=**5 **and** combc**=**9 **and** combd**=**5**)** **then**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**"0000"**;**

combd**:=**"0000"**;**

comb**:=**comb**;**

**elsif(**comba**=**9**)** **then**

comba**:=**"0000"**;**

combb**:=**combb**+**1**;**

comb**:=**comb**+**1**;**

**else**

comba**:=**comba**+**1**;**

comb**:=**comb**+**1**;**

**end** **if;**

**else**

sfen**<=**combb**;**

ffen**<=**comba**;**

fh**<=**combc**;**

sh**<=**combd**;**

comb**:=**totaltime**;**

**end** **if;**

**end** **if;**

**代码解说：**此处clk上升沿检测（此处clk为系统时钟10Hz的脉冲），变量comb从0开始自加1，在满足comb < total的情况下，当第一分钟位数值满9时，向第二分钟位进1；当第一分钟位数值满9时，当第二分钟位数值满5时，就会向第一个小时位进1，当下面所有的值满了，就向最高位进1，当comb =total的时候，把变量comba从ffen输出，将变量combb从sfen输出，将combc从fh输出，将变量combd从sh输出,用于数码管显示。

**关键VHDL源代码：**

**if** load**=**'0'**then**

**……**

**else** --数值清零

comb**:=**0**;**

comba**:=**"0000"**;**

combb**:=**"0000"**;**

combc**:=**"0000"**;**

combd**:=**"0000"**;**

sfen**<=**"0000"**;**

ffen**<=**"0000"**;**

sh**<=**"0000"**;**

**end** **if;**

**代码解说：**load=0的时候进行输入时间分位的操作，当load=1的时候，进行复位，将输出和变量全部置0。

**3.12.2 仿真输入信号的设计与分析**

**A、价格和里程分位模块**

仿真为了节约时间，clk时钟脉冲设置为10KHz，ce使能信号一直设为1，load复位信号前期置0，后期置1，模拟启动和复位的操作。total设置随机逐渐递增的价格，模拟出租车行驶途中的计价。

**B、时间分位模块**

仿真为了节约时间，clk时钟脉冲设置为10KHz，ce使能信号一直设为1，load复位信号后期输出一个复位脉冲，模拟复位操作。totaltime输入时间要分离的时间。

**3.12.3 仿真结果时序图、简要分析**

**A、价格和里程分位模块**

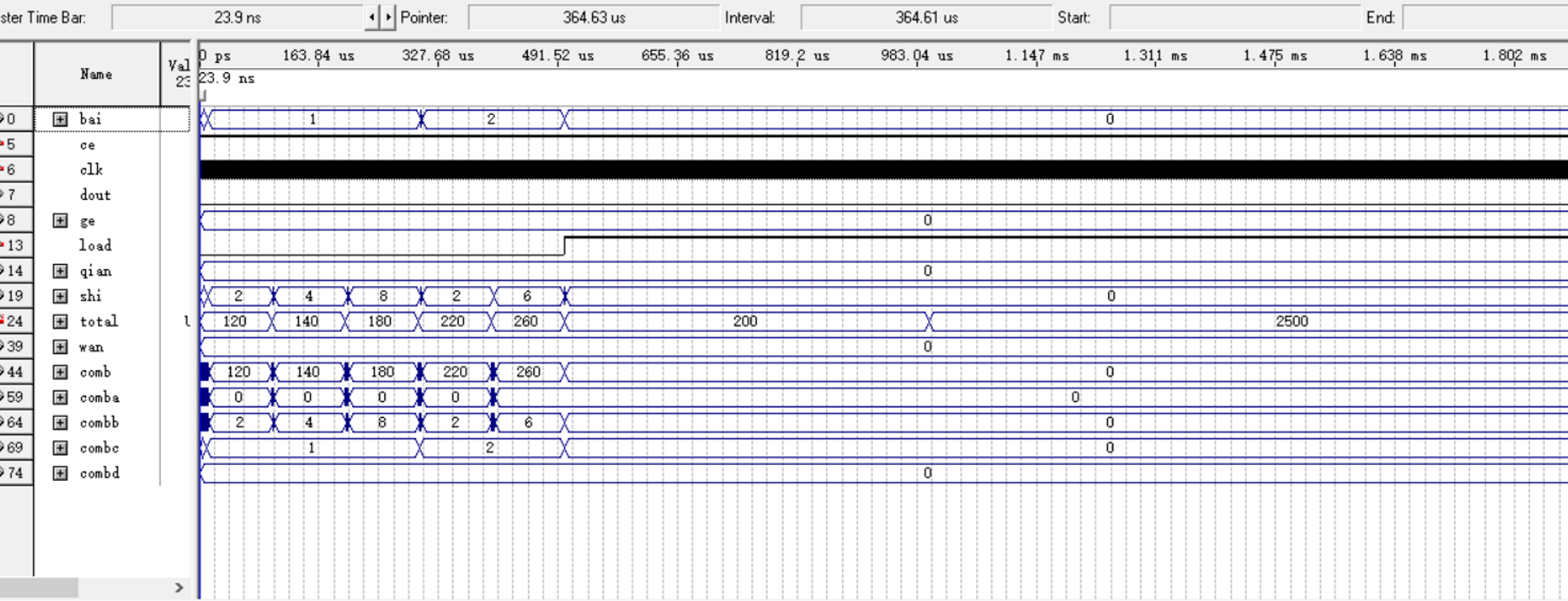
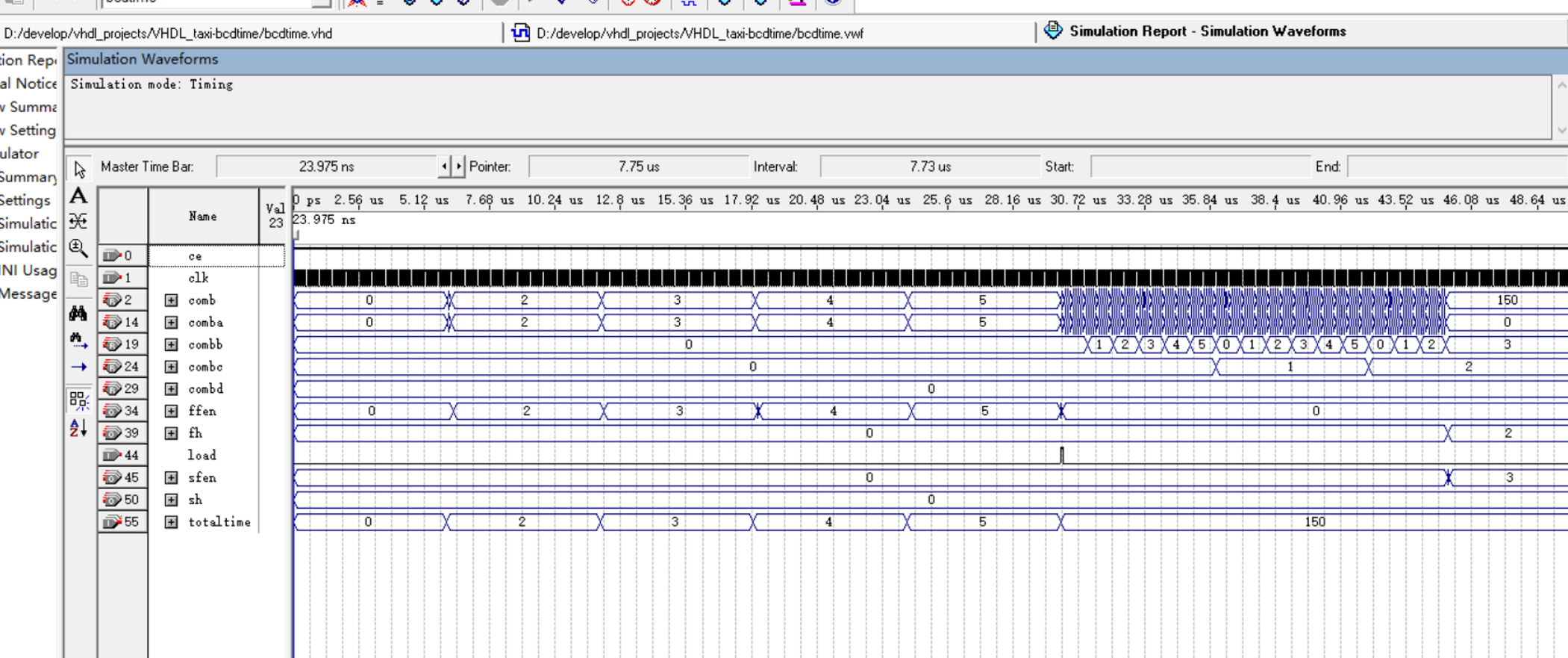


图 xx 价格和里程分位模块仿真结果

输入的total即是出租车行驶的价格，出租车价格被分位为ge,shi,bai,qian输出。如total输入120，bai输出1，shi输出2，ge输出0。在load复位电平输入后，变量和输出都被置0。该模块仿真结果达到预期。

**B、时间分位模块**



图xx 时间分位模块仿真结果

输入的totaltime即是出租车行驶的总时间，出租车价格按照时间显示格式被分位为ffen,sfen,fh,sh输出。如totaltime输入150，fh输出2，sfen输出3，ffen输出0。在load复位脉冲输入后，变量和输出都被置0。该模块仿真结果达到预期。

**3.13 选择模块**

**3.13.1关键VHDL源代码及解说**

**A、价格选择模块**

**if** ce**=**'1' **then**

**if** total1**>**0 **and** total2**=**0 **and** pricesel**=**0 **then**

total**<=**total1**;**

**elsif** total1**=**0 **and** total2**>**0 **and** pricesel**=**0 **then**

total**<=**total2**;**

**elsif** pricesel**>**0 **then**

total**<=**price**;**

**end** **if;**

**end** **if;**

**代码解说：**pricesel输入为1的时候，total输出价格设置模式输入的价格；pricesel输入为0的时候，total输出正常模式输入的价格；

**B、时间选择模块**

**if** ce**=**'1' **then**

vffentemp**<=**vffen**;**

vsfentemp**<=**vsfen**;**

vfhtemp**<=**vfh**;**

vshtemp**<=**vsh**;**

**if** timesel**=**'1' **then** --设置状态

rffentemp**<=**conv\_std\_logic\_vector**(**vffentemp**,**4**);**

rsfentemp**<=**conv\_std\_logic\_vector**(**vsfentemp**,**4**);**

rfhtemp**<=**conv\_std\_logic\_vector**(**vfhtemp**,**4**);**

rshtemp**<=**conv\_std\_logic\_vector**(**vshtemp**,**4**);**

rffen**<=**rffentemp**;**

rsfen**<=**rsfentemp**;**

rfh**<=**rfhtemp**;**

rsh**<=**rshtemp**;**

**else**

rffen**<=**ffen**;**

rsfen**<=**sfen**;**

rfh**<=**fh**;**

rsh**<=**sh**;**

**end** **if;**

**end** **if;**

**代码解说：**timesel输入为1的时候，total输出时间设置模式输入的时间；timesel输入为0的时候，total输出正常模式输入的时间；

**C、设里程选择模块**

**if** ce**=**'1' **then**

**if** pricesel**=**1 **then**--1 是显示白天起步价

price**<=**day\_starting\_price**;**

**elsif** pricesel**=**2 **then**--2 是夜晚起步价

price**<=**night\_starting\_price**;**

**elsif** pricesel**=**3 **then**--3 是白天里程

price**<=**day\_mileage\_price**;**

**elsif** pricesel**=**4 **then**--4 是 白天计时

price**<=**day\_timing\_price**;**

**elsif** pricesel**=**5 **then**--5 是白天长途

price**<=**day\_long\_distance\_price**;**

**end** **if;**

**end** **if;**

**代码解说：**根据pricesel的状态选择输出不同的价格到用于数码管显示。

**3.13.2 仿真输入信号的设计与分析**

**A、价格选择模块**

设置pricesel5种状态，给每种价格设置一个输入值，观察最后的输出值。

**B、时间选择模块**

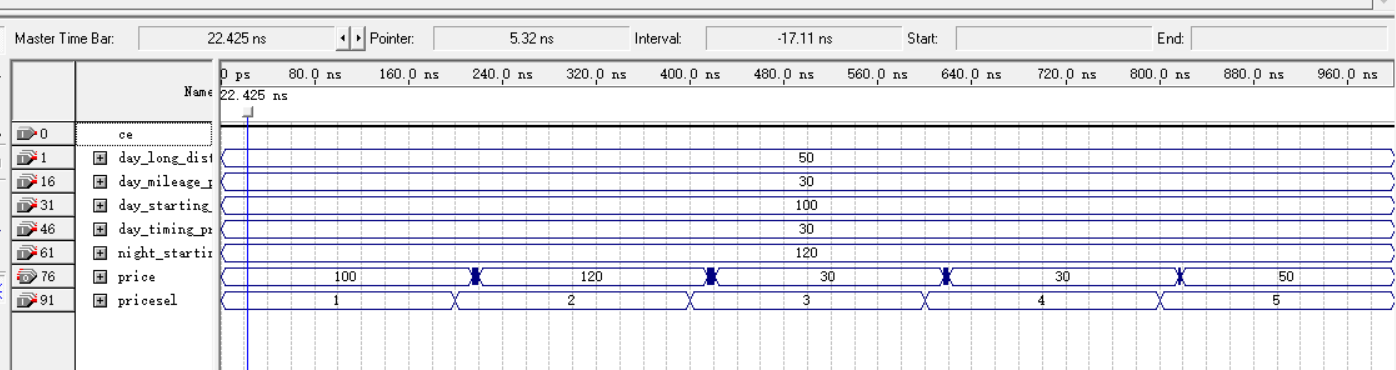
前半段时间设置timesel为1，即进行时间设置模式，后半段时间设置timesel为0，即正常的显示模式。其余输入设置一下具体的数字。

**C、计价结果选择模块**

前半段时间设置pricesel为1，即进行时间设置模式，后半段时间设置pricesel为0，即正常的显示模式。其余输入设置一下具体的数字。

**3.13.3 仿真结果时序图、简要分析**

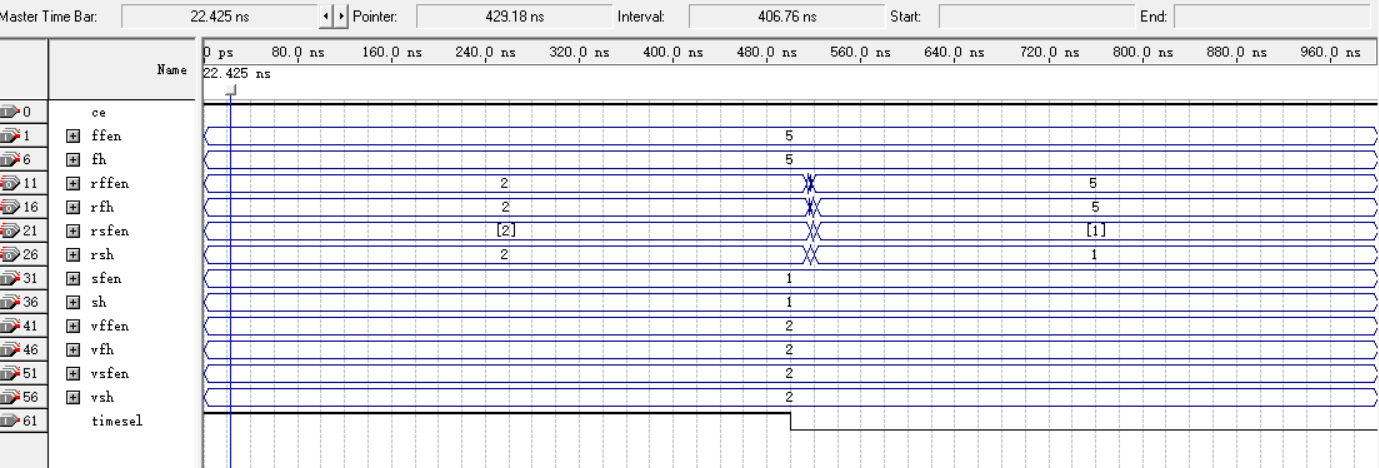
**A、价格选择模块**



当pricesel为1，price输出白天起步价；当pricesel为2时，price输出夜间记步价；当pricesel为3时，price输出白天里程价；当pricesel为4时，price输出白天计时价；当pricesel为5时，price输出白天远程里程价；仿真结果正确。

图xx 价格显示模块显示

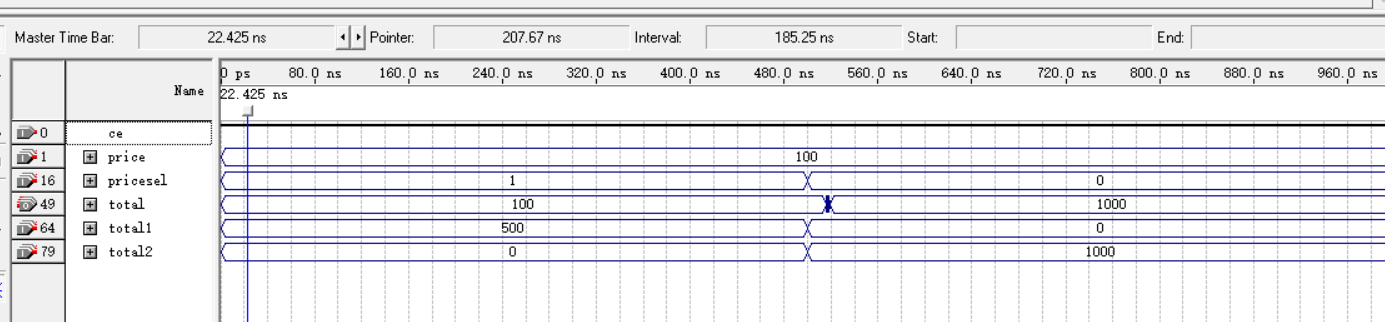
**B、时间选择模块**



图xx 时间显示模块显示

当timesel为1时，输出设置时间模式输入的时间；当timesel为0时，输出正常模式输入的时间；模块仿真结果正确。

**C、计价结果选择模块**



图xx 里程选择模块仿真

当pricesel为1时，输出设置价格模式输入的价格；当pricesel为0时，输出正常模式下，输入的价格，图中total2有输入，total就输出total2的值；模块仿真结果正确。

**3.14 显示模块**

**3.14.1关键VHDL源代码及解说**

**A、价格显示模块**

**关键VHDL源代码：**

**if** **rising\_edge(**clk**)** **then**

**case** sell **is**

**when** "000" **=>**

sell**:=**"001"**;**

temp**:=**a**;**

**when** "001" **=>**

sell**:=**"010"**;**

temp**:=**b**;**

--里程和计价小数点

li**:=**1**;**

**when** "010" **=>**

sell**:=**"100"**;**

temp**:=**c**;**

**when** "100" **=>**

sell**:=**"000"**;**

temp**:=**d**;**

**when** **others** **=>**

**null;**

**end** **case;**

**代码解说：**此处clk上升沿检测（此处clk为系统时钟10Hz的脉冲），case语句目的是循环显示不同位的数码管。用于价格显示的特殊性，第二个数字显示元需要附带小数点，其余价格的位置都是正常显示，将li作为标志位。

**B、时间显示模块**

**关键VHDL源代码：**

**if** **rising\_edge(**clk**)** **then**

**case** sell **is**

**when** "000" **=>**

sell**:=**"001"**;**

temp**:=**ffen**;**--00:0x

**when** "001" **=>**

sell**:=**"010"**;**--00:x0倒着显示

temp**:=**sfen**;**

sfenflag**:=**'1'**;**

**when** "010" **=>**

sell**:=**"100"**;**--0x:00

temp**:=**fh**;**

fhflag**:=**'1'**;**

**when** "100" **=>**

sell**:=**"000"**;**

temp**:=**sh**;**--x0:00

**when** **others** **=>**

**null;**

**end** **case;**

**代码解说：**此处clk上升沿检测（此处clk为系统时钟10Hz的脉冲），case语句目的是循环显示不同位的数码管。时间显示的特殊性，第二个数字显示第一个小时需要附带小数点，第三个数字显示第二个分钟数值需要附带倒着的小数点。分别使用fhflag和sfenflag作为标志。

**关键VHDL源代码：**

**if** fhflag**=**'0' **and** sfenflag**=**'0' **then**--正常显示不带小数点

**case** temp **is**--段选共阴,不包括小数点

**when**"0000"**=>**outp**<=**"00111111"**;**--0

**when**"0001"**=>**outp**<=**"00000110"**;**--1

**when**"0010"**=>**outp**<=**"01011011"**;**--2

**when**"0011"**=>**outp**<=**"01001111"**;**--3

**when**"0100"**=>**outp**<=**"01100110"**;**--4

**when**"0101"**=>**outp**<=**"01101101"**;**--5

**when**"0110"**=>**outp**<=**"01111101"**;**--6

**when**"0111"**=>**outp**<=**"00000111"**;**--7

**when**"1000"**=>**outp**<=**"01111111"**;**--8

**when**"1001"**=>**outp**<=**"01101111"**;**--9

**when** **others=>**outp**<=null;**

**end** **case;**

sel**<=**sell**;**

**代码解说：**不带小数点的数码管段选编码**。**temp中是待显示的数值。根据temp中的值输出对应的数码管的编码。

**关键VHDL源代码：**

**elsif** fhflag**=**'1' **then** --正常显示带着小数点

fhflag**:=**'0'**;**

**case** temp **is**

**when**"0000"**=>**outp**<=**"10111111"**;**--0

**when**"0001"**=>**outp**<=**"10000110"**;**--1

**when**"0010"**=>**outp**<=**"11011011"**;**--2

**when**"0011"**=>**outp**<=**"11001111"**;**--3

**when**"0100"**=>**outp**<=**"11100110"**;**--4

**when**"0101"**=>**outp**<=**"11101101"**;**--5

**when**"0110"**=>**outp**<=**"11111101"**;**--6

**when**"0111"**=>**outp**<=**"10000111"**;**--7

**when**"1000"**=>**outp**<=**"11111111"**;**--8

**when**"1001"**=>**outp**<=**"11101111"**;**--9

**when** **others=>**outp**<=null;**

**end** **case;**

**代码解说：**在位选代码中赋值的标志位，作为条件语句的条件，作为时间数码管显示的特殊值，此处根据temp的值，在case语句中选中对应的带小数点显示的数码管编码。

**关键VHDL源代码：**

**elsif** sfenflag**=**'1' **then**

sfenflag**:=**'0'**;**--倒着显示带小数点

**case** temp **is**

**when**"0000"**=>**outp**<=**"10111111"**;**--0

**when**"0001"**=>**outp**<=**"10110000"**;**--1

**when**"0010"**=>**outp**<=**"11011011"**;**--2

**when**"0011"**=>**outp**<=**"11110011"**;**--3

**when**"0100"**=>**outp**<=**"11110100"**;**--4

**when**"0101"**=>**outp**<=**"11101101"**;**--5

**when**"0110"**=>**outp**<=**"11101111"**;**--6

**when**"0111"**=>**outp**<=**"10111000"**;**--7

**when**"1000"**=>**outp**<=**"11111111"**;**--8

**when**"1001"**=>**outp**<=**"11111101"**;**--9

**when** **others=>**outp**<=null;**

**end** **case;**

sel**<=**sell**;**

**end** **if;**

**代码解说：**在位选代码中赋值的标志位，作为条件语句的条件，作为时间数码管显示的特殊值，此处根据temp的值，在case语句中选中对应的带小数点倒着显示的数码管编码。

**C、里程显示模块**

**关键VHDL源代码：**

**if** **rising\_edge(**clk**)** **then**

**case** sell **is**

**when** "000" **=>**

sell**:=**"001"**;**

temp**:=**ge**;**

**when** "001" **=>**

sell**:=**"010"**;**

temp**:=**shi**;**

**when** "010" **=>**

sell**:=**"100"**;**

temp**:=**bai**;**

**when** "100" **=>**

sell**:=**"000"**;**

temp**:=**qian**;**

**when** **others** **=>**

**null;**

**end** **case;**

**代码解说：**在一个clk上升沿之后，不断循环显示数码管显示路程的位选。并将位选对应的输入选择出来赋值给变量temp。

**关键VHDL源代码：**

**case** temp **is**--段选共阴,不包括小数点

**when**"0000"**=>**outp**<=**"00111111"**;**--0

**when**"0001"**=>**outp**<=**"00000110"**;**--1

**when**"0010"**=>**outp**<=**"01011011"**;**--2

**when**"0011"**=>**outp**<=**"01001111"**;**--3

**when**"0100"**=>**outp**<=**"01100110"**;**--4

**when**"0101"**=>**outp**<=**"01101101"**;**--5

**when**"0110"**=>**outp**<=**"01111101"**;**--6

**when**"0111"**=>**outp**<=**"00000111"**;**--7

**when**"1000"**=>**outp**<=**"01111111"**;**--8

**when**"1001"**=>**outp**<=**"01101111"**;**--9

**when** **others=>**outp**<=null;**

**end** **case;**

**代码解说：**将temp的值转变为对应的共阴极数码管的段选显示代码，赋值给outp输出，路程的显示是不带小数点的输出。

**关键VHDL源代码：**

**if** load**=**'0'**then**

**......**

**else**

--显示默认内容

outp**<=**"00111111"**;**

sell**:=(others=>**'0'**);**

**end** **if;**

**代码解说：**在复位信号load到来之后，变量置0，数码管输出置0的，数码管的初始显示状态显示码。

**3.14.2 仿真输入信号的设计与分析**

**A、价格显示模块**

数码管价格显示模块的仿真的思路就是给数码管价格显示模块特定的价格输入后观察其数码管位选和段选输出。在仿真时间内，分别给a,b,c,d输入值，用于时间显示模块仿真。

**B、时间显示模块**

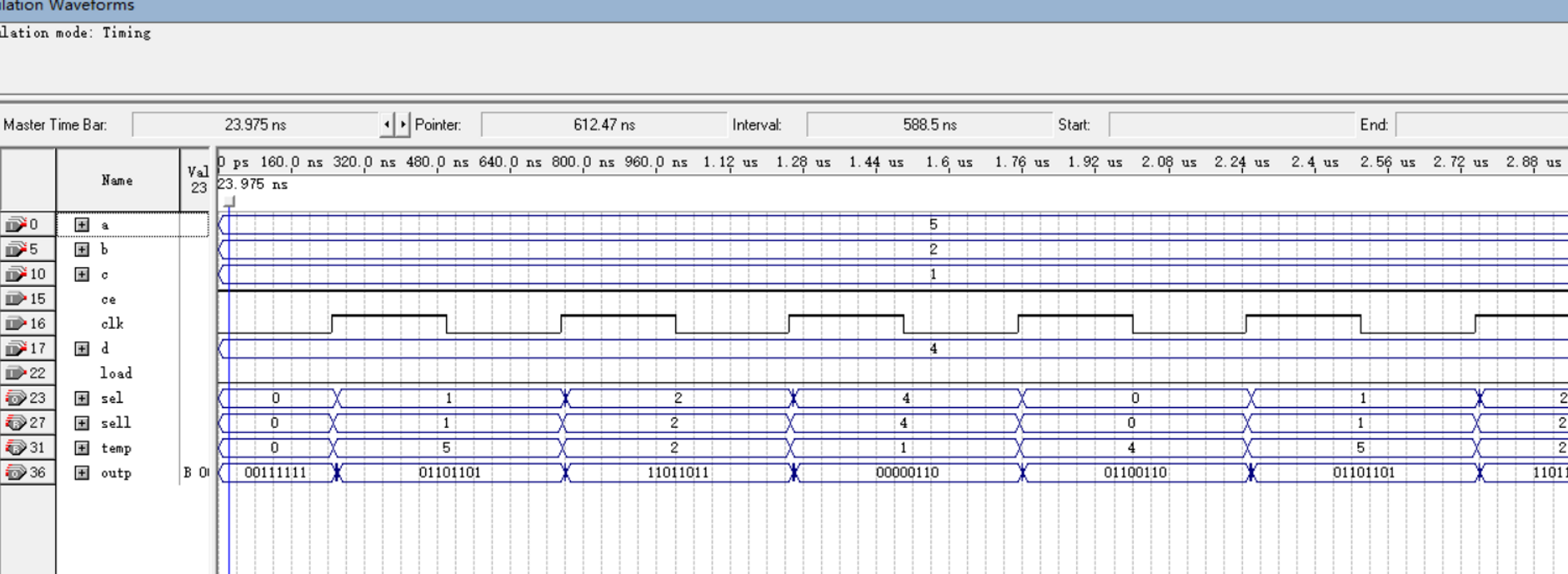
数码管时间显示模块的仿真的思路就是给数码管时间显示模块特定的时间输入后观察其数码管位选和段选输出。在仿真时间内，分别给ffen,sfen,sh,fh输入值，用于时间显示模块仿真。

**C、里程显示模块**

数码管里程显示模块的仿真的思路就是给数码管时间显示模块特定的时间输入后观察其数码管位选和段选输出。在仿真时间内，分别给ge,shi,bai,qian输入值，用于时间显示模块仿真。

**3.14.3 仿真结果时序图、简要分析**

**A、价格显示模块**

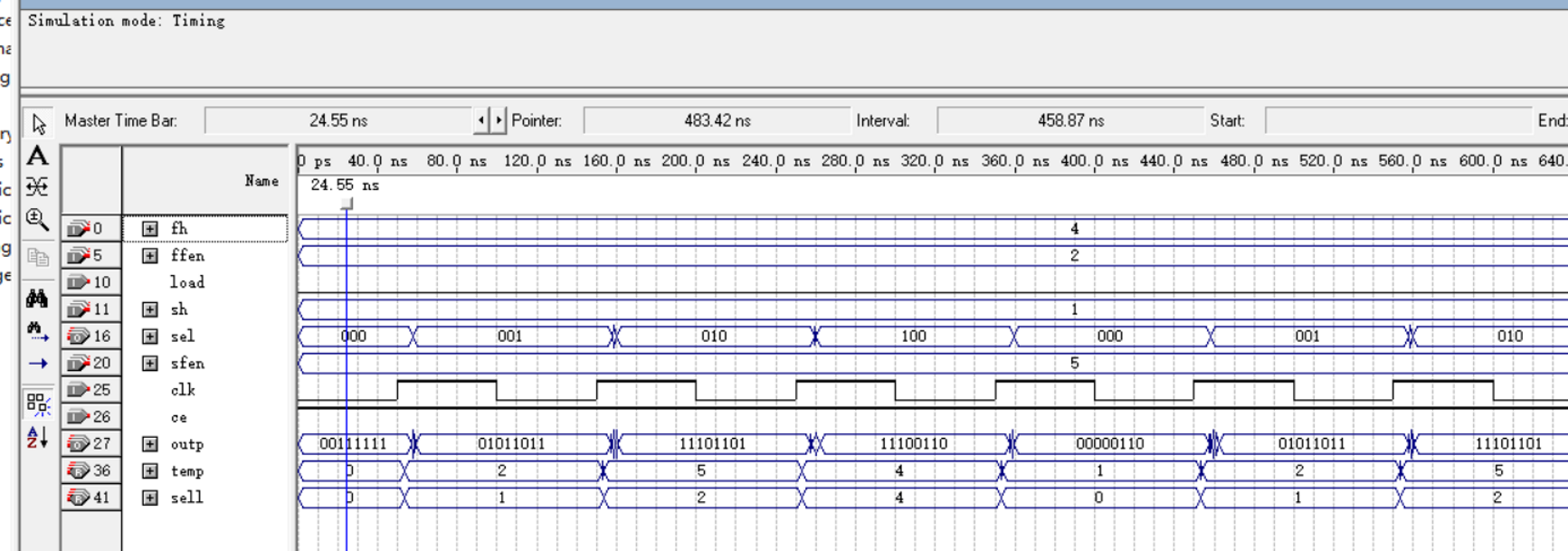


图xx 价格显示模块仿真结果

这里设置a,b,c,d分别为5、2、1、4,sel随着clk脉冲不停的循环，利用人眼视觉暂留效应，人眼可以看到稳定的数码管显示。outp依次输出5、2、1、4对应的数码管的编码。模块仿真结果正确。

**B、时间显示模块**

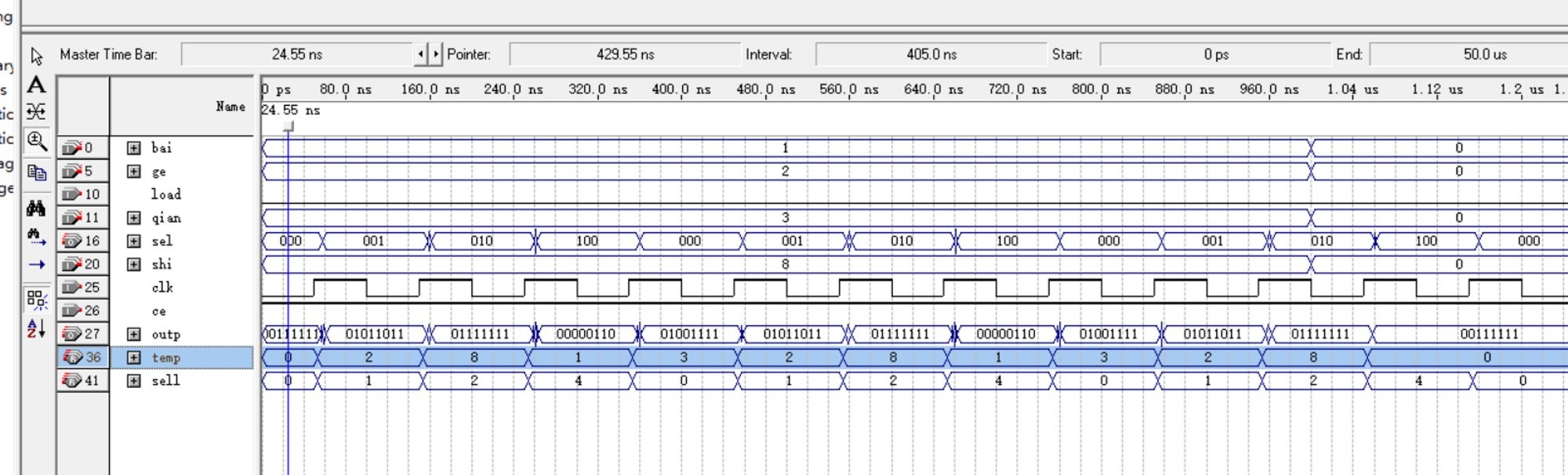
时间四个位置的数值输入



图xx 时间显示模块仿真结果

这里设置ffen,sfen,fh,sh分别为2、5、4、1,sel随着clk脉冲不停的循环，利用人眼视觉暂留效应，人眼可以看到稳定的数码管显示。outp依次输出2、5、4、1对应的数码管的编码。模块仿真结果正确。

**C、里程显示模块**



图xx 里程显示结果

这里设置ge,shi,bai,qian分别为2、8、1、3,sel随着clk脉冲不停的循环输出，利用人眼视觉暂留效应，人眼可以看到稳定的数码管显示。outp依次输出2、8、1、3对应的数码管的编码。模块仿真结果正确。

**3.15 蜂鸣器跳表提醒模块**

**3.15.1关键VHDL源代码及解说**

**关键VHDL源代码：**

**if** ce**=**'1' **then**

**if** qibu**=**'1' **then**

**if** run\_pulse**=**'1' **then**

bee\_pulse**<=**'1'**;**

**elsif** wait\_pulse**=**'1' **then**

bee\_pulse**<=**'1'**;**

**else**

bee\_pulse**<=**'0'**;**

**end** **if;**

**else**

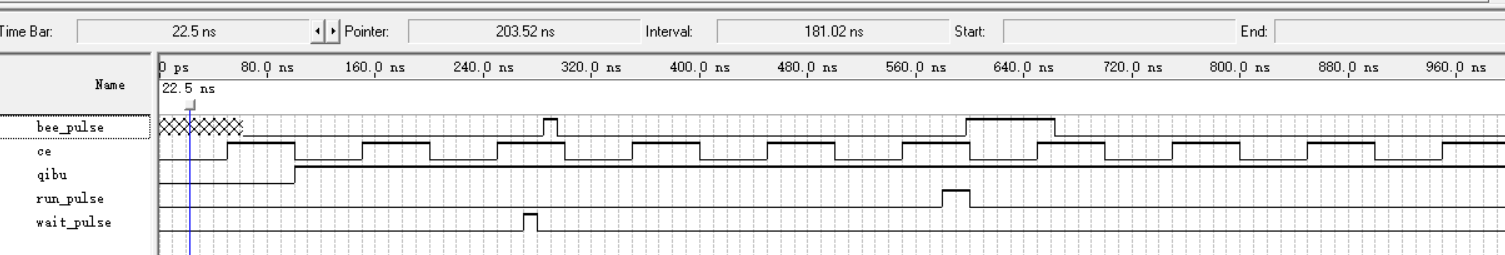
bee\_pulse**<=**'0'**;**

**end** **if;**

**end** **if;**

**代码解说:** 此处clk上升沿检测（此处clk为系统时钟10Hz的脉冲），分为两种情况，在起步价内，里程和等待时间跳表蜂鸣器不输出脉冲；在起步价外，即qibu=’1’，如果有里程脉冲和等待时间脉冲，这个时候会开始计价并且跳表，这个时候就开始输出蜂鸣器脉冲进行跳表提示。

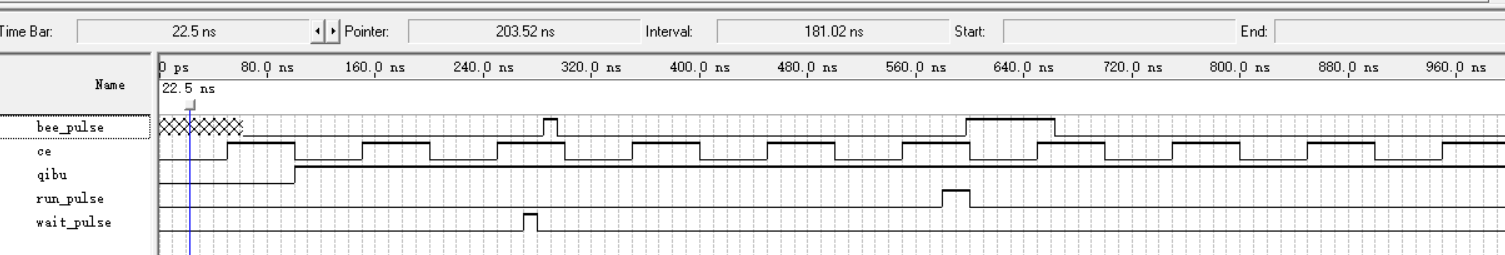
**3.14.2 仿真输入信号的设计与分析**



图xx 蜂鸣器跳表提醒模块仿真结果

应该输入里程脉冲和等待时间脉冲，还有起步信号的电平。这里直接利用整体的仿真结果来模拟蜂鸣器跳表提醒。

**3.14.3 仿真结果时序图、简要分析**



图xx 蜂鸣器跳表提醒模块仿真结果

由仿真结果可知，在起步价内计价(total\_price)和起步价外(total\_price)计价跳表的时候，蜂鸣器脉冲输出bee\_pulse都是有脉冲输出的，结果正确。

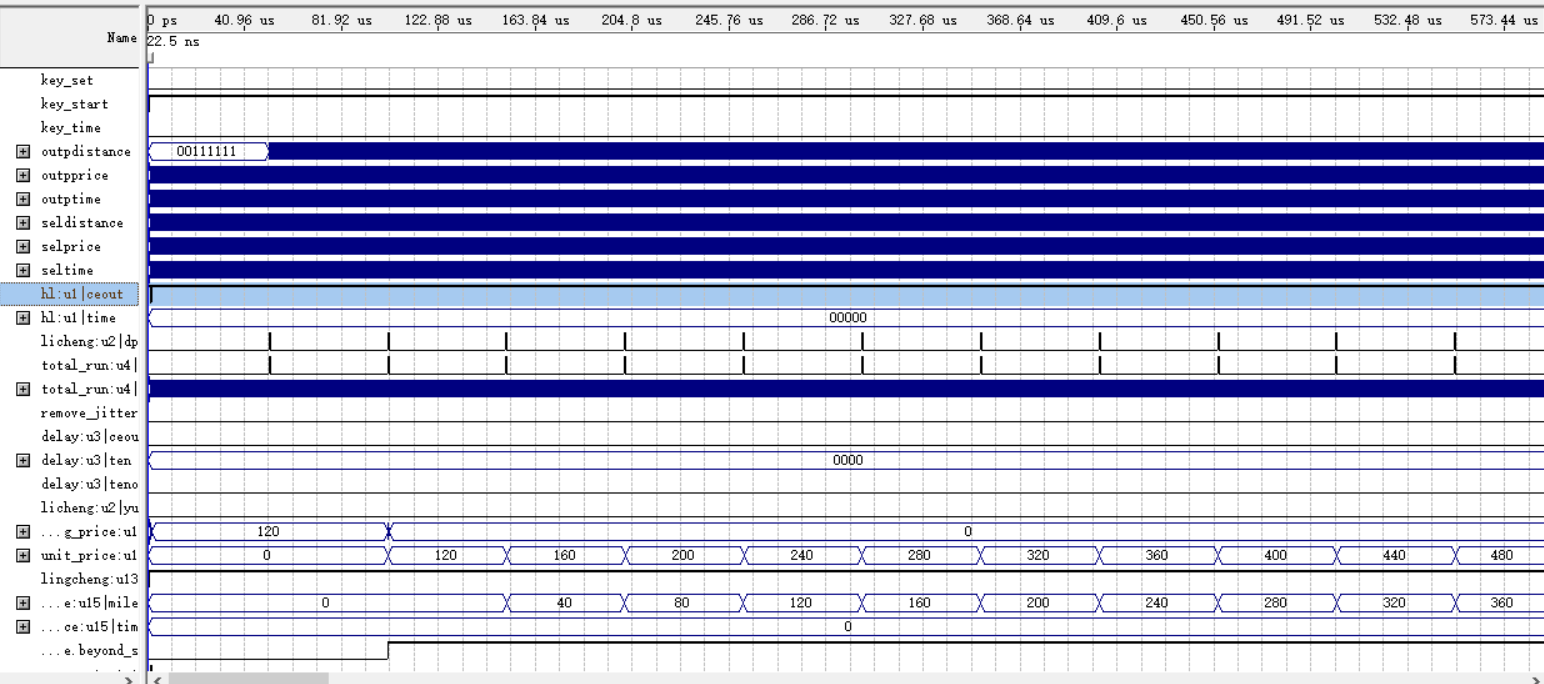
四、系统总体联调

4.1系统总体RTL图

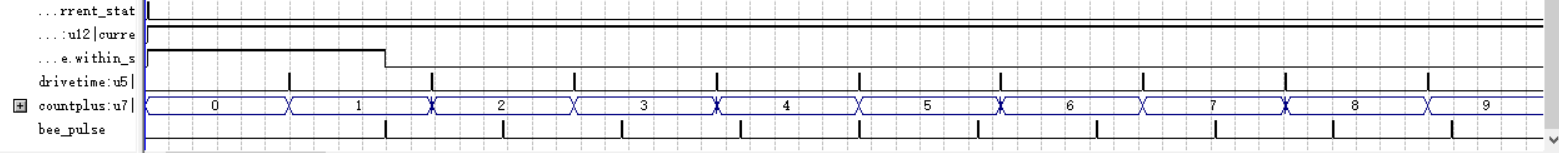


图xx 总系统的RTL图

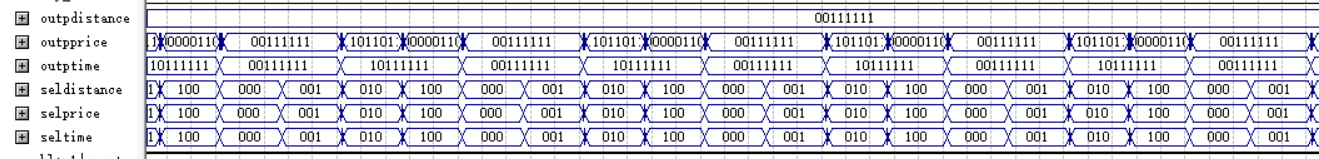
**4.2总体仿真结果**



图xx 整体仿真结果1



图xx 整体仿真结果2



图xx 用于数码管显示的位选和段选码输出

从仿真结果可以看到，出租车按下复位使能开始按键后，状态机先启动，后立即跳转到起步价计价状态，随着行驶里程脉冲输出2个之后，到达起步里程，立即跳到起步价外计价状态。licheng模块输出用于里程计价的里程脉冲，total\_run输出总的行驶里程；drivetime输出总的行驶时间，starting\_price变量输出起步价，最后在数码管上显示。unit\_price变量输出起步价外计价状态出租车计价。在每次价格变换时，即跳表的时候，蜂鸣器接收到脉冲bee\_pulse，蜂鸣器发出声音提醒乘客。