EDA 大作业二 投币式手机充电仪
实验报告

班级：_____自 72_____

姓名：_____高子靖_____

学号：___2017010917_____

目录

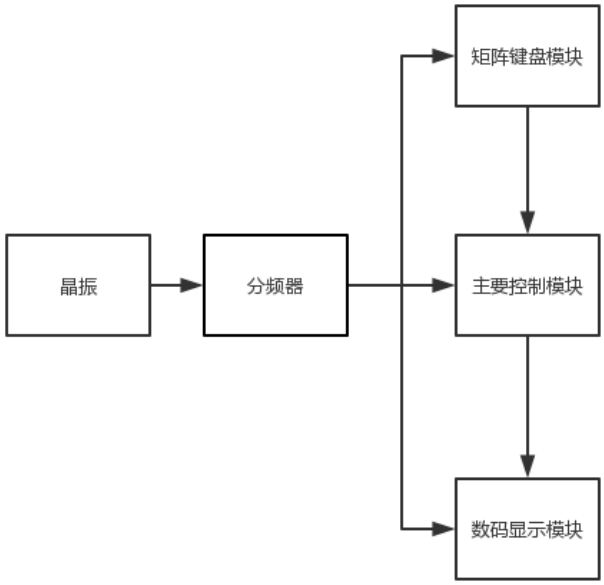
一． 实验目的

1. 学习自顶而下、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状态机电路的方法。

二． 预习任务

1. 阅读并分析任务要求，画出电路的总体框图。

根据任务要求，我们可自顶而下将任务进行划分，可以得到如下所示的整体框图。



可以看到，使用晶振以及分频作为整个电路的 CLK 信号，将电路功能的实现主要分为了四个模块，分别为：分频器、矩阵键盘模块、主要控制模块和数码显示模块，同时通过箭头指向可以看得出他们的输入输出对应关系，下面首先介绍这四个模块各自需完成的功能及引脚：

分频器：将来自晶振的 50MHz 的频率降频到合适的频率作为接下来三个模块的输入，这里选择与 EDA 中扫描频率相同的频率，也即 250Hz。

引脚为输入 ClkIn 和异步置零端 rst 输出 ClkOut，即分频前和分频后的信号。

矩阵键盘模块：该模块主要负责电路的输入部分，即将矩阵键盘的值以及是否输出值作为一个使能端 EN 输出到后级电路，该设计是为了后级电路识别是否有按键被按下，而该模块由于为了满足减少 I/O 口的需求，根据其电路图我们采用扫描显示，并且这里选择列作为输入不断对行进行扫描，输出键值（译码规则在源码中已注释）及使能端。

引脚为输入行 row，异步置零端 rst，CLK 信号，输出列信号键值 key_value 以及信号使能端 EN。

主要控制模块：该模块的输入为键盘模块给出的键值和 EN，输出为显示的时间以及投币的金额，同时也给出一个 start 信号，即给到最后的数码显示电路判别是否启动，也即是否将数码管全灭，其实现思想主要为状态机在闲置状态、输入状态和充电状态间转换来完成功能，在之后终结报告中详细叙述设计思路。

引脚为输入 CLK，异步置零 rst 和未经过译码的键值 key_value，输出为时间 outtime 以及金钱值 outmoney 和是否开始的选通信号 start_1。

译码显示模块：该模块完成对主要控住模块给出的时间值和金钱值进行扫描显示，

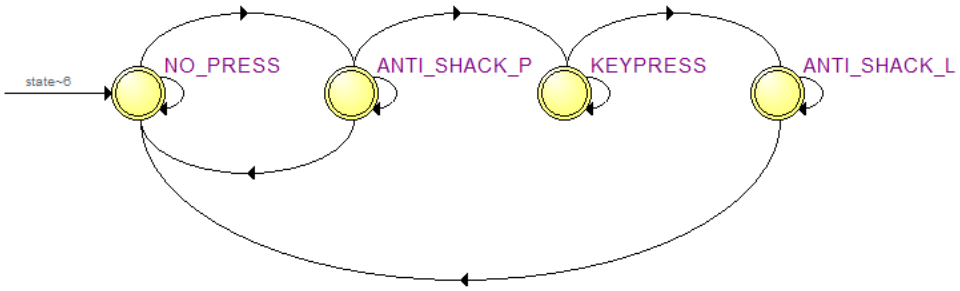
其扫描显示思想与 EDA1 中完全相同，不同之处在于不再使用 7448，而在该模块中完成译码，给出七段显示，并给出扫描数码管的位置，并给出每个数码管上的显示值。实现思路使用 case 语句即可.

引脚为输入端 CLK，异步置零端 rst，以及金钱 selmoney 和时间 seltime，输出为七段显示 a-g 具体见之后模块描述。

2. 根据实验任务要求画出电路的状态转换图。

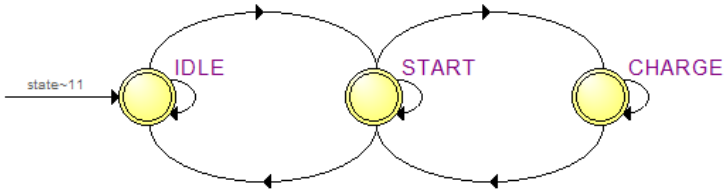
由于我在设计电路时只有两个模块是需要使用到状态机的，因此将下两个模块的状态转换图分别描述。

矩阵键盘模块：状态转换图如下所示



如上图所示，在该模块中我设计了四个状态，分别为未按键状态，按下防抖状态、按下状态、松键防抖状态。未按键状态下进行扫描，当扫描到列输入不全为 1 时，也即有键按下时会跳入按下防抖状态，通过计数防抖，计满后判别是否为按下状态，如果仍然在按下状态，则进行输出并进入到 Keypress 状态，Keypress 状态判别是否仍在按下，如按下则保持在当前状态，如已松开则进入到松开防抖状态，由于 NO_PRESS 时同样保持原值，因此对于输出值改变无影响，我们在这里的防抖只作为一个时间间隔作用，防止其他键抖动时从 NO_PRESS 状态开始到给值。

控制模块：状态转换图如下所示



在该模块中我设计了三个状态，分别为 IDLE（闲置状态）、START（输入状态）、CHARGE（充电计时状态），首先在没有输入时，处于闲置状态，只有当按下开始键时，进入 START 状态，数码管从 0000 状态跳出，否则始终保持在 IDLE 状态，进入 START 后进行输入，如果

按下确认键且 money 不为 0 时则跳入 CHARGE 状态，否则当 money 为 0 时开始计时，如果计时结束则跳回 IDLE 状态，数码管灭，其他情况将保持在原状态。进入 CHARGE 状态后，开始计时，计时结束后自动回到 START 状态，否则保持在 CHARGE 状态。

三．设计思路

本次 EDA2 我根据自顶而下的思想，将整体任务划分为四个模块，分别为分频模块、矩阵键盘模块、控制模块和译码显示模块，分块调试好每一个模块后在顶层电路中调用使用完成功能，且各模块使用 verilog 语言完成，下面分模块详细阐述设计思路。

分频模块

该模块在使用 verilog 语言完成时，主要利用计数器的思想，我这里使用 reg 型变量 count 用于计数，当达到其进制要求后，这里设有 parameter 常量 max 用于当做其取反的信号，将晶振 50MHz 进行降频使用，控制模块以及译码显示模块都采用与 EDA1 相同的 250Hz，而矩阵键盘模块为了更好地进行行扫描，这里单独使用 500Hz。

```
module CLK_DIV(ClkIn,ClkOut,rst);
   input ClkIn;
   input rst;
   output ClkOut;
   reg ClkOut;
   reg [31:0] count;
   parameter max=100000;
```

矩阵键盘模块

该模块的设计思路主要根据其矩阵键盘的电路图设计，由于其为减少 I/O 口，采用如下图所示的连接方式，因此在设计时采用依次对行线置低电平，不断检测列线状态的思路，因此也设计了三个状态。

未按下状态：也即扫描状态，在四根列线的全部为高电平时，开始对行线进行扫描，扫描频率为 500Hz，当有按键按下时，能够快速扫描到该线，扫描到该列线时会立刻跳入 ANTI_SHAKE_P 状态，而提高扫描频率也是能够使其在按键被按下时快速进入到按下防抖的状态，下图为其扫描状态。

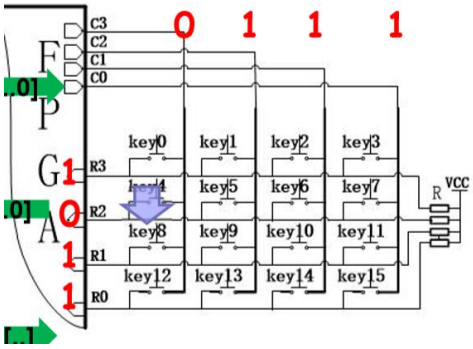
```
state <= is_col_idle ? NO_PRESS : ANTI_SHACK_P;
if(is_col_idle)
begin
   scanstate <= scanstate+2'b1;
   case(scanstate)
      2'b00:
         row <= 4'b1110;
      2'b01:
         row <= 4'b1101;
      2'b10:
         row <= 4'b1011;
      2'b11:
         row <= 4'b0111;
      default:
      begin
         row <= 4'b1101;
      end
   endcase
end
```

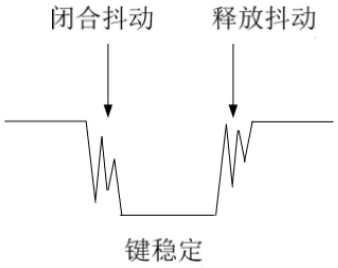
按下防抖状态：通过计数思想，按键抖动宽度大约在 20ms，而这里采用同步电路时钟

频率为 500Hz，因此使用四位二进制数 press 进行计数，计满 16 次后进行判别，值得注意的一点是该状态不再扫描，将行值和列值锁定。这里防抖宽度为 32ms 能够跨过抖动宽度，经检验能够起到良好的防抖效果。

按下状态：在该状态时进行键值的输出以及使能信号的给出，实际上为了使 EN 与 key_value 在同一个 CLK 上升沿给出而将其写在防抖状态在计满的 else 情况中，Keypress 状态在松开前进行保持，如果列线不全为 1 说明已经松开，则进入松开防抖状态。

松开防抖状态：该状态通过计数来进行延迟，使用二位二进制数 loose，其思想与按下防抖相同，但对于长按键的处理，我在除按下状态键值会进行新的译码外，其他三个状态键值都保持，因此对于松开防抖并不使其回到 Keypress 状态而在计数满后回到未按下状态。实际上起到的是延迟作用，为了防止在松开时的抖动改变键值而在松开时进行延迟，当其回到未按下状态后一定会有按下防抖来防止抖动，因此这里短暂延迟防止松开快速按下的误判。





控制模块

该模块同样设计三个状态间装换完成任务，设计思路主要是在 IDLE 闲置状态维持原值，且数码管的扫描选通信号不给出，START 状态即进行输入，根据题目要求金额显示最多为 20，因此我设计为循环输入通过计算完成。

```
else if(money_low < 2)
begin
if(key_value < 10)
begin
   outmoney <= (outmoney%10*10 + key_value) %100;
end
else if (key_value == key_nosense || key_value == key_start)
begin
   outmoney <= outmoney;
   outtime <= outtime;
end
```

这里 money_low 是对于当前输出的 money 模 10 也即取其低位，如果小于二，那么低位乘 10 作为高位加上按下的键值即完成输入，对于低位大于等于 2 的情况，我这里在按下数值键直接输出 20 与 40，其他情况进行保持，这也是进行输入的主要思路，在这里能够实现当先后输入 123 时，认定输入为 23，因此最终会显示 20，实现循环输入。

对于十秒无动作的任务要求，我的理解是在投币数额为 0 时开始计时，10 秒后没动作则回到初始数码管全部熄灭状态，因此我在 outmoney 为 0 时便开始 resetcount 的自加操作，否则一旦 outmoney 不是 0，立刻将之前的计数清零

另一方面在状态间转换的处理上，较为关键的一点是我在设计充电状态与输入状态时，如果按下充电按键时，必须输入金额大于 0，否则会保持在 START 状态。因此不会出现锁在充电状态的情况，当在充电状态如果充电时间回零，立刻将金钱显示回零并跳回输入状态，充电计时同样使用计数器实现，在这里为 downcount 用于计数，由于电路时钟频率为 250Hz 因此每一秒也即将其计到 250 后回 0 并将输出 outtime 减一。

思路上关键之处在于是否有按键被按下的处理，由于我在键盘模块中给出键值后便将其锁存住，知道下一次键值到来，如果仅取键值的改变作为按键按下的标志，无法判别两次按下相同按键的情况，因此我在键盘模块中给出了使能端 EN 与键值同步给出，因此在该模块中使用 LAST_EN 始终保存上一时刻的 EN，便可取其上升沿作为按下的标志。

```
always @ (posedge CLK or negedge rst)
begin
   if(rst == 0)
   begin
      last_EN <= 0;
   end
   else
   begin
      last_EN<=EN ;
   end
end
```

译码显示模块

该模块思路整体与 EDA1 的扫描显示相同，250Hz 频率依次选通数码管进行显示，这里由于我们需要对四个数码管进行显示，因此使用 reg 型变量 dig_num 对选通的数码管应显示的数据进行选择并寄存，且方便之后的译码，另一部分就是在该模块中的 BCD 译码，这里将使用 verilog 语言进行七段显示的译码而不再使用 7448 因此最终给出的为七段显示 a-g，译码时使用 case 语句对 reg 判别即可。而数码管是否进行扫描选通也由之前给出的 start 信号决定，也即跳出 IDLE 闲置状态后才不会数码管全灭。

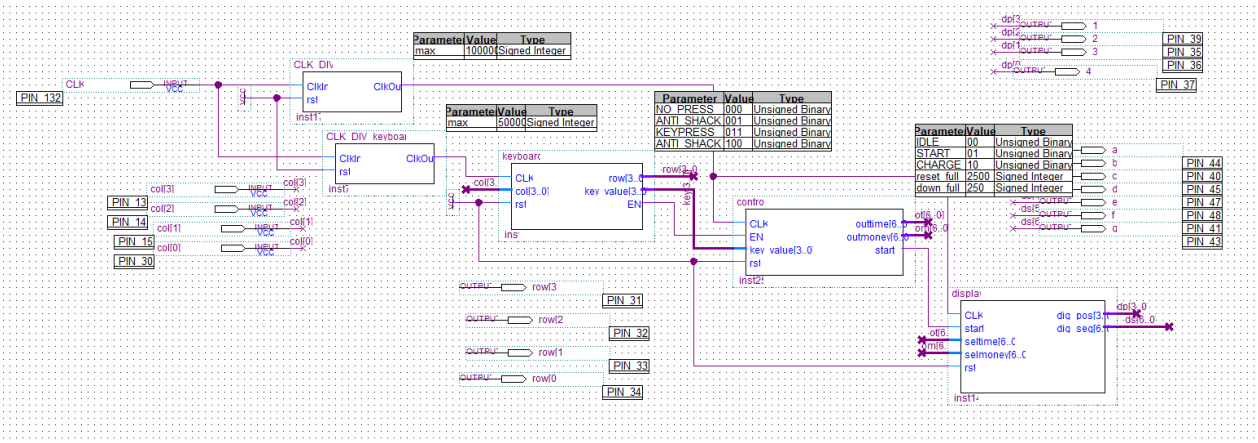
```
   if(start == 0)
   begin
      dig_pos <= 4'b0000;
   end
   else
   begin
   case(state)
      2'b00:
      begin
         dig_pos<=4'b0001;
         dig_num<=seltime/10;
      end
      2'b01:
      begin
         dig_pos<=4'b0010;
         dig_num<=seltime%10;
      end
      2'b10:
      begin
         dig_pos<=4'b0100;
         dig_num<=selmoney/10;
      end
      2'b11:
      begin
         dig_pos<=4'b1000;
         dig_num<=selmoney%10;
```

其他

除去上述模块的一些关键思路外，我所有的模块都设置了异步置零端，这是为了在进行 modelsim 联合仿真时为了方便的将模块中的中间变量以及输入都进行初始化，而不会在仿真时出现未知值而进行的改动，再者，该键盘模块验收前为列扫描由于其会跳回第一列因此将频率降到极低进行扫描，这样并不能使按键足够灵敏，因此在验收后将其改为行扫描后发现能够很好地在高频下扫描工作。以及全部模块都为同步电路，且我最终代码都为单进程实现状态的转换，这是由于我在多进程编写时很难生成状态机，为了能够使 quartus 生成状态转换图，改为单进程编写。

四．顶层电路图及模块电路的功能

1.顶层电路图



可以看到，顶层电路进行了明显的模块划分，左侧两分频器分别提供 250Hz 和 500Hz 的频率。得到的低频依次连接到封装好的三个模块中实现相应的功能，最终输出选通数码管以及七段显示的引脚，具体功能已在之前讲解，下面对于每个模块一些关键功能的实现进行代码上的讲解。

2.模块电路的功能

分频器模块

分频器模块实现从晶振的降频，通过计数后取反相实现。

矩阵键盘模块

矩阵键盘模块实现的功能主要为两部分，第一部分为读取按键按下的信号并确定相应的按下位置，第二部分为将按下的键位置译码为相应为二进制数便于之后的控制模块对于数据的处理，译码如下图所示。

```
else  if(!is_col_idle)
   begin
   state <= KEYPRESS;
   EN<= 1;
   case({row,col})
   8'b1110_1110: key_value<= 4'b0001;
   8'b1110_1101: key_value<= 4'b0010;
   8'b1110_1011: key_value<= 4'b0011;
   8'b1110_0111: key_value<= 4'b1011; //按键11 开始
   8'b1101_1110: key_value<= 4'b0100;
   8'b1101_1101: key_value<= 4'b0101;
   8'b1101_1011: key_value<= 4'b0110;
   8'b1101_0111: key_value<= 4'b1100; //回0  键值12
   8'b1011_1110: key_value<= 4'b0111;
   8'b1011_1101: key_value<= 4'b1000;
   8'b1011_1011: key_value<= 4'b1001;
   8'b1011_0111: key_value<= 4'b1010;  //10 进行计时
   8'b0111_1110: key_value<= 4'b0000;  //0
   8'b0111_1101: key_value<= 4'b1111;  //无效键15
   8'b0111_1011: key_value<= 4'b1111;  //无效键
   8'b0111_0111: key_value<= 4'b1111;  //无效键
   default: key_value<=4'b0000;
   endcase
end
```

数字键全部按照其值进行译码，其他键按照注释译码，无效键全部译码为同一数值便于之后的处理。

控制模块

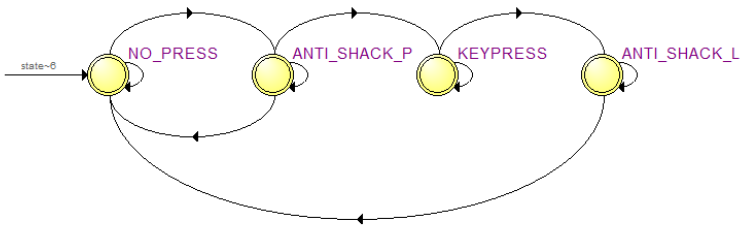
该模块实现的功能主要为读取键值，处理数据。首先输入接收来自键盘模块的键值和使能信号，在中间变量处理判别按键信号后处理数据，并且完成充电倒计时的数据处理。这里给出的均为七位二进制数，一次保证能够输出 40 以内的数值，以及给出是否选通数码管的信后给下一级模块，主要功能为数据处理。

数码显示模块

该模块的主要功能为显示作用，在 250Hz 频率下扫描选通数码管，并且将输入的 money 和 time 进行一定得转换，也即取其高位和低位分别在不同的数码管进行显示，在根据七段显示数码管来译码为相应的数字，完成显示。

五．状态转换图及其说明

键盘状态转换图：



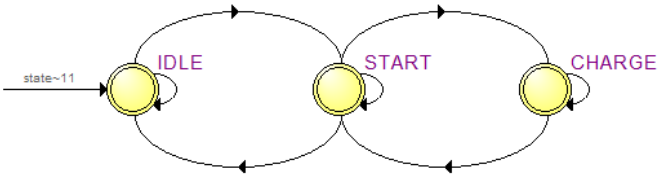
NO_PRESS→ANTI_SHAKE_P:当 col 不全为高电平时。

ANTI_SHACK_P→KEYPRESS/NO_PRESS：当防抖计数记满后，如果 col 不全为高电平则回到未按键状态，如果仍按下则进入按下状态。否则将继续计数，保持原状态。

KEYPRESS→ANTI_SHACK_L:当松开按键后 col 不全为高电平，则跳入松开防抖状态，否则继续保持原状态。

ANTI_SHACKE_L→NO_PRESS：在该状态下，只进行计数，其作用为延迟，因此不会回跳到按下状态，计满后自动跳回 NO_PRESS 状态。

控制模块状态转换图：



IDEL→START:当按下开始键时，状态跳入 START 状态，可以进行输入，数码管被点亮并全部显示 0，否则保持在闲置状态。

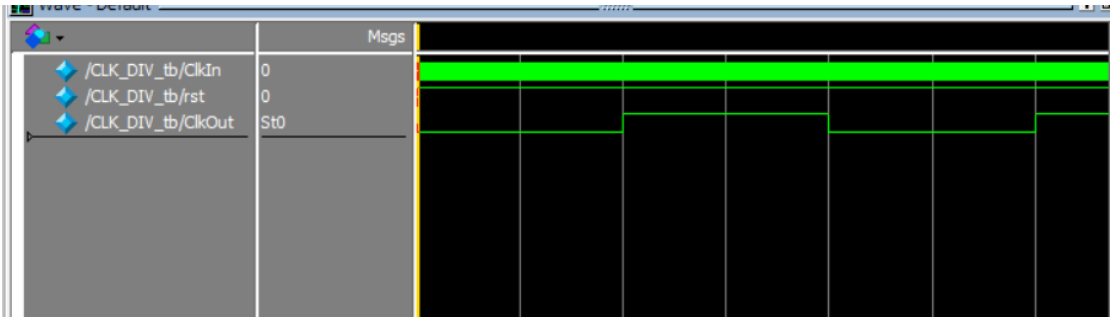
START→CHARGE/IDLE：当输出的 money 为 0 时，开始计时十秒无动作则回到 IDLE 状态，数码管熄灭，如果按下确认按键且 mone 不为 0，则进入 CHARGE 状态，否则保持在原状态。

CHARGR→START：当倒计时回 0 后，自动回到 START 状态，否则保持在当前状态继

续进行倒计时。

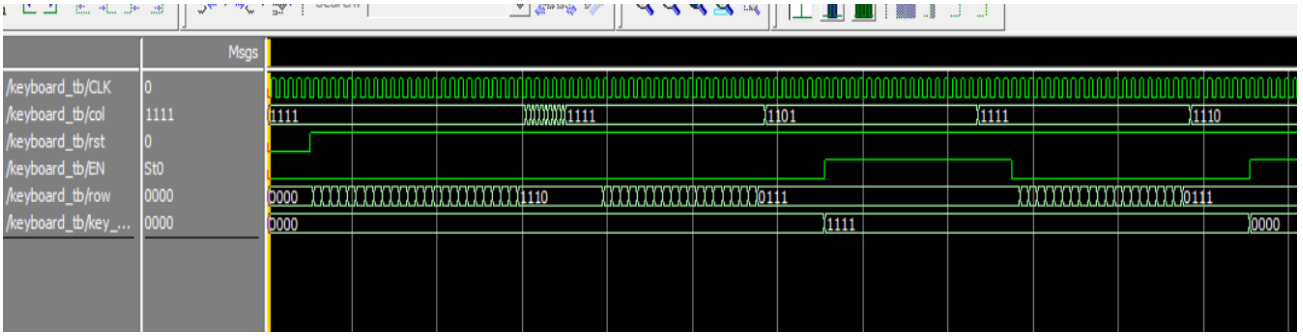
六．波形仿真图及其说明分析

1.分频器仿真

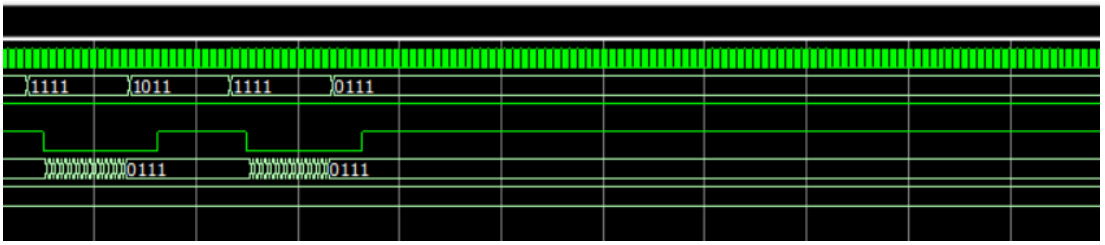


从以上波形能够明显看出分频器的分频功能，由于不断缩小波形图，导致 CIKIN 信号在上图中十分密集，这也是因为将 50MHz 的频率降为 250Hz。

2.矩阵键盘仿真

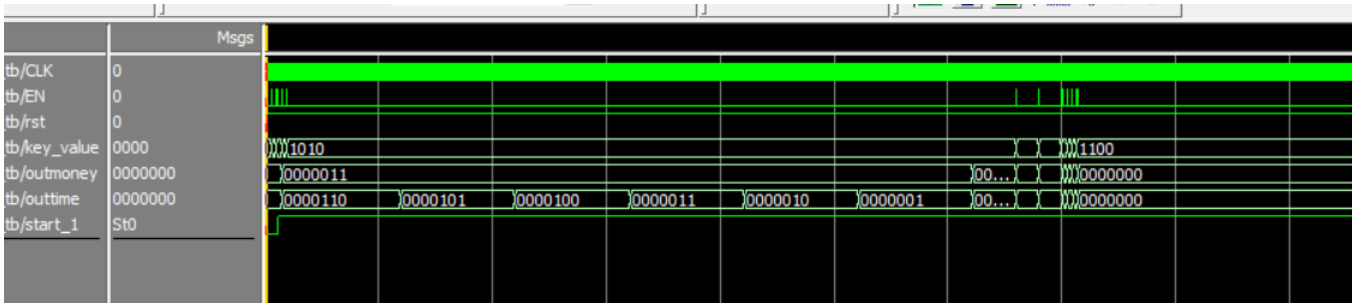


从波形图中可以看出，当异步置零信号从 0 给到 1 之后，开始进行行扫描，其中第一次 col 不全为高电平为对抖动状态的模拟，可以看到在抖动期间使能信号 EN 和 key_value 键值均为发生改变，这是由于抖动宽度一定小于我所设置的防抖宽度，在实际电路中，防抖宽度到达 32ms，超过了抖动宽度 20ms。可以看到 col 抖动期间 row 值不会改变，因此跳入防抖状态会立刻停止扫描，经过防抖宽度后继续扫描行，之后模拟按键，可以看到 EN 与 key_value 同步上升，因此可以在之后模块取 EN 的上升沿作为按键按下的标志。

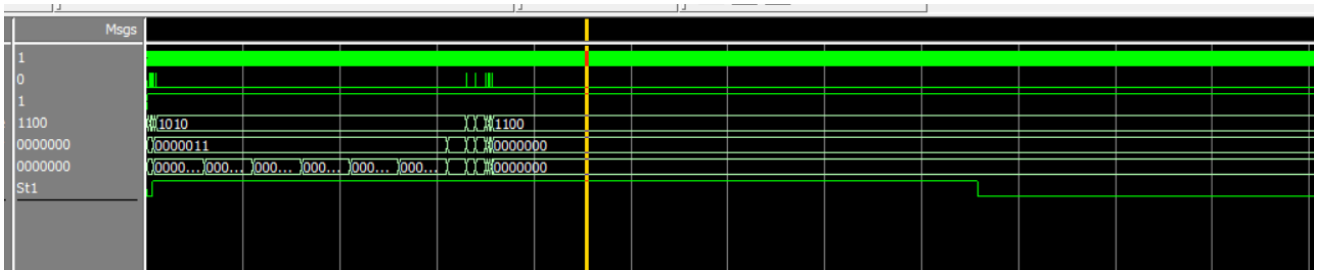


如上图所示，最后波形仿真的为长按键的情况，因为我在每次按下后，无论在松开防抖状态还是回到未按下状态，我都会使键值 key_value 保持键值，因此最后的长按键保持在 Keypress 状态，EN 端保持 1，键值保持不变。

3.控制模块仿真

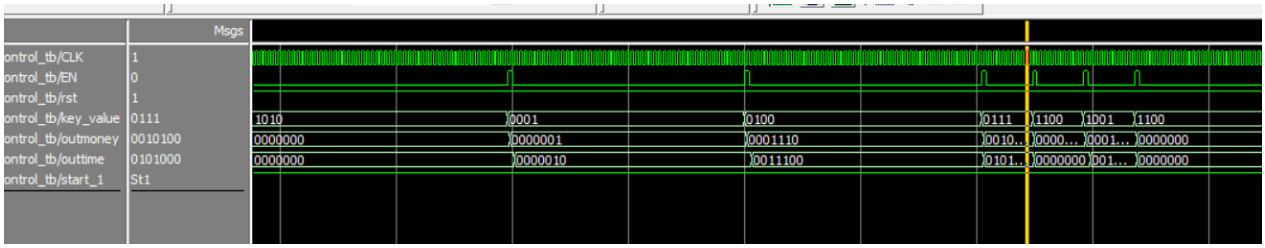


首先在按下开始键之前，按任何其他键输出都不会改变，只有当按下开始键时，输出金额和时间保持不变，start_1 代表数码管选通，即开始扫描显示，进入 START 状态，在十秒内进行输入，例如计时开始时表示金额为 3，而时间为 6，按下确认键后开始计时，可以看到每隔相同时间（实际上是 250 个 CLK 信号）输出时间便会减一，计数结束后停留在输出为 0 的 START 状态。



我们放大之后的状态，可以看到在输出回到 0000 以后，经过十秒无动作，start 信号回到 1，也即数码管熄灭。

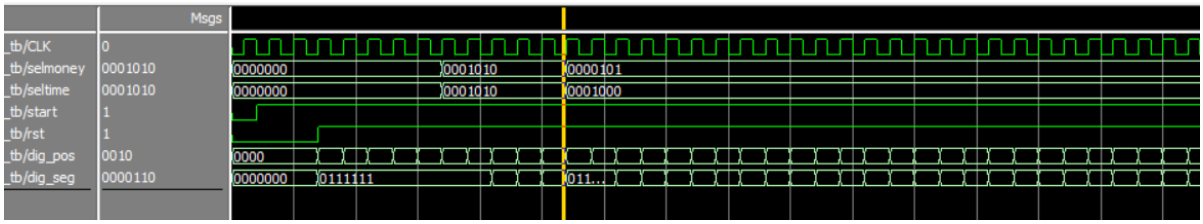
对于多次输入的情况可以对上图计时后的部分放大观测，如下图所示：



能够看到，当我连续输入键值 1、4、7 时，应改首先显示 01、14、20，由图可以看得到波形显示和预想的结果相同，即实现了循环输入且不会金额超过 20。

4.显示模块仿真

该模块的波形仿真为扫描显示以及给出的七段显示的数据，检验时需要根据数码管的七段显示符所在的位置进行检验，其波形如下。



从上图能够看到，当 rst 信号为 1 且 start 为 1 时开始扫描，并且当 money 和 time 给出后，将进行扫描显示，在不同的数码管选通时，将显示不同的信息，具体的显示原则为自左至右依次显示为 money 高位 money 低位，time 高位和 time 低位。

七．设计和调试中遇到的问题及解决方法

总体而言，由于这是第一次使用 verilog 语言，对于语法的不熟悉以及使用语言设计方法的陌生，在完成大作业的过程中还是出现了不少的问题，我也花费了很长的时间去解决这些问题，包括在最后进行验收的时候，仍被助教指出键盘由于防抖过长而被认定为功能上的问题，在验收后我也继续思考将键盘模块进行了改进，最终实现了能够良好实现功能的版本，下面我将叙述我的大作业完成过程中出现的典型问题。

键盘模块回跳问题：这个问题是我在首先开始完成大作业时便出现，由于我首先去实现键盘的输入功能，因此大作业初期我单独完成键盘模块并搭接顶层电路，使用 7448 显示在板子上对键盘进行调试，在初次尝试时我便采用三段式来写，但防抖方法为依次扫描列，每一列的扫描作为一种状态，由此构成较为复杂的状态转换图，但经检验会出现严重的问题，每次都会回到其第一列值稳定下来显示。我曾认为这是在列扫描时列线可能因传输延迟等原因回到第一列被按下的情况。因此我在第二版键盘中换用计数防抖，扫描为为按下时扫描，基本思路和最终版大致相同，但在高频下出现第四列会回跳的情况，导致我的无效键也会被当做按下 0 处理，我通过修改其扫描频率到 25Hz 才得以解决这个问题，这也使得操作时需较长按键，也是我在验收时被助教指出的问题。但验收后我想修改此错误，便想要不再回跳到第一列，便不再扫描列，改为对行扫描进行尝试，经修改后，发现行扫描的确能够解决问题，可以在 500Hz 的频率下扫描，按键反应灵敏。

控制模块键值处理问题：我在处理键值时，由于起初在进行键盘译码时为顺序译码，即依次译为 0-15，这样也方便输出，但在主模块处理时便出现问题，由于对于数字键三行所代表的键值只有一行是与译码相同的，其余均差 1，因此处理起来十分复杂，我首先使用不等式来分三行处理，但发现上板子后，判别条件没有起到作用，因此我改为了逻辑等号把每一个键值的情况都用 if 语句写了出来，但感觉这样写十分复杂，因此我决定修改键盘模块的输出译码，数字键译 0-9，其余均译为 10 及以上，并在 control 模块通过 parameter 替代，最终处理起来非常方便，也能够正确读取键值。

仿真及状态机问题：这部分问题体现在用代码写完后，并不能生成状态机，以及仿真时输出变量均为 xxxx，也即未知值，之后经上网查阅得知是 modelsim 对于输出并没有一个起始值，因此输出是未知的，我也修改代码给每一个模块加入异步置零端，之后键盘模块的输出仍然没有改变，为了查看问题所在我单独拉出防抖的中间变量 press 在波形图观察，发现并没有计数，可能是由于多进程写法我在一些输出用了异步电路导致 modelsim 无法识别，因此我将所有模块都改为了一段式同步电路写法，以及在一些中间变量没有改变的地方加入锁存，最终经检验达到了理想的效果。

总而言之，这次的 EDA2 我有很大的收获，不仅在解决问题的思路上，也有在 verilog

的运用上，经过不断的修改不断的改进最终才能没有 bug 地实现功能。