基于元件/EDA 的密码锁

软 73 沈冠霖 2017013569

1.引言

最近因退不出来押金的 ofo 公司在小黄车上使用过一种密码锁：可以通过扫码获取小黄车的内置密码，输入正确的四位密码就能开锁。这种锁在旅行箱等器件上也有用武之地。这次我就来大致复现一把这样的锁。我既自己用教材上的中规模集成电路和门电路设计了电路图，又用 verilog 编程实现了功能，生成了电路图，并且进行了一定的比较。

2.需求设置

这个锁需要完成以下功能：有一个 1-9 的输入键盘，还有删除，清空两个按键，输入需要实现消抖。还有一个 LED 灯显示已经输入的密码。如果已经输入了四位数，则和内部寄存器的密码比对。如果一致，则播放开锁音效，开锁。否则，播放密码错误音效。如果连续三次密码错误，则让锁不能再开启。

3.模块分析

这个锁一共分为个模块：1.输入转换模块：负责读取键盘的输入信息，并且进行消抖；2.存储输入模块：负责识别输入到哪一位了，并且将输入的数字存到对应的寄存器里，也负责完成清空输入和删除一个输入的功能；3.显示模块：显示已经输入的数字，并且由于只能输入 1-9，还要进行灭零显示；4.密码存储模块：一个出厂就设置好的密码，存储在一组寄存器里；5：比较模块：在输入四位数字后，自动将输入数字和密码比较；6：硬件部分：如果比较成功，则播放成功音效，开锁，否则播放失败音效。如果连续 k 次开锁失败，则不能开锁。

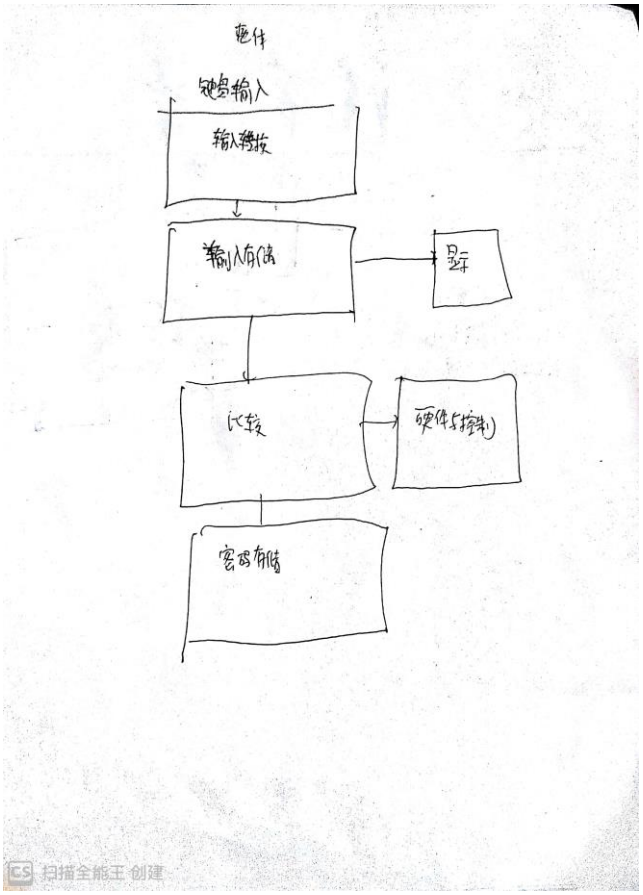

图 1.1 总体模块设计

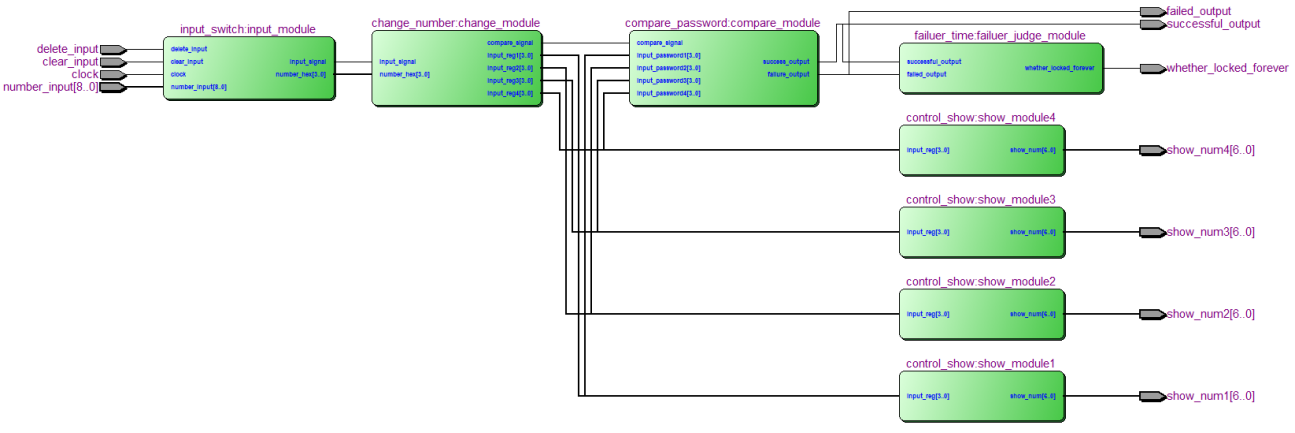

图 1.2 EDA 程序自动生成的模块（四个 show 是同样的，只是分别显示四位）

4.模块设计

4.1 输入转换模块

输入转换的思路是用一个 10-2 进制转换器，将输入的键盘信息转化成二进制数据备用，同时还要识别是否有输入。防止消抖的办法是使用一个低频率（比如 100HZ）的时钟，每到时钟下降沿才判断进行相应的操作。

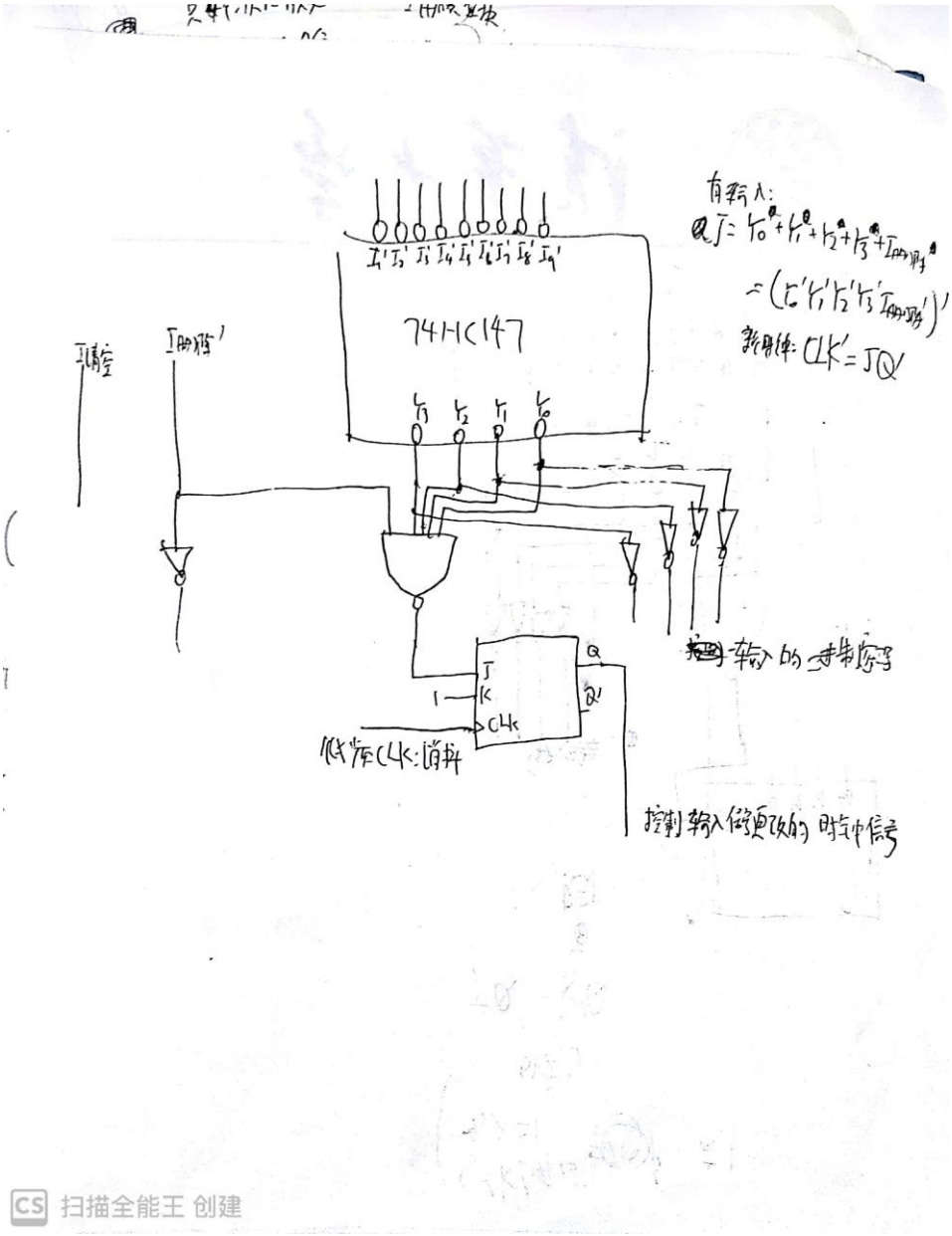

图 2.1 自己设计的输入转换模块

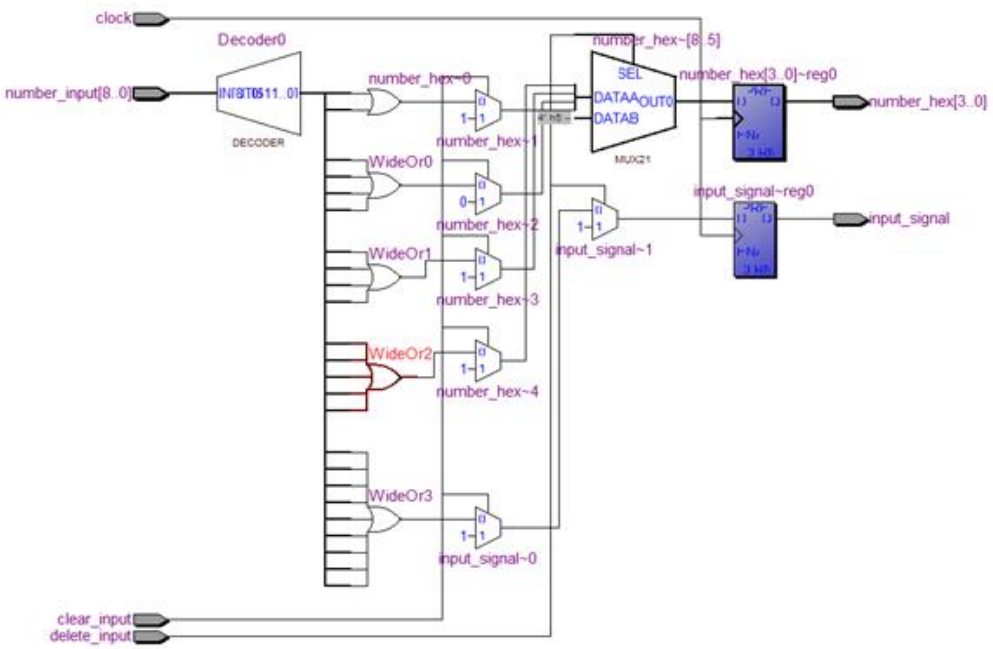

图 2.2 用 EDA 实现输入转换模块

4.2 存储输入模块

这个模块的核心是“知道你输入的是第几位”。我的想法是用四个固定的寄存器来存储输入的 1,2,3,4 位，如果没输入就存储 0，已经输入就存储输入的数字。

而用一个移位寄存器来存储输入了几位，如果输入了新的一位，寄存器就左移一位，低位标 1；如果删除了一位，寄存器就右移一位，高位标 0；清空则全部置 0；如果寄存器是 1111，则自动进入比较模块。

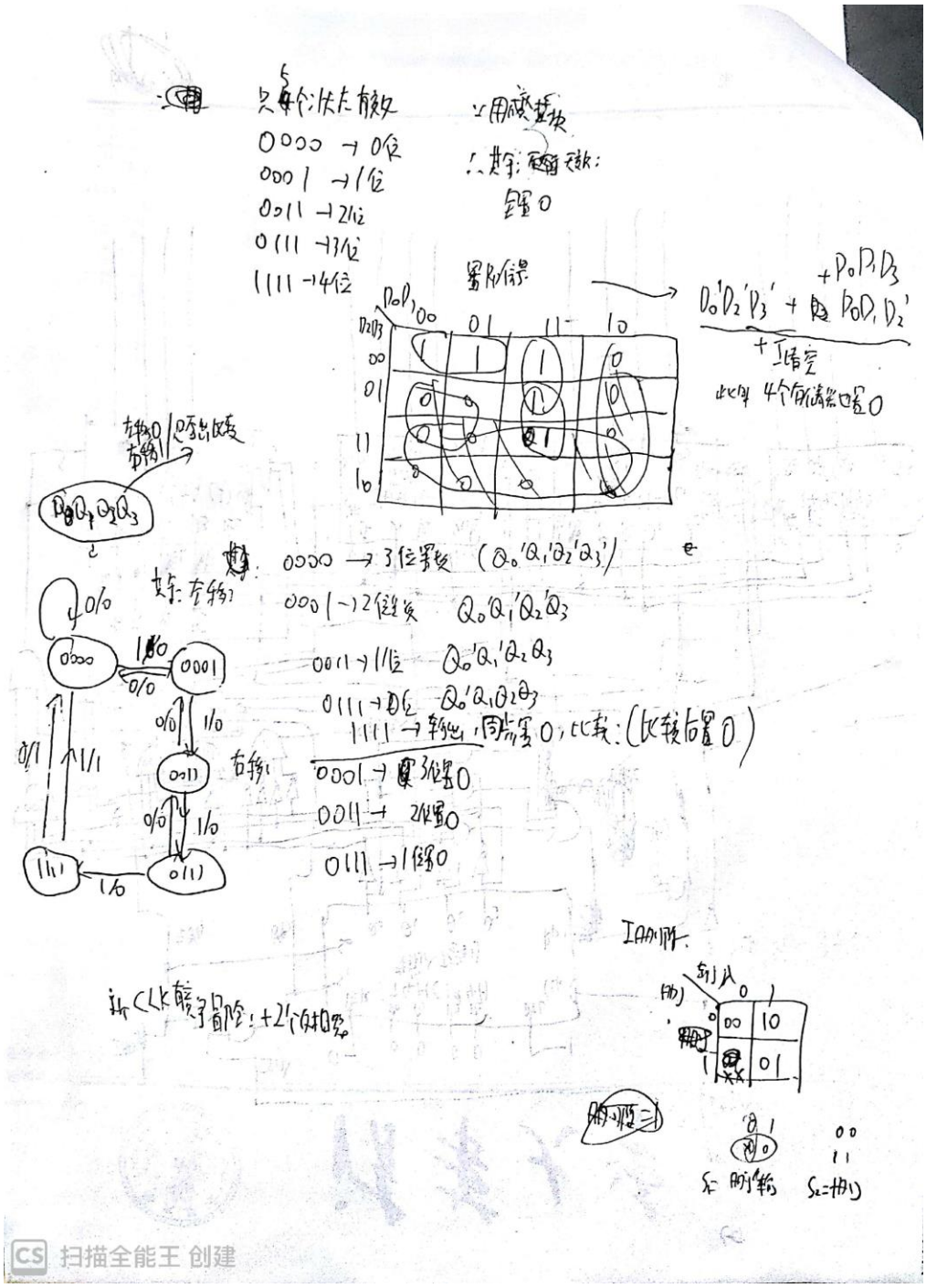

图 3.1 我对存储输入模块的分析

其中，控制位数的寄存器有效位数只有 0000,0001,0011,0111,1111 五种，其余都要保证能自启动---异步置 0

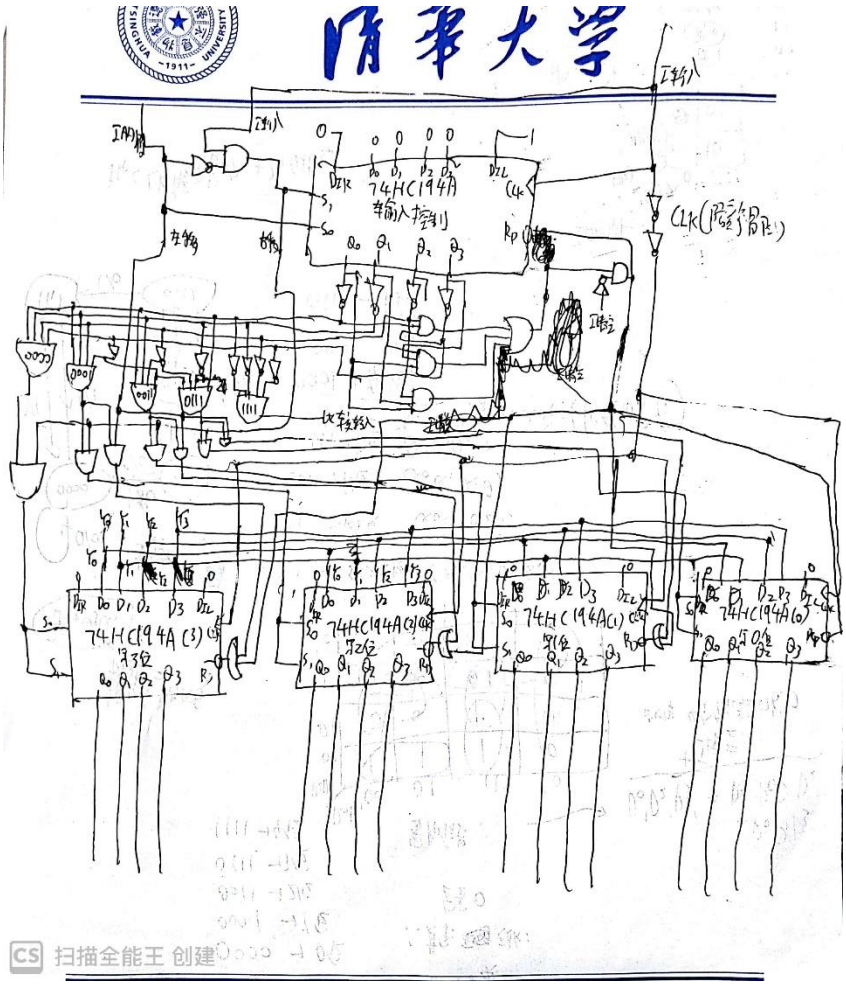

图 3.2 自己设计的存储输入模块

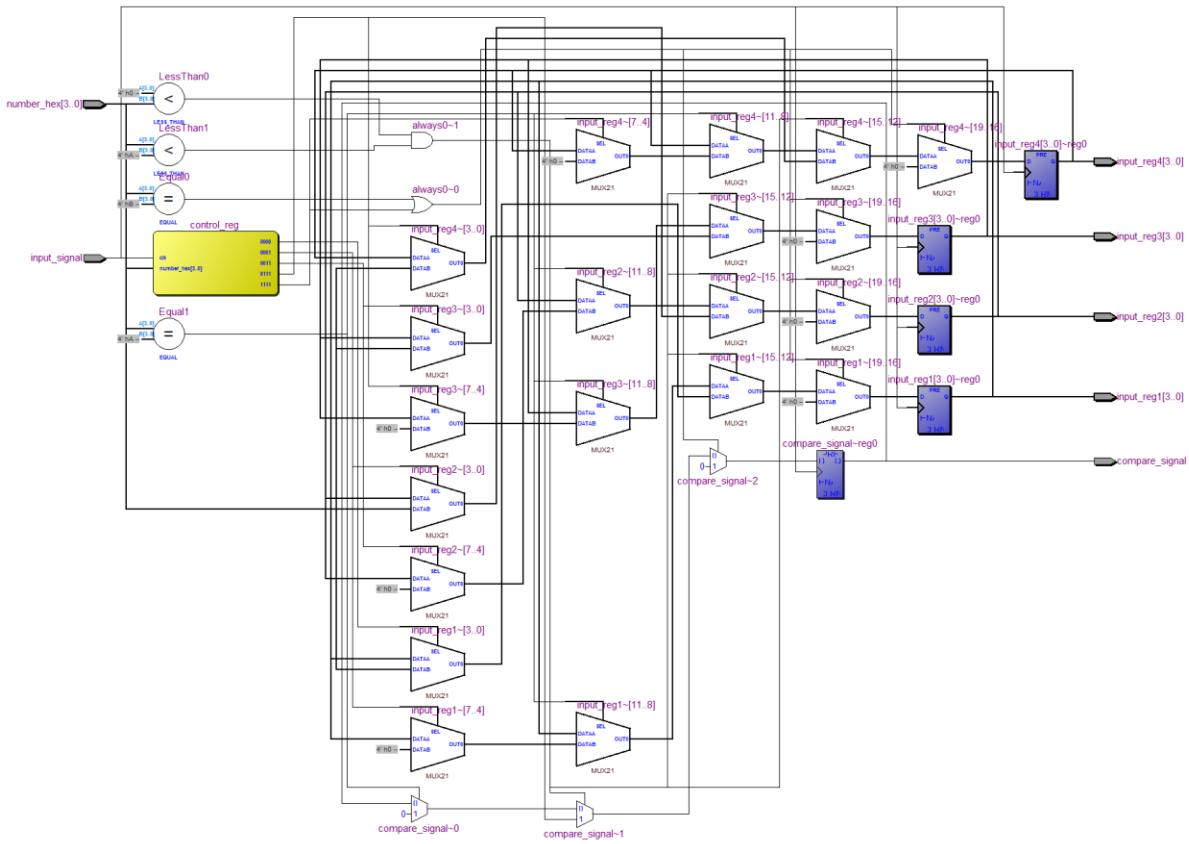

图 3.3 EDA 设计的存储输入模块

4.3 显示模块

这个模块就是分别将四个寄存器连接四个数码管控制电路，并且因为输入的密码是从左到右的，因此从最右到最左执行灭零输入。

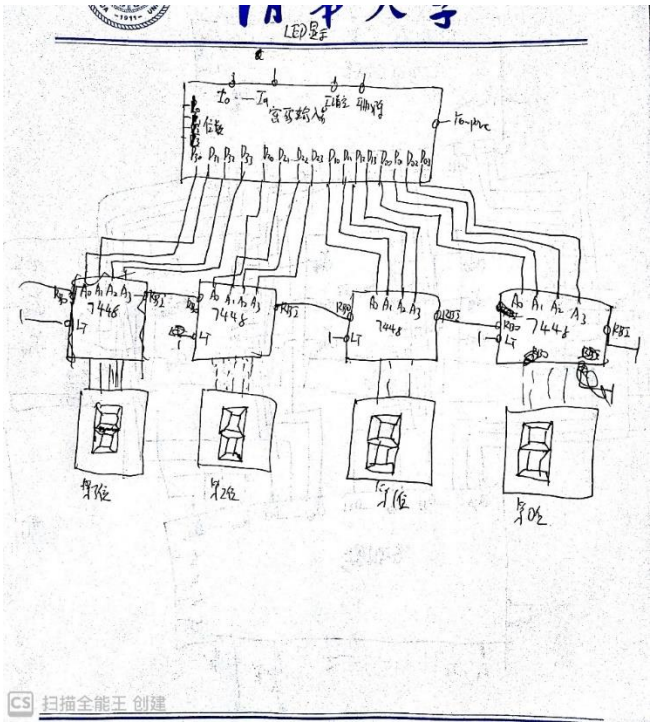

图 4.1 自己设计的显示模块

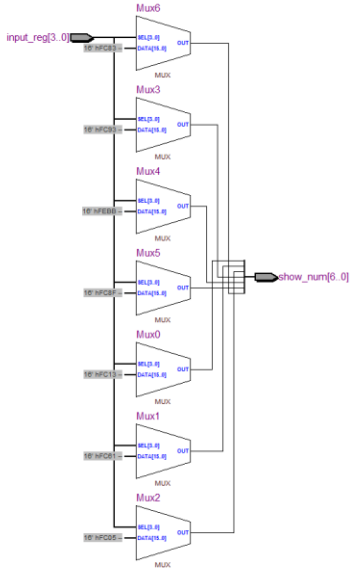

图 4.2 EDA 的一个数码管显示模块（输出的是数码管显示的控制数据）

4.4 密码存储和比较模块

这个模块就是把 4 个 4 位数据比较器连到一起，把比较结果传输给对应硬件部分。因为正确密码是出厂设计好的，所以只需要存储在一个存储器里即可。

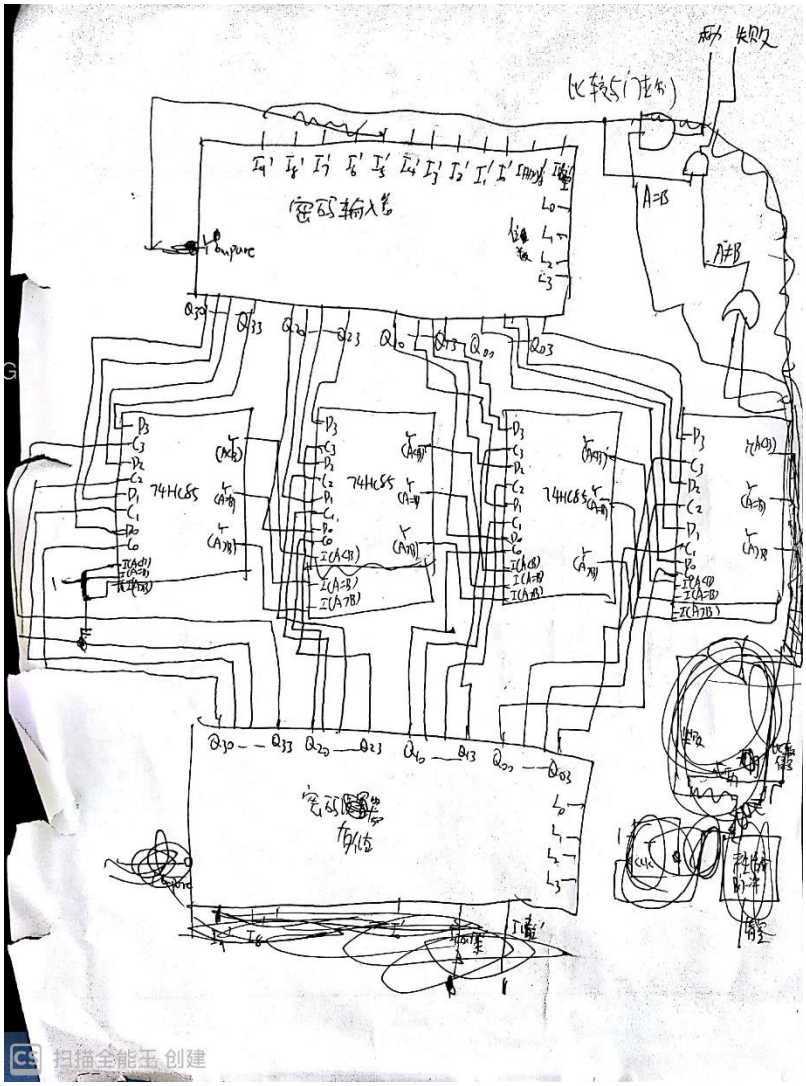

图 5.1 自己设计的密码比较器

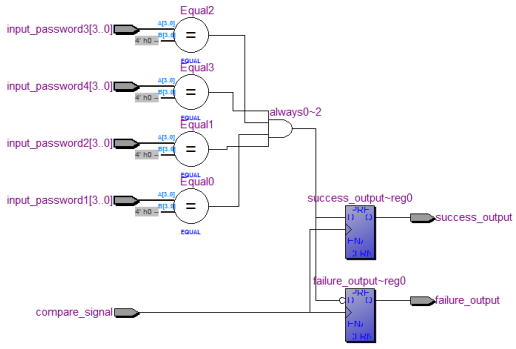

图 5.2 EDA 实现的密码比较器

4.5 硬件部分

硬件的控制部分：用开锁成功信号的脉冲控制成功蜂鸣器，开锁，还有清空数据；用开锁失败信号的脉冲，控制失败蜂鸣器，还有清空数据。如果连续开锁失败 k 次，则不能再开锁

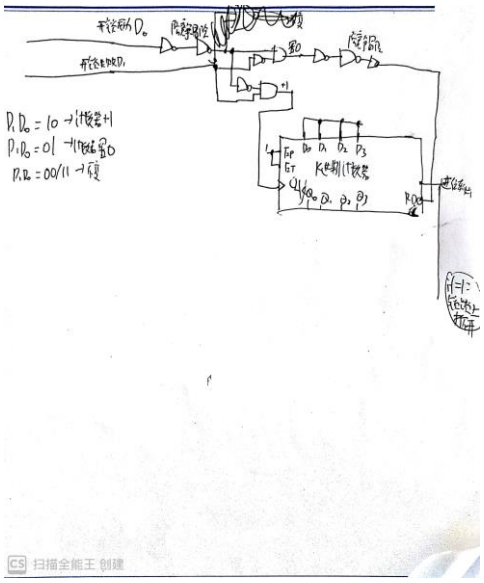

图 6.1 自己设计的连续开锁失败 k 次则不能开锁，用一个 k 进制计数器实现。

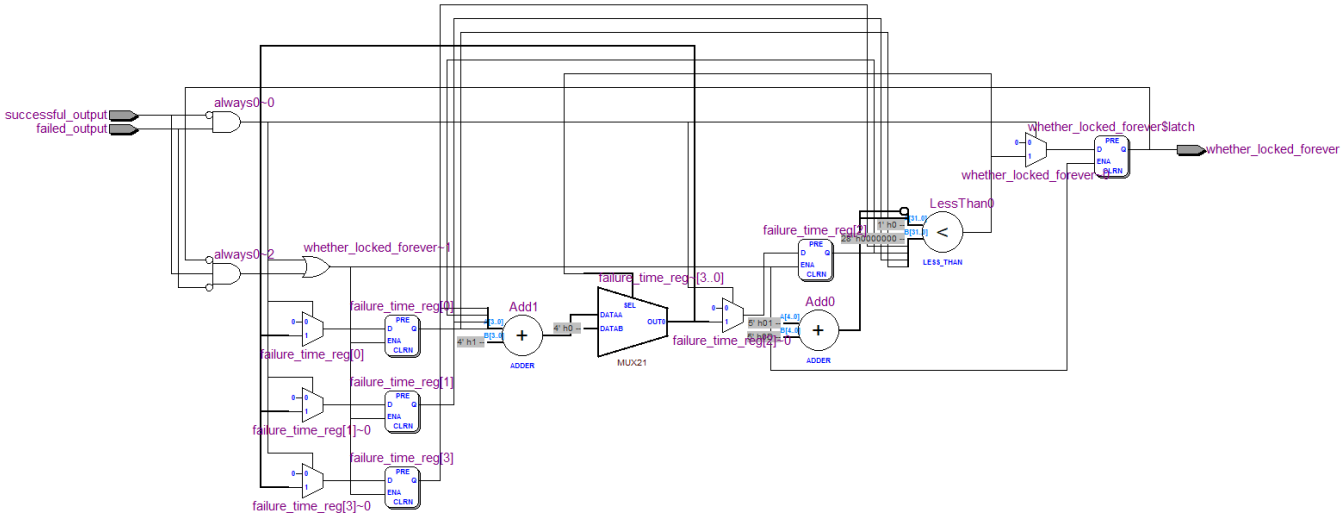

图 6.2 EDA 生成的电路

硬件需要 1 把可以输入信号就打开的锁，需要 2 个不同音效的蜂鸣器（或者一个音乐芯片，可以采用 QGME0001）

图 6.3 蜂鸣器和 QGME0001 音乐芯片

5.比较与总结

由于我同时自己用元器件和 verilog 语言设计了这个电路，因此，我进行了对比。

用元器件有一定的优点：对于较小规模的电路，成本相对较低。比如这个电路，使用元器件设计，只需要 5 个 74HC194A,一个 74HC147，一个 JK 触发器，4 个 7448+数码管，还有四个四位比较器。它们的成本低于一个 FPGA 板子。

但是，用元器件设计这种有一定规模的电路，则比较麻烦。首先，用元器件设计，可能会有各种竞争-冒险现象，异常情况等，这是小规模电路所遇不到的。而 verilog 则用程序帮你规避了这样的问题。其次，用元器件设计的话，因为我了解的元器件有限，选择的元器件不是很适合，因此会有很多冗余状态出现，而 verilog 能规避这种问题。最终，元器件电路实际修改很复杂，而且难以复用，扩展。而 verilog 则复用性很高，易修改和扩展。

这次作业，我既锻炼了 verilog 设计能力，也熟悉了使用译码器，比较器等组合逻辑电路模块，以及使用寄存器，触发器，移位寄存器等时序逻辑电路模块，还有组合，时序逻辑电路的分析，设计过程。还有，我锻炼了如何消除竞争-冒险现象和处理一定的冗余状态。然而这个设计可能还有许多我未发现的问题，以及其从设计到实物还有很多事情没有做，希望以后能不断提升能力，以进一步用硬件做出可用的产品。