实验八 EDA 作业三

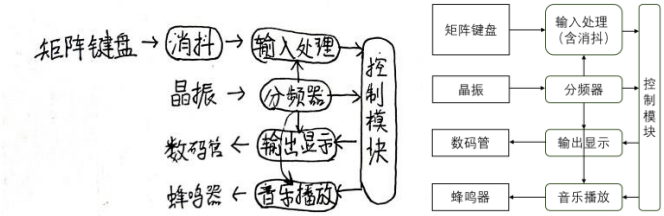
自 35 夏弘宇 2023011004

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. 掌握规范使用硬件描述语言描述状态机电路的方法。

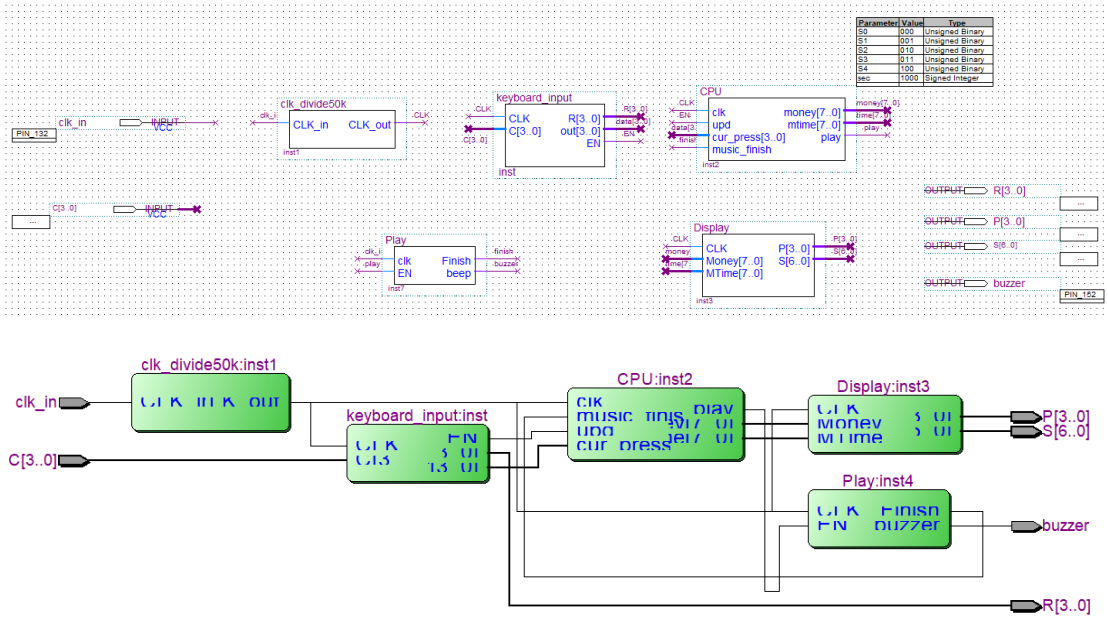
二、实验报告

1. 分析任务要求，划分主要功能模块，附上顶层电路图。



前者是初步设想，后经分析，发现消抖可以与输入处理有机结合起来，简化消抖功能的实现。

附：顶层电路图



2. 结合各模块电路工作原理，说明模块引脚功能。

（1）输入处理 keyboard_input（含消抖）

工作原理：本质上是一个状态机，两个大状态：S0 无按键被按下，S1 有按键被按下。每个大状态下还有 31 个状态，类似从 0 到 30 的计时器。

S0→S0：输入 C=1111 没有检测到低电平（按下），输出 R 进行移位操作（持

续扫描）

输入 C 检测到低电平，停止扫描，持续监视当前的 R

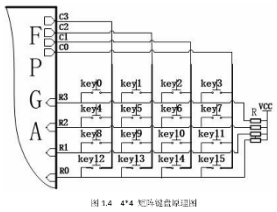
S0→S1：检测信号持续一段时间（连续计数 30 次即 30ms），则证明信号已稳定，转移到按下状态

S1→S1：持续监视当前的 R，输入信号仍然存在就保持按下状态

S1→S0：检测信号消失一段时间（连续计数 30 次即 30ms），则证明信号已消失，转移到释放状态

输入引脚：时钟信号 CLK、矩阵键盘的列线 C[3:0]

输出引脚：矩阵键盘的行线 R[3:0]，按下按钮的编号 data[3:0]，指示数据更新的信号 EN（数据更新后输出一个正脉冲）



注：编码方式与右图一致

评价：该模块具有很强的移植性，基本所有关于矩阵键盘的操作（不要同时按 2 个键）都可以用该模块进行处理，实现了标准接口设计。

（2）分频器 clk_divide50k

工作原理：50k 倍分频器，每 25k 个输入上升沿进行一次反转。

输入引脚：晶振信号（50MHz）

输出引脚：CLK_out（1000Hz）

（3）控制模块 CPU

工作原理：一个较为复杂的状态机，见第 3 部分。

输入引脚：时钟信号 CLK、示数据更新的信号 upd、当前按下按钮的编号 cur_press[3:0]、音乐播放完毕信号 music_finish

输出引脚：当前金额（money[7:0]）、剩余时间（mtime[7:0]）、音乐播放信号 play

（4）显示模块 Display（分为位选和段选模块）

工作原理：由位选、段选部分组成，位选扫描四个位并选择数据，段选实现 BCD 码向 7 段数码管的转化。

输入引脚：时钟信号 CLK、当前金额（money[7:0]）、剩余时间（mtime[7:0]）

输出引脚：位选信号 P[3:0]、段选信号 S[6:0]

（5）音乐播放模块 Play

工作原理：音乐播放模块通过 PWM（脉宽调制）技术生成音频信号。模块的主要输入包括时钟信号 clk 和使能信号 EN，而输出则是一个指示完成状态的 Finish 信号和一个用于发声的 beep 信号。

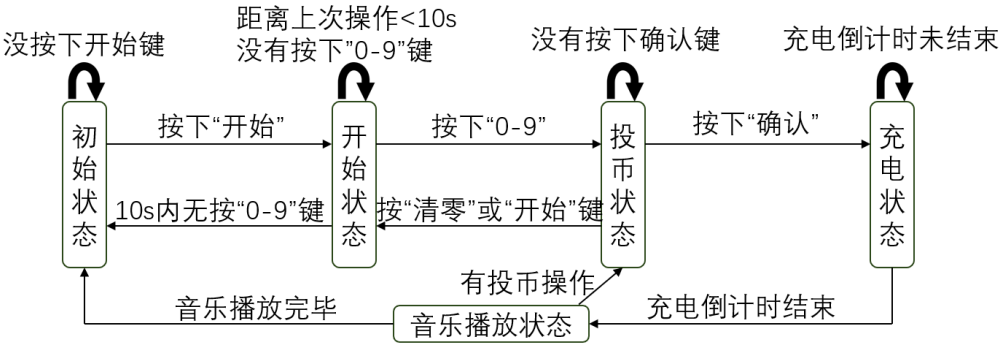
cnt0 的计数在 EN 为高时进行，beep 信号的生成基于 cnt0 和预设值

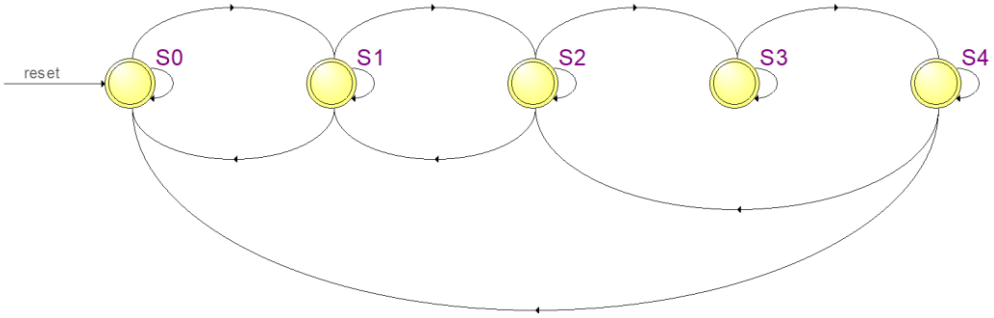
pre_set。cnt1 和 cnt2 的计数逻辑类似，用于控制音符的节奏和选择音符频率，通过 case 语句决定 pre_set 的值。pre_set 值遍历一个数组，数组中预先编写相应的音符序列。

输入引脚：时钟信号 clk（50MHz）、音乐播放信号 play

输出引脚：音乐播放完毕信号 music_finish、输出给蜂鸣器的电压信号 beep

3. 控制电路状态转换图，说明每个状态的含义和状态间跳转条件





（S0）初始状态：无显示，按除了开始键都有反应

（S1）开始状态：显示 0000，按下数字键可以投币

（S2）投币状态：按下数字键可以投币（十进制）（超过 20 均按 20 算），显示 XXYY，XX 是投币数额，YY 是相应充电时间，值为金额的 2 倍

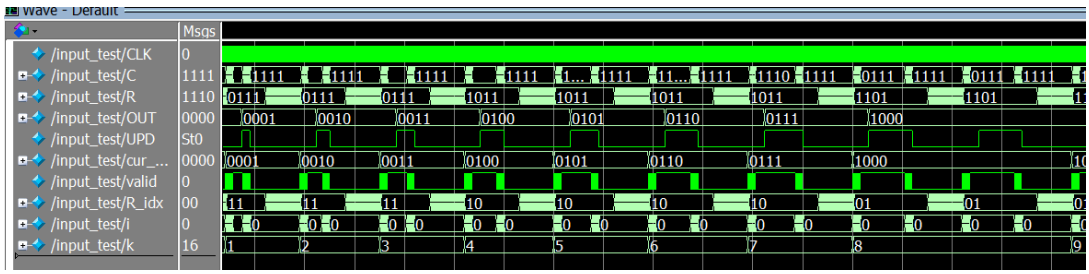
（S3）充电状态：倒计时，金额不变；倒计时完成后均清零

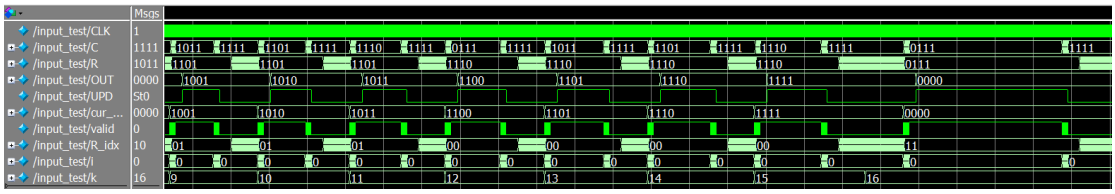
（S4）音乐播放状态：播放一段音乐

跳转条件见框图

4. 仿真波形图及其分析说明

（1）输入模块 test 仿真波形

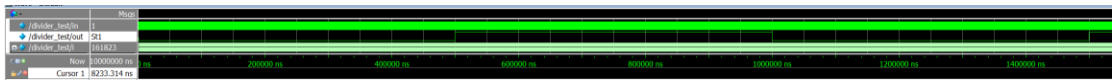




如上图所示，CLK 为时钟信号，C/R 分别为矩阵键盘的列和行信号，OUT 指示哪个键被按下，UPD 上升沿表示新信号来了（方便后续器件识别），valid 为抖动过程中指示信号传输情况的 Bool 量，抖动过程中不断反转 valid。

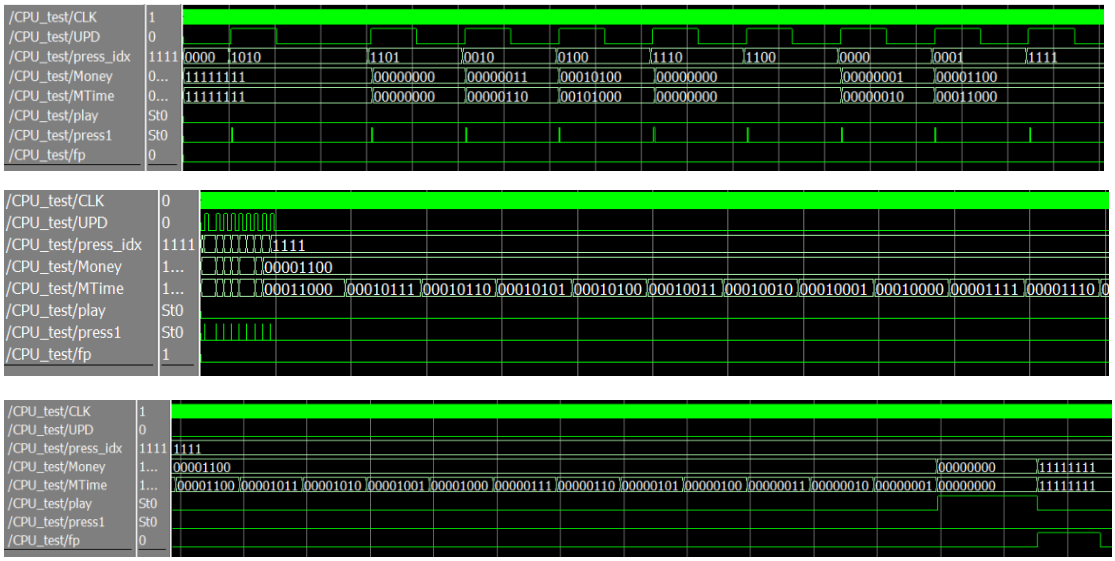
从 0-15 依次遍历各按键（k 即按键编号，二进制表示为 cur_press），其中重复按 2 次 8，最后长按 0，可以发现得到的 OUT 均与 k 与 cur_press 一致，且 UPD 能在按下一小段时间后产生上升沿，高电平持续到按键确定释放后；重复按 2 次 8 就有两个上升沿，长按 0 稳定地只有一个上升沿，符合预期，可以认为电路很好的实现了对长按键输入和抖动按键的读入。

（2）分频模块的仿真波形



仿真时输入时钟周期是 20ns，输出为 1000000ns，实现了 50k 倍的分频。

（3）控制模块的仿真波形

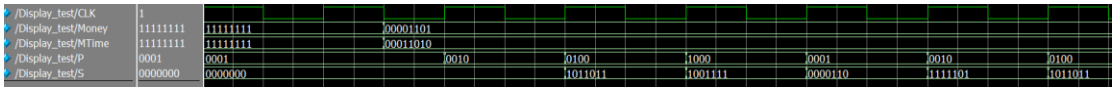


各变量含义：CLK 时钟信号 1kHz，UPD 即前述 EN 信号，指示按键被按下，press_idx 是指定按键的编号，Money 为显示金额，MTime 为显示倒计时，play 是使音乐播放的信号，press1 是 CPU 识别到新按键信号后的响应（即指示 UPD 的上升沿）。

输入完成了从开始——输入数字——清零——输入数字——确定——计时——过 10 秒自动关闭的完整流程。具体地，第一张图中，初始状态下，Money 和 MTime 都是 8'hff，表现为无显示；按下 9，识别到了 press，且没有其它动静，符合要求。按下“开始”，Money 和 MTime 都是 0，按下 3，显示 0306；按下 4

由于超过了 20 的上限，就显示为 2040（我的设计中超过 20 就一直显示 20，再按键就不动了），按下“清零”显示 0000，依次按下 0,1,2 键，依次显示 0000, 0102, 1224，按下确定开始计时，计时过程在第二张图中，Money 保持不变，MTime 每秒减 1；第三张图中，倒计时结束时，Money 也同时变为 0，并产生音乐播放信号，音乐播放完毕反馈一个完毕信号，就回到初始状态。

（4）显示模块仿真波形



位选信号能重复对各位进行扫描，并选出相应位的数，段选信号可以准确译码。

三、实验报告-实验总结

1. 设计和调试中遇到的问题，采用的测试方法及解决方法。

• 产生如下报错：time 是保留关键字，不能作为变量名，更换变量名即可

```
syntax error at input.v(56) near text "time";  expecting an identifier ("time" is a reserved keyword )
```

• 如下的 for 循环是 C++常用的格式，但在 Verilog 中不能这么写，必须在 initial 块外部进行变量声明，且 int 类型也要用 integer 进行声明。修改的同时也要注意变量名不要重了！而且++，-- 在这里也不适用，要改成 i=i+1 的完整形式。

```
syntax error at input.v(60) near text "i";  expecting "="
for(int i=15; i!=0; i--)
```

• 仿真结果出不来（如右），实际上，仿真并不要求与现实中严格对等，尤其是时间上，可以将 10ns 看成一个时钟周期，进行放缩。

```
# .main_pane.objects.interior.cs.body.tree
# run -all
# Press for 4000000000 ns
# Press for  100000000 ns
# Press for  200000000 ns
# Press for  300000000 ns
# Press for  400000000 ns
# Press for  500000000 ns
# Press for  600000000 ns
# Press for  700000000 ns
# Press for  800000000 ns
# Press for  900000000 ns
# Press for 1000000000 ns
# Press for 1100000000 ns
# Press for 1200000000 ns
# Press for 1300000000 ns
# Press for 1400000000 ns
# Press for 1500000000 ns
# Press for 1600000000 ns
```

• 仿真结果中，部分变量值一直没变，都是结束时的值：按顺序运行的代码一定要用阻塞式赋值（上一句执行完才能执行下一句！）

•上一条也有可能是拖到太后面了，要从 0ns 开始看（鼠标滚轮不利索的话，可以点击左上角的+/-符号）

• 编译时出现如下报错，意为同一个变量不能在多个触发条件下修改赋值，否则会出现冲突。这主要源于设计时为了将上升沿信号转化为电平信号，使用之后又要及时清零，因此出现在了两个不同的触发条件中。为了避免这样的情况，不妨开两个变量，上升沿信号控制一个变量取反，clk 信号到来后进行对比，若两个变量值不同，则说明上升沿信号来过了，那么在 clk 触发状态下对相应变量赋值即可。

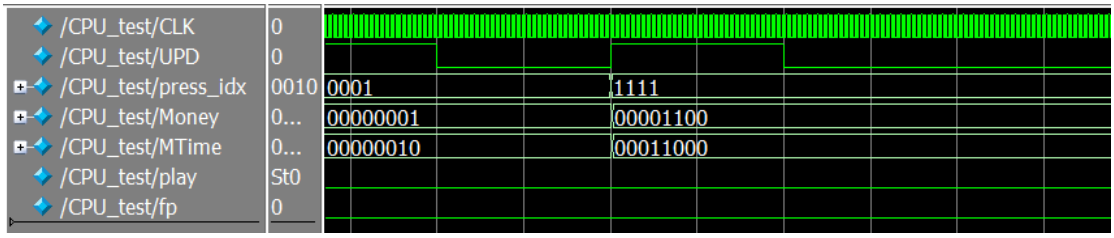
```
10028 Can't resolve multiple constant drivers for net "press" at CPU.v(58)
10029 Constant driver at CPU.v(34)
10028 Can't resolve multiple constant drivers for net "finish_play" at CPU.v(58)
10029 Constant driver at CPU.v(38)
```

• 调试时发现仿真波形中出现按下 Reset 信号后 money 并未清零的情况，发现确实忘记清零了，补上。

• 变量 money 同时使用了阻塞赋值（=）和非阻塞赋值（<=），这在 Verilog 中是不允许的。所有对同一变量的赋值必须一致，要么全部使用阻塞赋值，要么全部使用非阻塞赋值。

```
10110 Verilog HDL error at CPU.v(6): variable "money" has mixed blocking and nonblocking Procedural Assignments -- must be all blocking or all nonblocking assignments
12153 Can't elaborate top-level user hierarchy
```

• 波形中，输入 1111（按下确认键）后，Money 值从 1 变为 12，不符合预期，经检查，是 testbench 中按键信号与 upd 信号同时出现，形成竞争关系；将 upd 信号滞后 5unit 时间再出现解决该问题。这一情况也启发我对 input 模块进行了修改，将 upd 信号与输出的出发信号 EN 分离。



• 原因：连接模块端口的中间变量应该使用 wire 类型，而不是 reg 类型

```
10663 Verilog HDL Port Connection error at Display.v(9): output or inout port "S" must be connected to a structural net expression
```

• vsim-3037： 对象实例化要加名称，在这一实验中无需关注名称具体是什么，只需不与保留关键字重复即可。

```
Display(
    .CLK(CLK),
    .Money(Money),
    .MTime(MTime),
    .P(P),
    .S(S)
);
```

• ID:275058 Signal <signal name> drives an input pin，官方原因：指定的信号驱动输入引脚。但是，input pin 不能由信号驱动。实际原因，搞混变量名了，这时候主要看一下有没有变量名（形参与实参）对应错误的情况。

• 实物中按下确认就会跳成 2040 的样子，怀疑是输入模块有问题，经多次尝试，发现是数据与按下信号不同步的问题。当前信号对应上一个按下的数字！即 EN 信号来了，但数据 cur_press 还没成功赋值。让 EN 在下一个时钟信号才更新就能解决这一问题！

• 第二次完成充电时，音乐是从上次结束的地方开始放，而不是从头开始：需要在 Finish 信号发出的同时对播放指针进行清零操作。

• 音乐播放完并不会停止，而是从头开始循环播放：因为 clk(50MHz)与 CLK(1kHz)不同步，Finish 信号只有 20ns 的宽度无法被有效探测到，需要保持 Finish 信号一段时间，直到 EN 信号变为低电平（即增加一个反馈机制）。

• 刚载入程序就开始播放音乐：未对 play 信号进行初始化；刚载入程序就

开始发出尖锐的声音：初始条件下 cnt0=0 与 pre_set 初始值 0 相等，确实会使得 beep 输出 1，令初始时 pre_set 为 1 即可——所有变量都要初始化，且要关注初始化值的合理性！

• 验收前发现自己有大量不同步时序的情况（比如用 EN 信号高电平触发），存在隐患。这样的问题有比较 Brute Force 的解决方法：记录上一时刻的该信号，与当前时刻的信号比较，如果上一时刻为 0，当前时刻为 1，则认定为上升沿信号到来。

2. 在功能的完善、电路的设计与调试等方面做出的尝试或创新。

• 将防抖与输入模块有机结合，计算信号持续时间，而非直接处理信号波形本身，降低实现难度，有效解决抖动问题。在探测到按动信号后停止其它部分的扫描，确保多按键时只取最先按下/扫描到的按键情况。

• 为实现同步时序电路，总结出统一的方法：需要其它信号上升沿触发时，记录上一时刻的该信号，与当前时刻的信号比较，如果上一时刻为 0，当前时刻为 1，则认定为上升沿信号到来。