

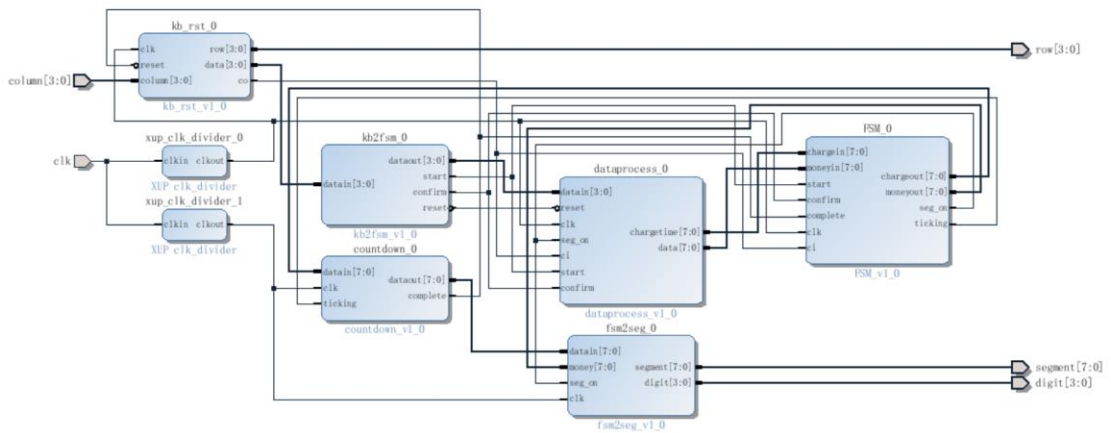
EDA 大作业二 投币式手机充电仪

一、实验目的

- 1、学习自顶向下、分模块的数字系统分析、设计与调试方法。
- 2、编写测试文件对设计电路进行仿真验证。
- 3、掌握规范使用硬件描述语言描述状态机电路的方法。

二、实验原理

1、电路的总体框图



如图所示，电路由以下几个模块组成。

①矩阵键盘检测电路 (kb_rst)

电子电路实验五已有论述，此处省略。

②键盘译码电路(kb2fsm)

用于将键盘的输出中的特殊功能（开始、清零、确认）与一般输出区分出来。

③充电时间计算电路(dataprocess)

用于在输入金额后计算最大投币金额与充电时长。

④状态机 (FSM)

用于在特定的条件下翻转电路的状态。

⑤倒计时电路 (countdown)

用于实现 10 秒倒计时。


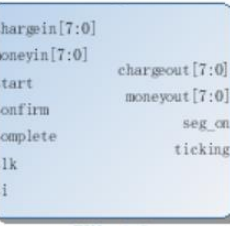
⑥数码管显示电路 (fsm2seg)

用于金额与充电时间的数码管译码与数码管选通。

⑦输入输出说明

输入	作用
column	键盘的列输入
clk	Basys3 实验板上自带的 100MHz 时钟信号
输出	作用
row	键盘的行输出
segment	数码管输出数字信号
digit	数码管选通信号

2、各模块的端口说明

模块	端口	作用
	clk	键盘扫描时钟信号，100Hz
	reset	用于完成计时后的整体电路清零
	column	键盘的列输入
	row	键盘的行输出
	data	被按下的键的输出
	co	描述是否有键被按下
	datain	来自键盘的输入信号
	dataout	经过初步译码的输出数据
	start	描述开始键是否被按下
	confirm	描述确认键是否被按下
	reset	描述清零键是否被按下
	datain	经过初步译码的纯一位数值输入
	reset	清零输入端
	clk	前级电路输出检测频率，100Hz
	seg_on	用于监视数码管是否点亮，不点亮则输出为 0
	ci	用于判断前级电路是否输出新数据
	start	用于判断若被按下的键是开始，则忽略本次输入数据
	confirm	用于判断若被按下的键是确认，则忽略本次输入数据
	chargeout	输出充电时间
	data	输出投币金额
	chargein	充电时间输入
	moneyin	投币金额输入
	start	检测开始是否被按下
	confirm	检测确认是否被按下
	complete	检测倒计时是否结束
	clk	状态机工作频率，100Hz
	ci	用于辅助判断是否开始被按下，而非键盘输出数据为开始，而实际数据来自于上一次按键
	chargeout	充电时间输出
	moneyout	投币金额输出
	seg_on	用于控制数码管是否被点亮
	ticking	用于控制倒计时的开始与结束

(续表)

	datain	充电时间输入
	clk	输入监测工作频率，500Hz
	ticking	控制倒计时开始或停止的输入信号
	dataout	充电时间输出（含倒计时）
	complete	充电结束后输出充电完成信号
	datain	充电时间输入（含倒计时）
	money	充电金额输入
	seg_on	用于控制是否点亮数码管
	clk	数码管选通扫描频率，500Hz
	segment	数码管输出信号
	digit	数码管选通信号

3、电路的状态转换图设计

依据实验任务，将电路分解为三个状态：初始状态，输入状态和充电状态。

其中上电后为初始状态，按下开始键后为输入状态，投币金额为 0 持续 10 秒后回到初始状态。输入状态下按下确认键进入充电状态，充电状态下若充电结束返回输入状态。

初始状态	开始→	输入状态	确认→	充电状态
	←无输入 10 秒		←充电结束	

4、电路设计思路

依据实验任务，可把此电路需要实现的功能分解为键盘输入、键盘输入译码与数据处理、倒计时、数码管译码四个部分，由于将电路划分为 3 个状态，还需要一个有限状态机模块。

键盘输入模块套用了综合设计实验的键盘代码，键盘输入译码由一个组合逻辑电路与时序逻辑电路构成。其中组合逻辑电路用于区分键盘的数值输出与控制信号输出，时序逻辑电路用于在键按下时捕捉数据并实现移位寄存投币金额，对超出上限的金额进行修正，并且同步计算出充电时间。

状态机由三个状态初始状态、输入状态、充电状态组成，上电后进入初始状态，初始状态下检测到开始键被按下进入输入状态，输入状态在无输入金额下开始计时，满 10 秒回到初始状态。输入状态下检测到确认键被按下进入充电状态，充电状态下检测到充电结束后返回输入状态。

倒计时电路在收到状态机发出的充电时间与充电开始信号后即开始倒计时并同步输出时间，倒计时结束时向前级电路返回充电结束信号。数码管译码电路用于将输入的金额信号与时间信号输出到数码管并实现数码管同步选通，而其输出与否受到状态机的直接控制。

但在完成电路架构的基本设计后，我们发现该逻辑中其实存在三个逻辑漏洞，这些逻辑漏洞都是因为直接套用键盘代码带来的。矩阵键盘在一个键被按下后用输出 co 表示有键被按下，但输出数据将恒定为对应的键位。若希望用键盘的清零端口来实现，则键盘会直接停止扫描、保持数据输出并将 co 置为 0。并且由于清零键在键盘上，所以若我们用键盘上的清零键实现对键盘的清零，一旦按下清零键，键盘将直接反复对自己清零，导致电路进入无法接受输入的死循环。这就意味着我们需要引入其他控制手段来弥补键盘代码带来的缺陷。

而三个逻辑漏洞分别为：一是在按下开始后电路进入输入状态，进入初始状态 10s 后电路返回初始状态。在这个过程中若没有任何其他键被按下，则键盘的数据输出将恒定为 1，导致状态将马上翻转到输入状态，无法停留在初始状态。解决办法为将状态机的翻转条件改为收到开始输入的同时检测到键盘上有键被按下。

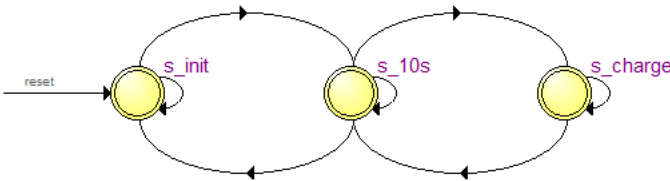
二是在电路处于初始状态下，键盘的输出可以进入到移位寄存器中并被存储。若在上电之前按下键盘上的数据键，上电后数码管显示的不是 0000 而是被按下的数据键。解决方法为在数据处理模块中引入状态机输出的控制数码管点亮的信号为选通信号，若检测到数码管不亮，则不接收来自键盘的输入数据。

第三个逻辑缺陷是最严重也是最根本的，即在充电结束后，由于我们依然没有实现键盘的清零，导致上一次充电的确认键输出还残留在电路之中，状态机便不会开始 10 秒倒计时，此时数码管不会自动熄灭。故我采取了一种没有普遍性的解决办法，即因为我们实际上没有用到键盘的清零端，故把键盘的清零端的功能设置为输出清零信号。在充电完成信号输入到键盘清零端时，键盘模拟一次清零键被按下，从而将后级电路中残留的输入清零。由于状态机跳变所需的时间很短，故模拟按键的时间也很短，不会对电路的时间特性造成明显影响。

此时再把键盘上不必要的键（A、b、C）的输出改成数字 0，即完成了对键盘模块的修改，电路整体设计完成。

三、仿真结果

1、状态转换图

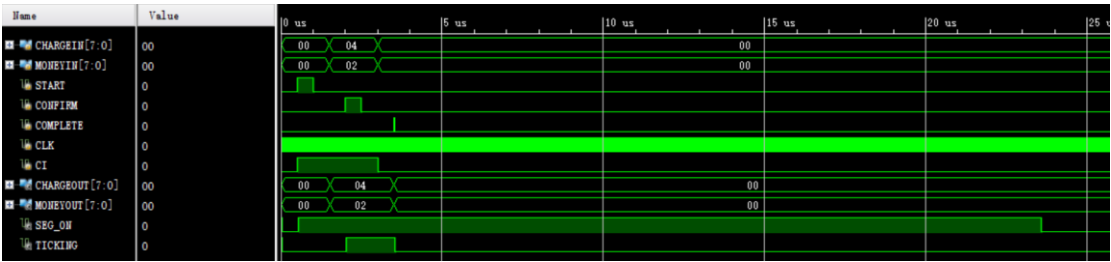


Source State	Destination State
1 s_10s	s_init
2 s_10s	s_charge
3 s_10s	s_10s
4 s_charge	s_charge
5 s_charge	s_10s
6 s_init	s_init
7 s_init	s_10s

由于在使用 vivado 调用 modelsim 仿真后无法看到状态转换图，故将代码导入到 Quaratus 中查看状态转换图。另外由于在状态转换的条件中引入了与逻辑，导致 s_init 向 s_10s 转换的条件识别错误，实际代码如下：

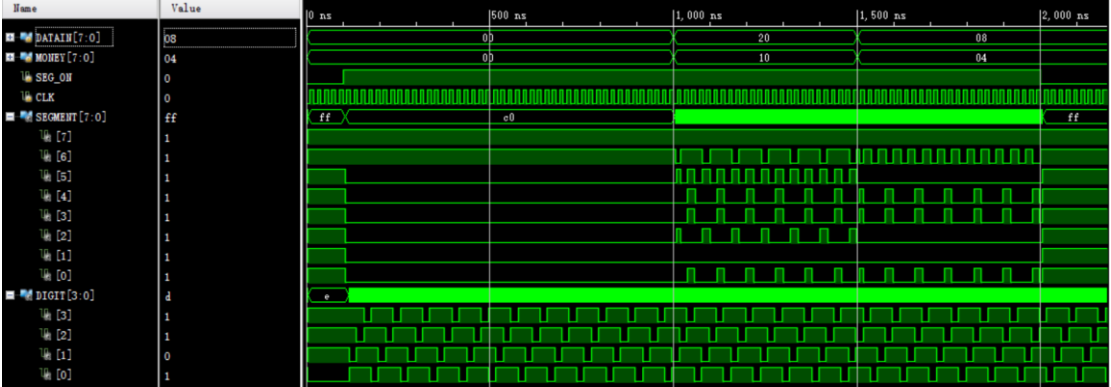
```
if(start==1'b1 && ci==1'b1)
    state<=s_10s;
```

2、状态机电路时序仿真



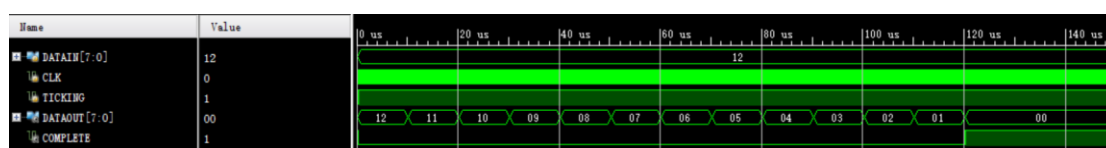
如图所示，按下开始 start 后数码管被点亮，并在有数据输入时同步输出数据。按下确认 confirm 后状态机输出开始倒计时信号 ticking，在接收到充电完成信号 complete 后停止输出倒计时信号。在输入状态下持续 1000 个工作周期 0.01s 后，数码管关闭，回到初始状态。

3、数码管显示电路时序仿真



如图所示，数码管启动信号 seg_on 为 1 时，数码管被点亮，电路同步输出显示数据 segment 与选通信号 digit。当 seg_on 再次关断时，四位数码管全部熄灭。

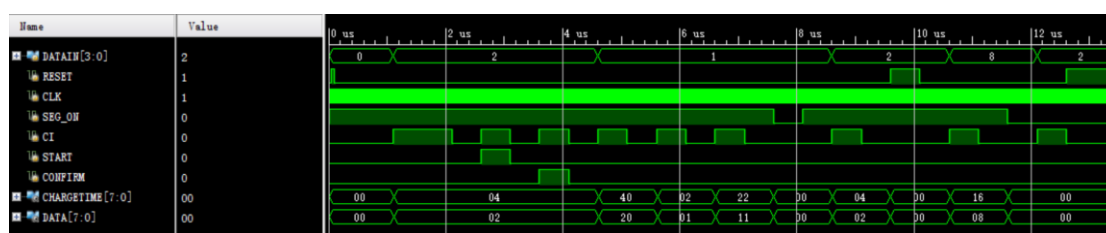
4、倒计时电路时序仿真



由于不同的 always 块不能对同一变量赋值的限制，本设计把倒计时模块与显示译码模块做了拆分，故单独仿真倒计时电路。

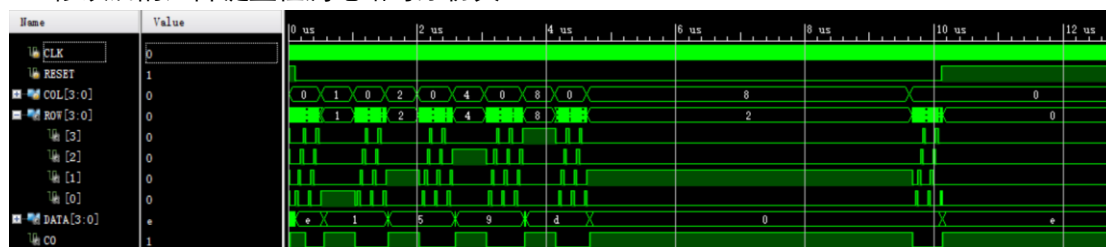
电路工频为 500Hz，故充电开始信号 ticking 为高电平时，每检测到 500 个 clk 上升沿，充电时间便-1s。当充电时间为 0 时，输出充电完成 complete 信号。

5、充电时间计算电路时序仿真



如图所示，电路能够对数码管启动信号 seg_on、清零信号 reset 做正确的反应，在数码管关断时不输出，在有清零输入时将输出全部清零；也能对开始信号 start 与确认信号 confirm 进行按键忽略。电路实现了输入金额的移位寄存与充电时间的计算，当充电时间达到上限时，由于采用了移位寄存，使得金额与充电时间的第二位均为 0，则逻辑为若再次输入，视为从 0 开始重新输入。

6、修改后的矩阵键盘检测电路时序仿真



如图所示，清零后键盘停止扫描，并输出 E 被按下（对应清零键）。在键盘不被清零时，行信号 row 进行扫描，扫描到对应的列信号 col 为高电平时，判定对应键被按下，仿真时通过选取激励时延，模拟了主对角线上的四种输出，说明每行每列的输入检测均有效。由于检测到输入电平变化，表示键被按下的信号 co 即为 1，故实现了防抖；长按键视为一次输出。

四、实验总结

1、实验中遇到的问题和解决方法

① 由于在做综合设计实验时，我的移位寄存器采用了异步设计，直接受键盘输出 co 的下降沿控制，故在本实验中需要对电路进行修改。于是我引入了对信号 co 的检测，在每次 clk 上升沿时若第一次检测到 ci 为 1，则有键被按下。此时对一个变量置 1，在 ci 退出之前，该变量始终为 1，表示不再在 clk 的上升沿对同一按键执行移位寄存操作。待某一上升沿时检测到 ci 为 0，即按键松开后，对该变量置 0，使其准备再次接收输入信号。由于该电路工作频率为 100Hz，即 0.01s，远快于人按键的速度，故电路能够稳定地读取每次一

输入。通过此方法实现了异步电路到同步电路的转变。

② 由于该电路模块多、较为复杂，故在构成整体电路并将代码写入硬件之前，我先对各个模块进行了仿真，确保输出无误。并且由于顶层模块过大，仿真效率很低，故对顶层模块，直接写入实验板上进行硬件调试观察问题较为高效。事实证明该方法取得了很好的效果。

③ 对于电路的模块间整体逻辑问题，需要判断应在哪个模块上做出修改，以解决问题。而能否解决问题通常取决于是否在对应的模块上做出修改，以及修改是否能改正逻辑问题。对于倒计时结束后的电路清零，我尝试过 10 种以上的设计方法，后来总结出问题的关键是键盘的输出，故修改键盘电路后最终实现功能。

2、实验收获

① 熟练掌握了小型电路的分模块设计方法，学会了编写由各类输入控制的状态机电路代码，提升了编写测试文件的能力。

② 训练了我书写代码的良好习惯，包括对 `always` 与 `assign` 的认识，阻塞赋值与非阻塞赋值的认识，`if` 与 `else` 的运用，`begin` 与 `end` 的运用。让我深刻意识到 `verilog` 与 C 语言间的不同。

③ 让我有了用代码描述电路，而非用代码直接实现功能的意识。