东南大学实验报告

数字系统课程设计

实验名称：密码锁控制器

实验时间：2022 年 7 月

实验人员：04020108 梁芊 13397849287

指导教师：翟建峰

一、功能实现

模仿密码锁的工作过程，完成密码锁的核心控制功能。

平时处于等待状态。

① 管理员可以通过设置（专用按键）更改密码。

② 如果没有预置密码，密码缺省为“0000”。

③ 用户如果需要开锁，拨动相应的开关进入输入密码状态，输入 4 位密码，按下确定键后，若密码正确，锁打开，若密码错误，将提示密码错误，要求重新输入，三次输入都错误，将发出报警信号。

④ 报警后，只有管理员作相应的处理（专用按键）才能停止报警。

⑤ 用户输入密码时，在按下确定键之前，可以通过按退格键修正，每按一次退格键消除一位密码。

⑥ 正确开锁后，用户处理完毕后，按下确定键，系统回到等待状态。

⑦ 密码正确，锁打开时，可以使用开关上方的 LED 灯配合显示效果，比如 LED 全亮等。密码错误，提示信号也可以使用 LED 进行显示。报警信号也可以使用 LED 进行显示，比如不停的闪烁等。

⑧ 数码管要充分使用，用以显示用户输入的数字等。

二、使用说明

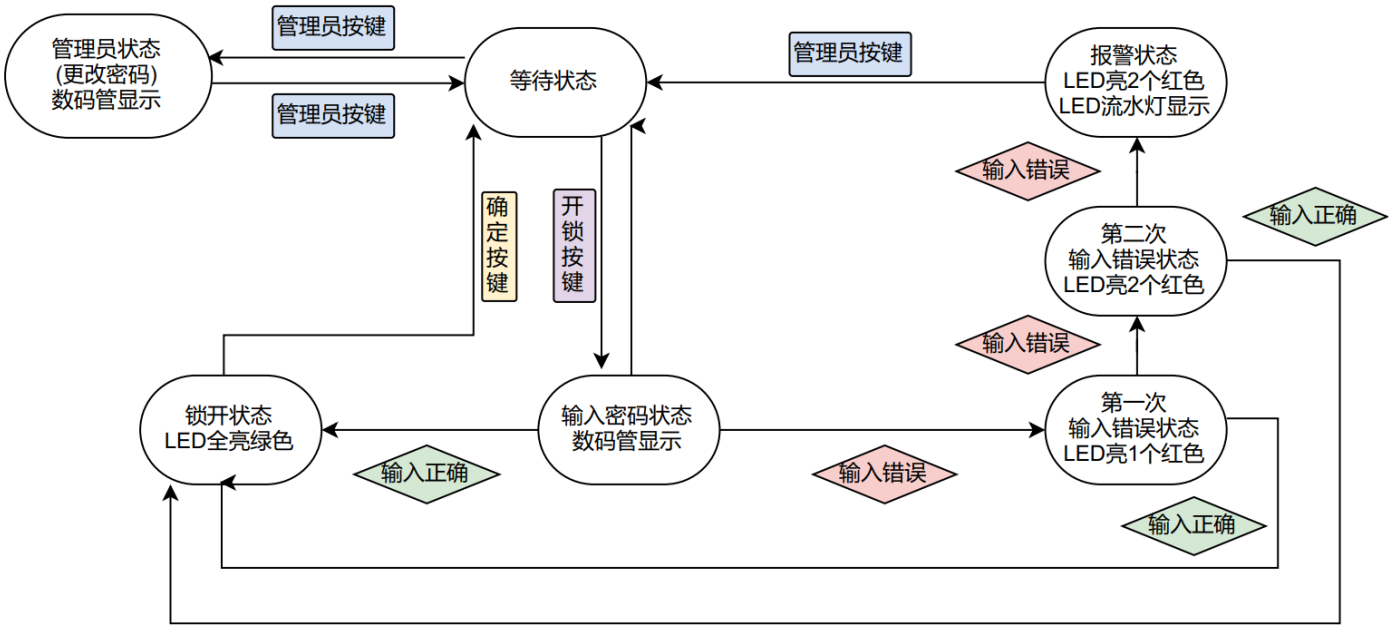


图 1 功能流程图

1. 初始状态与复位：

电源接通后电路板处于待机状态，要保持待机状态则需要将 SWITCH15 保持 0 状态。任何时候都可以拉下 SWITCH15 到 0 状态进行复位。SWITCH15 拉起为 1 状态时，进入密码锁的等待状态，此后一切操作都需要在 SWITCH15 为 1 时进行。

2. 数字的输入：

使用 SWITCH0~SWITCH3、SWITCH4~SWITCH7 进行密码的输入，其中 SWITCH4~SWITCH7 代表当前输入的是第几位数字，如 SWITCH4=1，其余为 0 时代表输入第 1 位数字；而

SWITCH0~SWITCH3 用于输入该位数字的 BCD 码。

3.密码的更改：

抬起 SWITCH9 保持状态 1，进入管理员更改密码状态。此时数码管亮起，显示初始密码 0000。用上述输入数字的方式更改密码，更改完成后，拉下 SWITCH9 为 0，退出管理员状态。

4.验证开锁：

抬起 SWITCH8 保持状态 1，进入验证开锁状态。此时数码管全灭，表示未输入任何密码。用上述输入数字的方式输入密码，输入完成后，按下 BTNC 按键代表确定键，系统会将当前输入的密码和存储的正确密码进行对比。

如果一致，2 个 3 色 LED 都亮起为绿色，此时再次按下确定键，代表处理完成，回到等待状态，密码锁上锁；

如果不一致，第一次输入错误亮起 1 个 3 色 LED 为红色，第二次输入错误亮起 2 个 3 色 LED 为红色，第三次输入错误亮起 2 个 3 色 LED 为红色，16 个单色 LED 进行流水灯显示，表示报警状态。

5.报警状态的处理：

当系统进入报警状态后，抬起 SWITCH9 保持状态 1，2 个 3 色 LED 全灭，16 个单色 LED 不再流水灯显示，系统回到管理员更改密码状态，此时可以输入新密码进行修改。

三、功能设计思路

1.数码管显示模块：

控制数码管显示时，利用视觉暂留原理，每次只亮起 1 位数字，在很短的时间内切换显示不同的数码管，从而达到 4 个数码管显示 4 个不同数字的视觉效果。为实现该功能，需要简单的分频。

2.数字输入模块：

实际上数字输入模块主要由 10 个拨动开关控制，其中 2 个用于选择输入到 save 或 code，4 个用于选择当前输入的是第几位，4 个用于表示当前输入数字的 BCD 码。通过这 10 个拨动开关不同的拨动组合情况，可以按位输入数字到 save 或 code 寄存器。

3.确定键消抖模块：

BTNC 按键连接到的是变量 yes，yes 经过消抖后为 nodou_yes。确定键消抖模块由 3 个 always 块构成，实现的功能为：当 yes 稳定在 0 或 1 超过 100ms 之后，才将 yes 的值赋给 nodou_yes。为实现该功能，需要简单的分频。

4.3 色 LED 显示模块：

该模块是核心逻辑模块。每当按下 yes，拉上管理员键、或是拉下复位键时会执行该模块。

变量 warn 存储的是当前报警状态，0 为不报警，1 为报警；变量

wrong 存储的是当前输错密码的次数。

在模块内将判断当前管理员键是否为 1，如果为 1 的话，3 色 LED 灯灭，报警状态置 0，错误次数置 0;

如果管理员键为 0，并且输入密码键为 1，模块会将当前输入的密码与存储的正确密码进行比较:

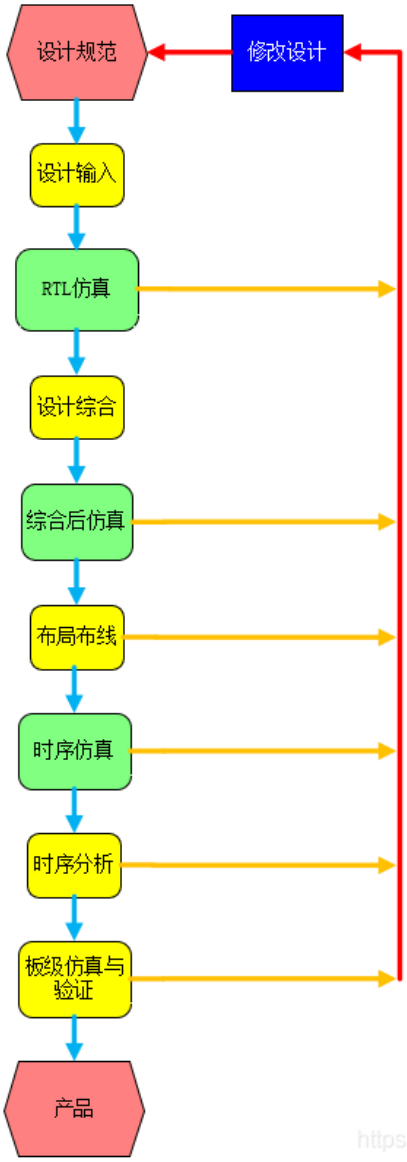
如果一致，错误次数置 0，如果当前锁为打开状态，将把锁关上，如果当前锁为关闭状态，将把锁打开。

如果不一致，将把 wrong 错误次数加 1，根据 wrong 的取值不同，2 个 3 色 led 灯的亮起情况不同。如果当前错误次数为 3，将报警状态 warn 置 1.

5.流水灯显示模块:

流水灯显示模块包含简单的分频，从而达到肉眼可见的 LED 灯依次移动的效果。当报警状态 warn 为 1 时，会进行移位的操作。

四、设计流程



五、验证

1.仿真验证：

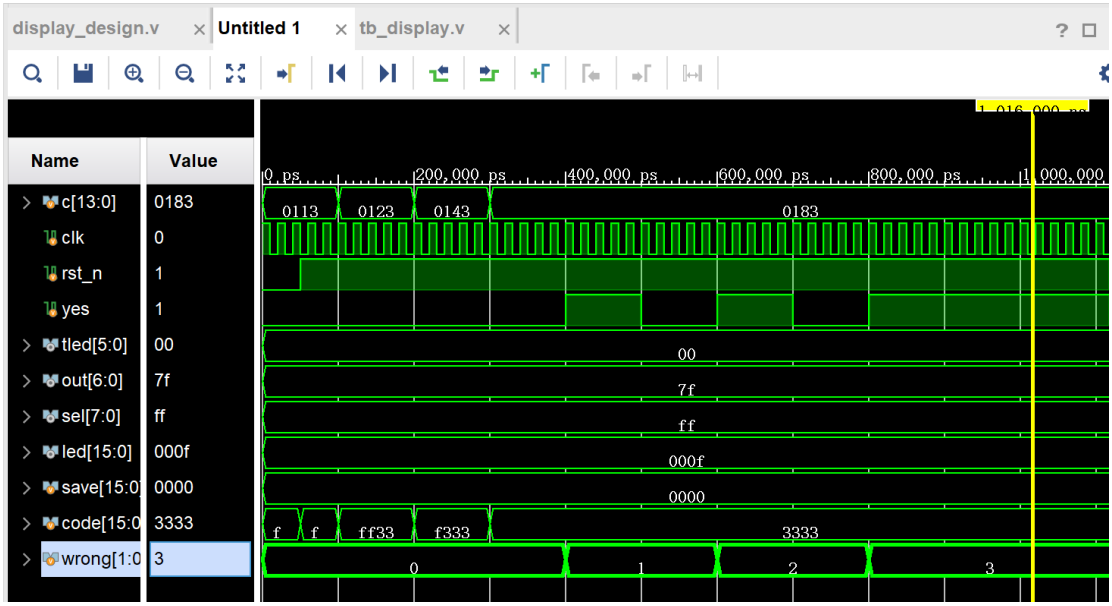


图 2 仿真逻辑图

以该仿真为例：首先输入密码，c 改变，观察到 code 随之改变，然后模拟按下确认键，yes 改变，观察到 wrong 错误次数随之改变。

其余情况不在此一一赘述，经过仿真验证功能逻辑设计无误。

2.实物验证：

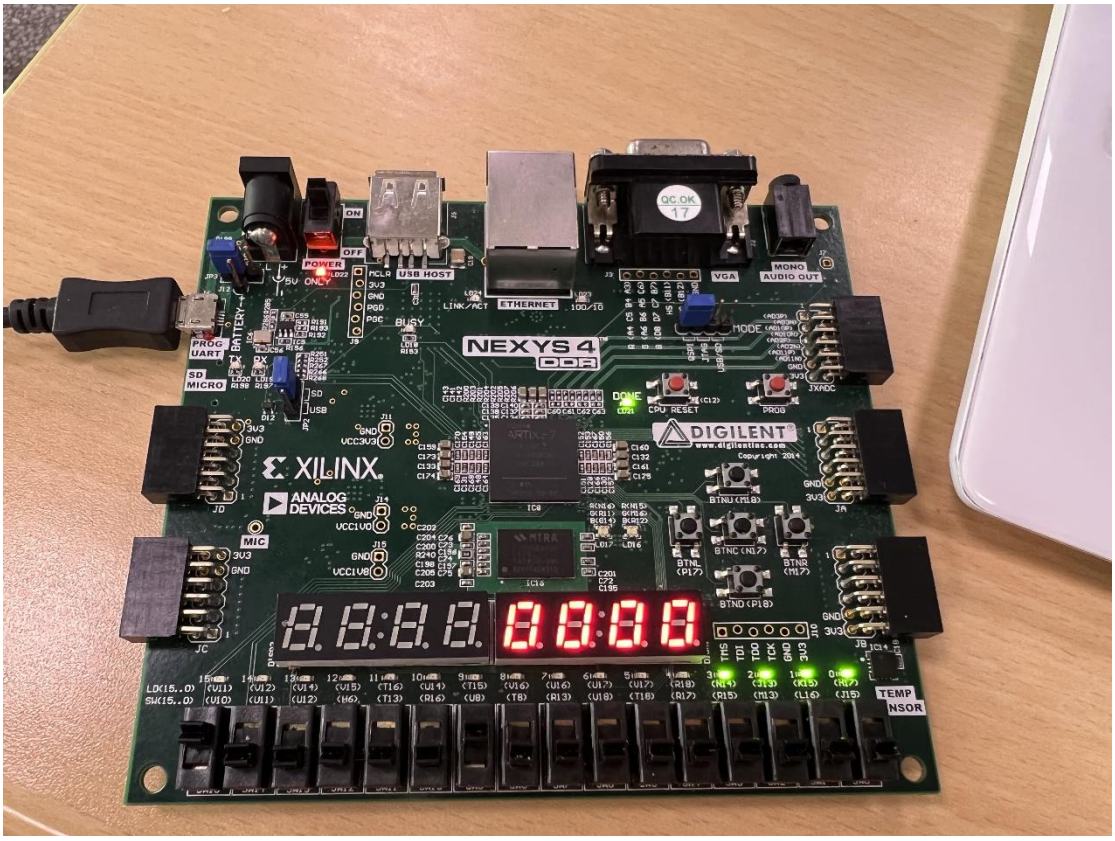


图 3 抬起管理员键 显示初始密码 0000

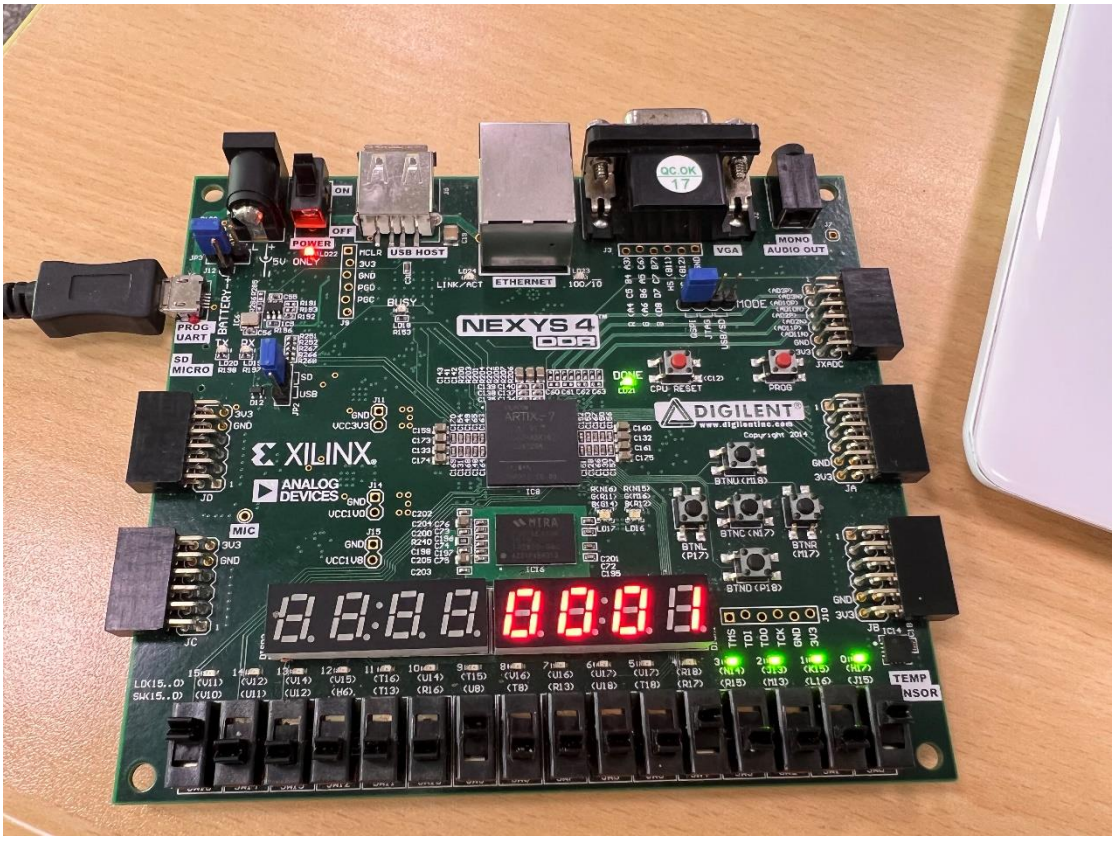


图 4 输入第 1 位密码为 0001 是 1 的 BCD 码

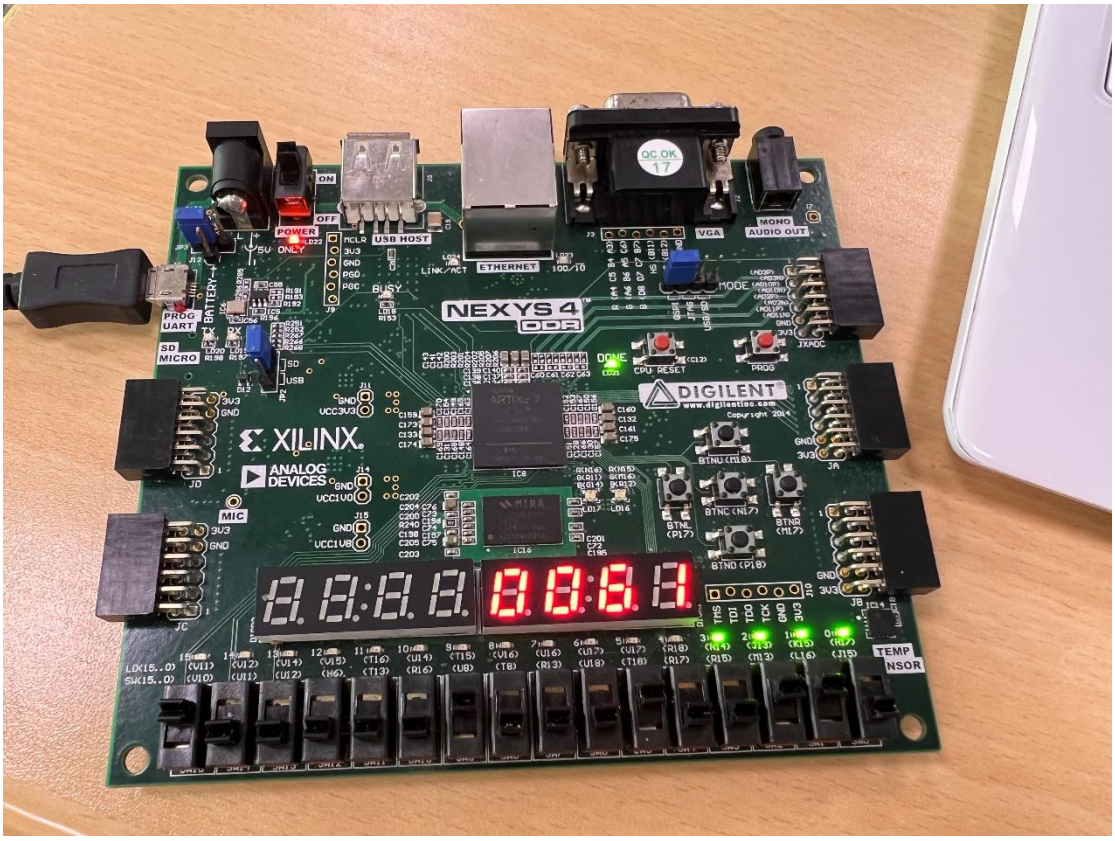


图 5 输入第 2 位密码为 0110 是 6 的 BCD 码（最终设置密码为 0001）

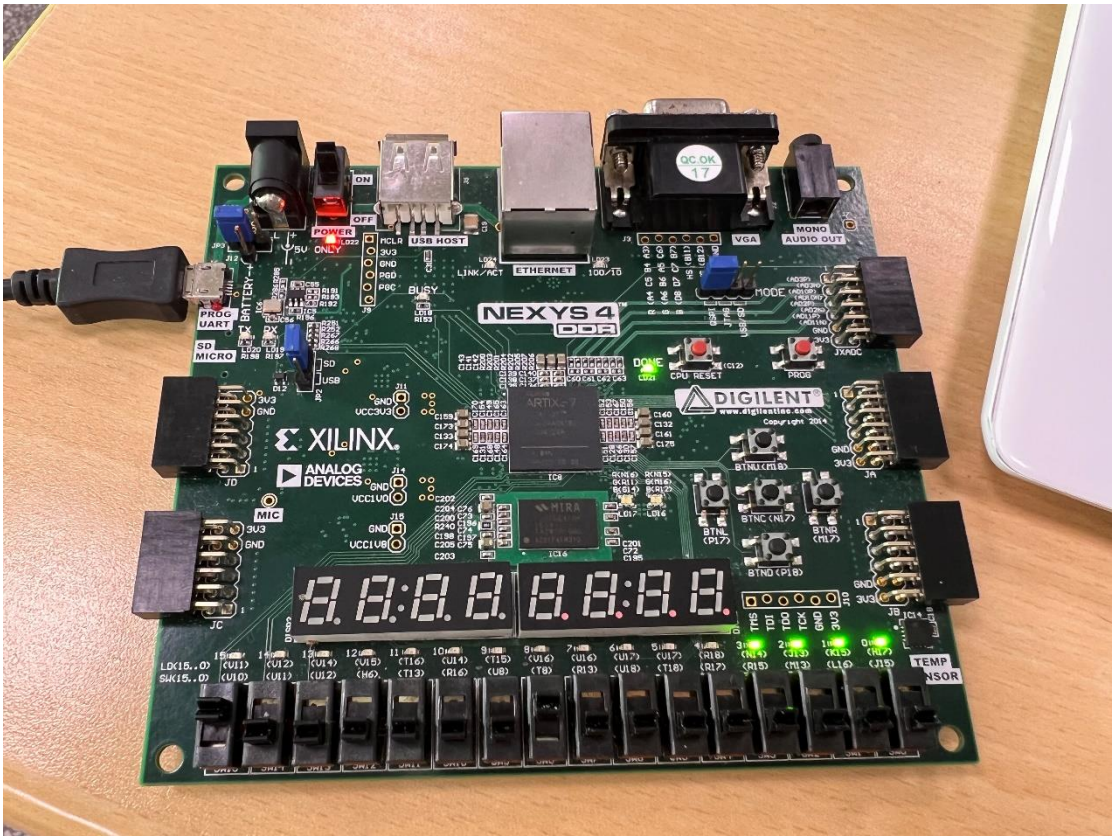


图 6 输入密码状态 此时数码管全灭代表未输入任何密码

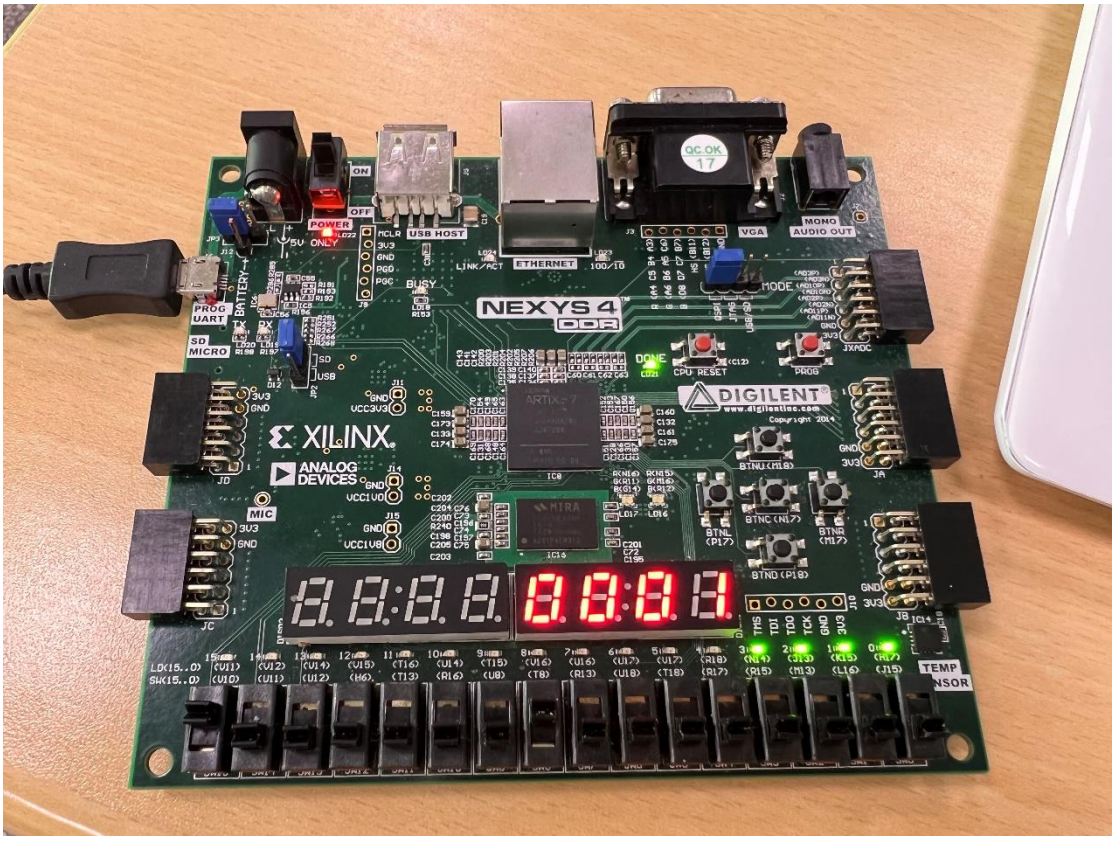


图 7 输入密码为 0001

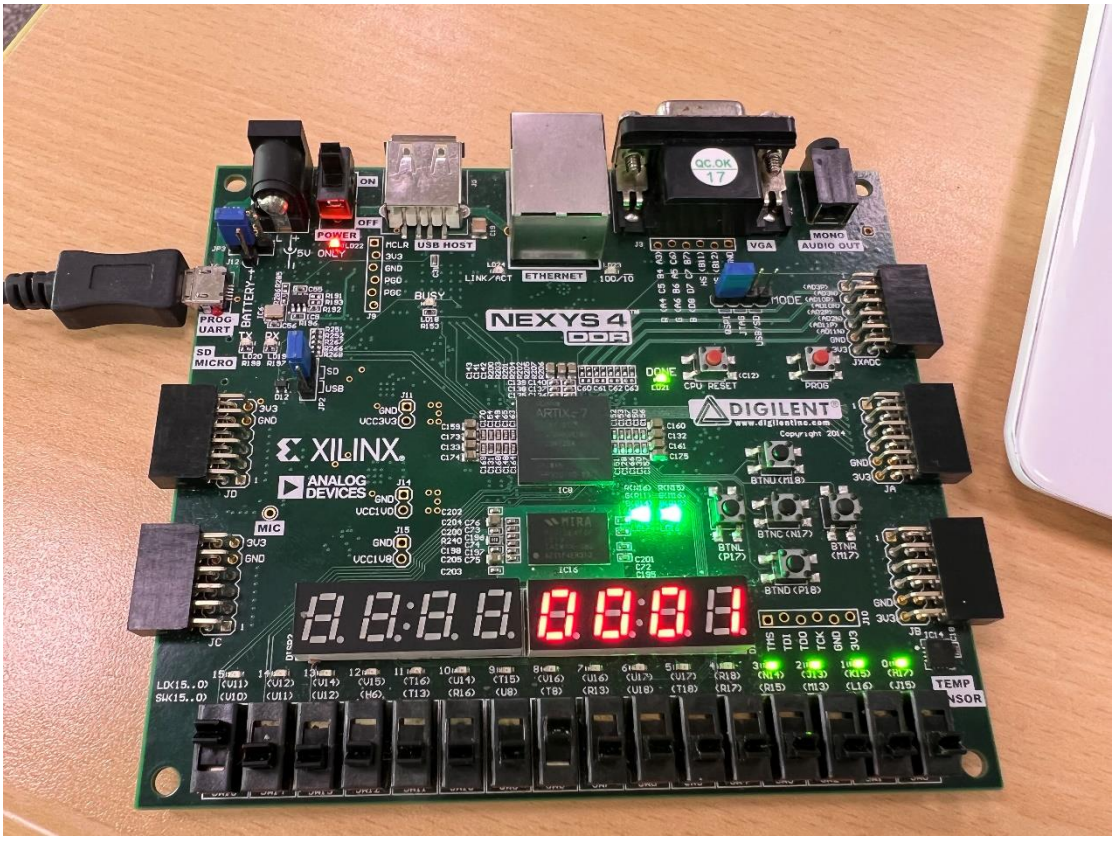


图 8 按下确定键 密码锁打开 LED 全亮绿色

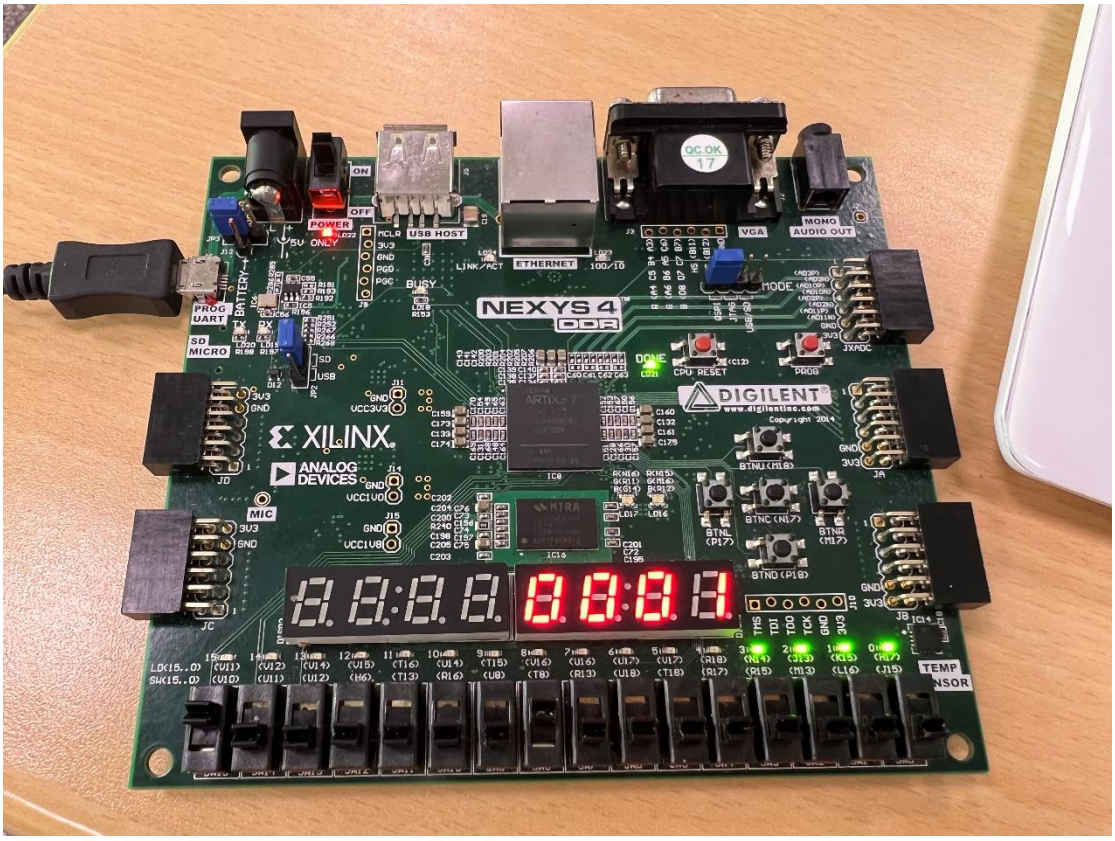


图 9 再次按下确定键 密码锁锁上

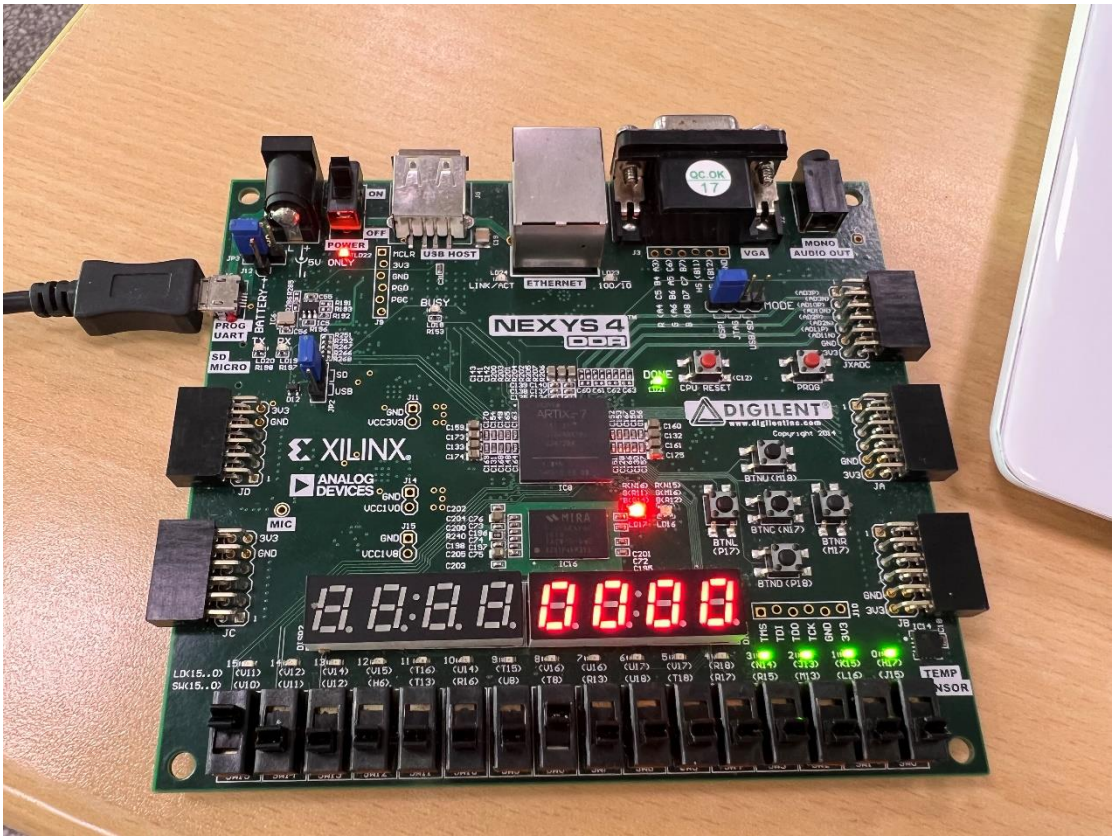


图 10 输入错误密码 0000 按下确定键 亮 1 个红色 LED 代表输入错误 1 次

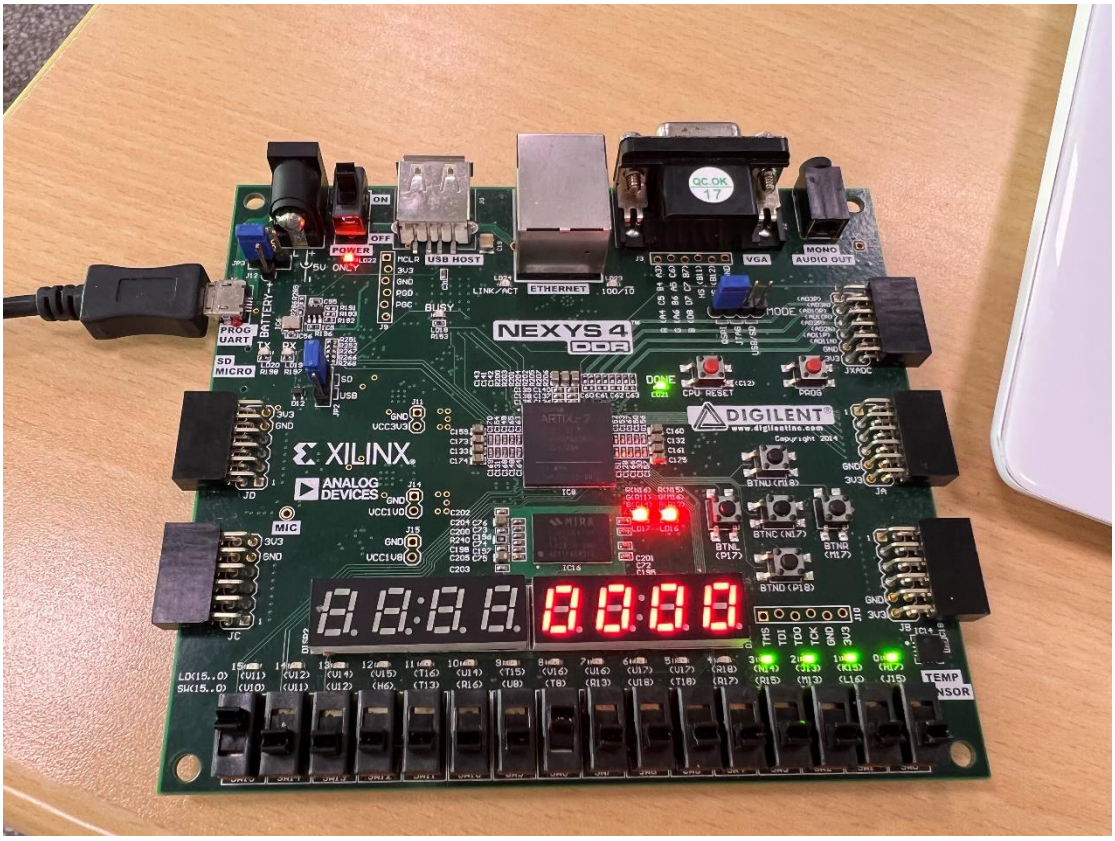


图 11 再次输入错误密码 0000 按下确定键 亮 2 个红色 LED 代表输错 2 次

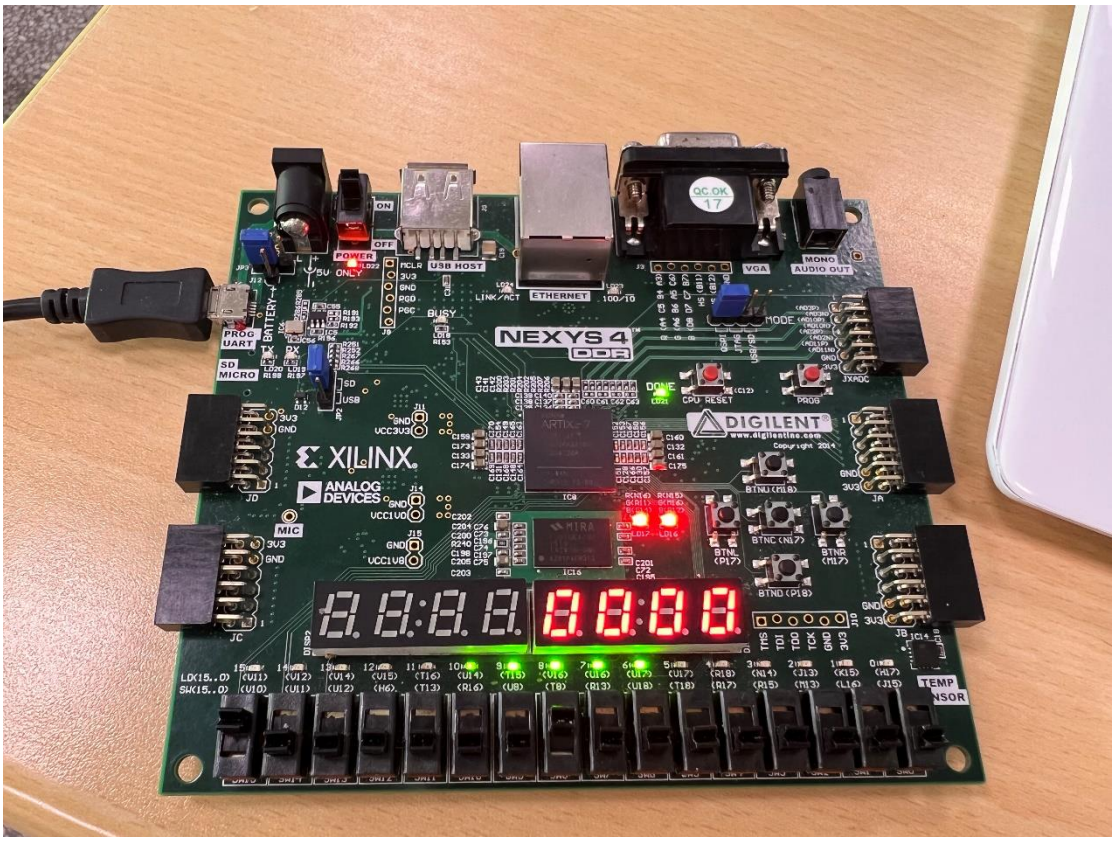


图 12 第 3 次输错密码 16 个 LED 流水灯显示

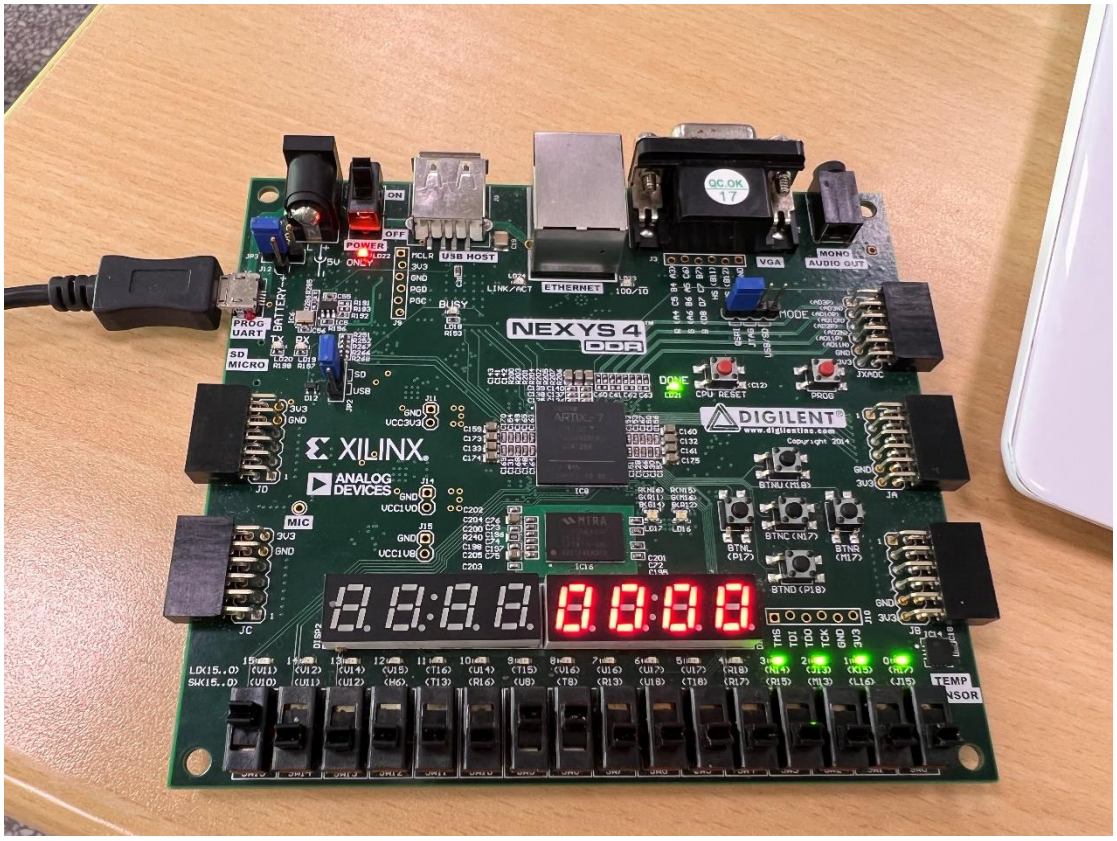


图 13 拉上管理员键 解除报警 回到等待状态

六、学习中遇到的问题与经验总结

1. always 块注意要点：

①不能在不同的 always 块内为同一个变量赋值

②不能在同一个 always 块内同时使用阻塞赋值(=)和非阻塞赋值(<=)

③在使用 always 块描述组合逻辑（电平触发）时使用阻塞赋值(=)；使用 always 块描述时序逻辑（边沿触发）时使用非阻塞赋值(=)

④ 任何在 always 块内被赋值的变量都必须是寄存器型(reg)

⑤ always 的敏感列表中可以同时包括多个电平敏感事件，也可以包括多个边沿敏感事件，但不能同时有电平和边沿敏感事件

⑥在敏感列表中,同时包括一个信号的上升沿敏感事件和下降沿敏感事件是不容许的，因为这两个事件可以合并为一个电平事件

⑦可以使用*替换敏感列表，表示缺省，编译器会根据 always 块内部的内容自动识别敏感变量（不是会直接全都加上的）

⑧如果没有了敏感列表，则为零延迟，那么就会不断的触发，如 always #10 clk = ~clk；这种没有敏感列表的显示延迟，是不能综合的，只能用于仿真

2. initial 块注意要点：

initial 从其执行路径的属性来看，它不应该存在于硬件设计代码中，它本身不可综合，对于描述电路没有任何帮助。

initial 就是为了测试而生的，由于测试需要按照时间顺序的习惯即软件方式来完成，所以 initial 便可以实现这一要求。

3.project 中各个文件含义：

project_name.cache：Vivado 软件的运行缓存

project_name.hw：所有波形文件

project_name.ip_user_files：用户关于 IP 的文件

project_name.runs：编译与综合结果，.\impl_1 文件夹中的.bin 和 .bit 即为编译生成的可执行文件

project_name.sdk：SDK 环境代码，一般是 ZYNQ 设计中关于 PS 端的代码

project_name.sim：仿真结果

project_name.srcs：所有用户编写的源码、仿真文件与约束文件

project_name.tmp：自制 IP 核时的临时工程文件夹，IP 核设计完成后会自动清理

project_name.xpr：Vivado 工程启动文件

4.wire 型变量和 reg 型变量的辨析：

Verilog HDL 中的变量可以定义为 wire 型和 reg 型，这两种类型的变量在定义时要设置位宽，缺省为 1 位，变量的每一位可以取 0、1、x、z，其中 x 代表未预置初始状态，z 代表高阻状态。

reg 相当于存储单元，wire 型相当于物理连线，即 reg 型变量保持最后一次的赋值，而 wire 型变量需要持续的驱动。

①若变量放在 begin…end 内，声明为 reg 型；否则，声明为 wire 型

②在 always 块中的变量，只能是 reg 型；使用 wire 型变量时，必须

搭配 assign；input、output、inout 声明的变量，默认都是 wire 型

③在设计中，一般来说我们并不知道输入信号是来自上一级寄存器的输出还是组合逻辑的输出，那么对于本级而言就是一根导线，也就是 wire 型。而输出信号则设计者决定是寄存器输出还是组合逻辑输出，wire 型和 reg 型都可以，但通常整个设计的外部输出（即最顶层模块的输出）是寄存器输出，这样电路比较稳定。

5.数码管的显示：

更换显示哪一个数码管的频率要合理，如果频率过高，数码管的显示亮度不够；如果频率过低人眼会看出差别，视觉效果为一个数码管接一个数码管地闪烁。一般而言数码管闪烁频率在 60Hz~100Hz 之间比较合理。

6.仿真时出现不定态 X：

表示 Unknown，仿真发生了不能解决的逻辑冲突。检查发现是我没有在仿真文件中为该变量赋初值，赋值后解决。

7.按下一次确认键显示输错多次：

这是由于确认键未消抖造成的。虽然在操作者看来只按下了一次按键，但是操作板十分灵敏，微小的抖动都会使其判断为按下抬起，如此反复几次。在本系统中，只有按键稳定在 1 或 0 持续 100ms 之后，才会视为按下一次按键，达到消抖的目的。