寝室楼底自助外卖寄存机

软件92 诸葛向文 2018010115

引言

打包形式是最早出现的外卖形式，虽然古老，却延续至今。随着电话、手机、网络的普及，外卖行业得到迅速的发展。数据显示，从2016年的0.63亿人到2019年的4.23亿人，中国在线外卖用户规模逐年增长。预计2020年中国在线外卖用户规模继续增长至4.56亿人。

根据笔者观察，用外卖就餐而非前往食堂的现象在清华大学也是稀松平常，然而这也催生了很多问题。一方面，外卖放在楼下的柜子上，风沙比较大很可能不卫生；另一方面，外卖露天寄放保温效果也欠佳，如果同学们不及时下床下楼，外卖的口感会大打折扣。

更为关键的是，由于同学们点外卖的热情持续高涨，外卖订单数过多，以及楼底外卖柜并没有严格地管理。这使得寝室楼下经常出现同学们外卖被偷窃或者误拿的情况。甚至在今年的11月份，紫荆4号楼“外卖小偷”事件在全网引起了关注。幸运的是，笔者的外卖还未遭受如此厄运，但是设身处地地思考一番，无论对方有意无意，属于自己的美食被别人拿走，而自己却要饿肚子总是令人非常恼火的。

据此，笔者试着构思了寝室楼底自助外卖寄存器。当勤劳的配送小哥将外卖送达之后，可以将外卖锁入柜中并设置密码，购买此次外卖的同学可以凭事先约定好的密码开箱得到食物。这样的自助外卖售货机在避免了同学们误拿外卖的同时，也在一定程度上解决了露天存放外卖不卫生、保温效果差的情况。希望这是一个实用的设计。

功能介绍

主体部分类似于自助取快递机器，主要的功能在于配送小哥寄存外卖和同学们取走外卖。

1. 当外卖小哥抵达寝室楼底，点击寄存功能键，可以查询外卖寄存机是否尚存空位：
    - 如果存在空位，则外卖小哥可以寄存外卖并设置密码。
    - 如果不存在空位，那么非常遗憾外卖小哥只能将外卖寄存在露天的柜子上抑或是暖气片上。
2. 当同学们抵达寝室楼底取外卖的时候，点击取货功能键，可以查询外卖寄存机中是否存有外卖
    - 如果存有外卖，则同学们可以输入密码取餐。

        如果输入的寄存位置和密码匹配，则取货成功；

        否则，同学可以再试一次密码或者放弃取外卖。
    - 如果不存有外卖，那么非常遗憾，请同学们再等一会，外卖一定马上就到。

设计思路

出于简化问题的考虑，暂定本设计中有以下的假设:

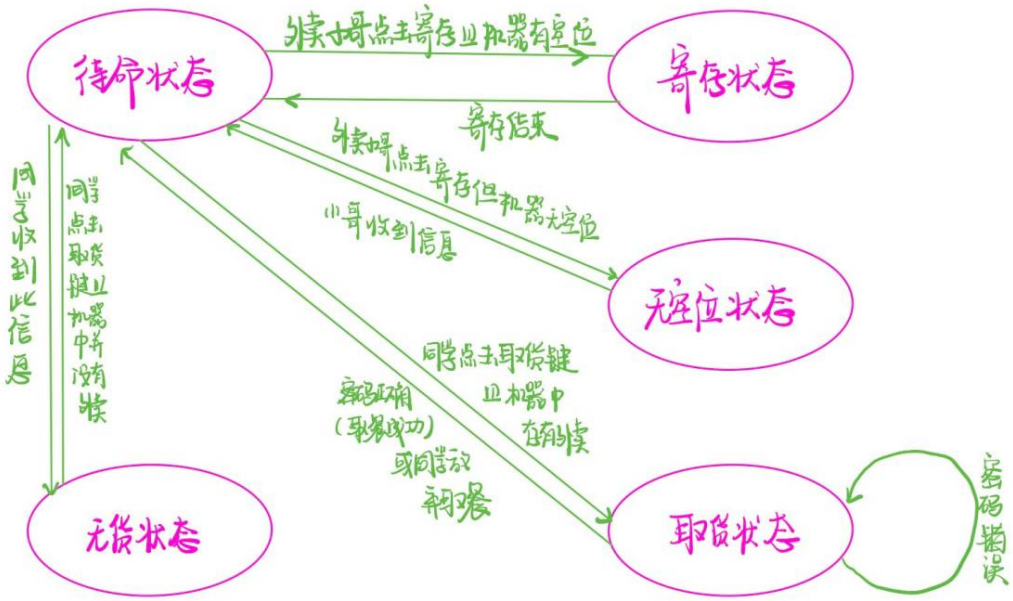
(1)寄存器中有四个储物柜;

(2)每个储物柜的密码输入是四个按钮，即为4位2进制数;

(3)功能选择键是两个按钮，即两位二进制;

(4)储物柜选择键是两个按钮，即两位二进制。

1. 本外卖售货机的主体部分应当是一个有限的状态机。具体的状态及其转换关系可以见下图:
   有效的工作状态为五个：待命状态、寄存状态、无空位状态、取货状态和无货状态。
   设计其余无效状态可以自动转化为待命状态以实现电路的自启动。



2. 当同学们输入柜子的号码和密码时，就需要一个四选一的数据选择器将储物柜对应的密码调出。
   （举例：当同学选择柜号00时，就调出00号柜子的密码XXXX。）

3. 同时，系统还需要一个4位的数字比较器，以判断同学输入的密码和同学们留下的密码是否相等。当且仅当这两者相等时，数字比较器模块输出1，即密码正确; 其余情况一律输出0，即密码错误。
   （举例：输入0000和0000，输出1； 输入0000和0001，输出0）

4. 利用一个三位的寄存器来存储当前有几个柜子被占用，并以此作为判断本外卖自助寄存机是否无空位或是否有外卖的标准。（000：无外卖且未满载；001~011有外卖且未满载；100有外卖且满载）

具体实现

1. 状态机部分的实现。

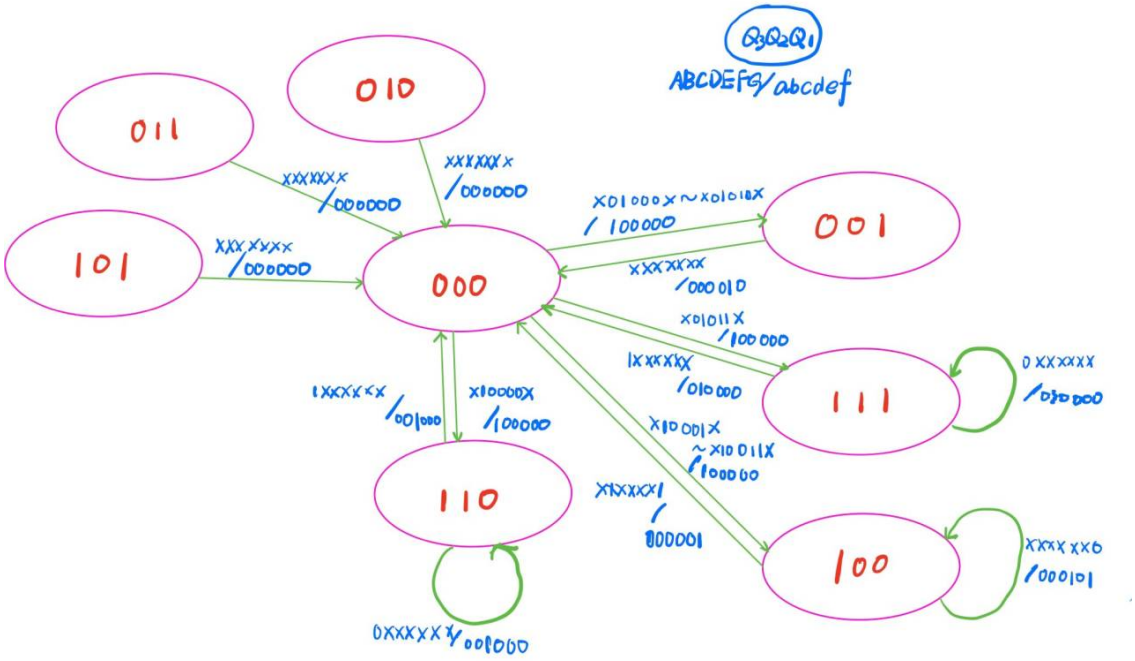
   首先获取时序逻辑标准的状态转化图。（如下图）

   参数解释：

   （1）$Q_3Q_2Q_1$: 000为待命状态(WAIT)，001寄存状态(PUSH)，111无空位状态(OVERLOAD)，100取货状态POP)，110无货状态(NOFOOD)，其余均为未定义状态。

   （2）$ABCDEFG$: 六个输入，A，getinfo(是否收到信息，1代表收到)；BC,func(选择存货还是取货，01存货，10取货)；DEF, size(当前有几个寄存器被占用,为000~100)；G,iscorrect，密码是否正确（1正确,0错误）。

   （3）$abcdfg$: 六个输出，分别代表六个指示灯waitled, overloadled, nofoodled, keyerrorled, pushled, popled。（1亮0灭）



采用verilog代码实现上述的状态转换图（两段状态机）：用两个always模块来描述状态机，其中一个always模块采用同步时序描述状态转移；另一个模块采用组合逻辑判断状态转移条件，描述状态转移规律以及输出。

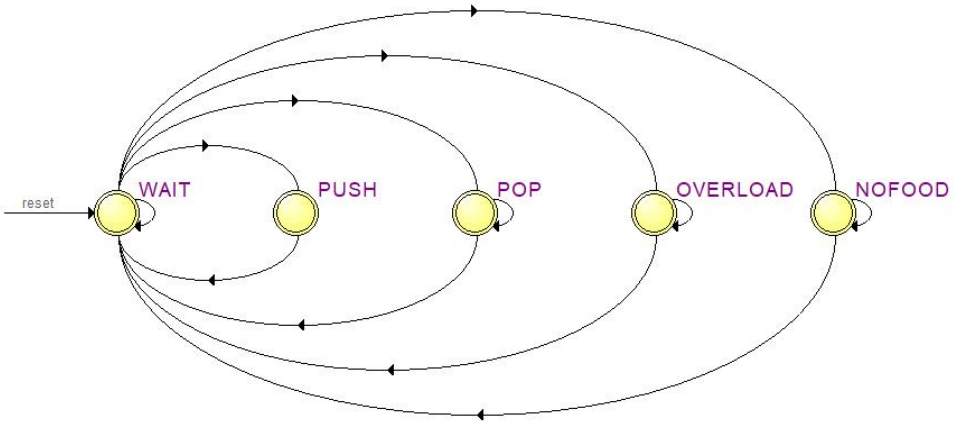
```
always@(posedge clk or posedge reset)
begin
    if(reset)
        curstate <= WAIT;
    else
        curstate <= nextstate;
end

always@ (curstate)
begin
    case(curstate)
        WAIT:
            begin
                case(func)
                    2'b01://试图存入外卖
```

```
                    begin
                        if(size < 4)
                            begin
                                size = size + 1;
                                nextstate <= PUSH;
                            end
                        else
                            nextstate <= OVERLOAD;
                    end
                2'b10://试图取走外卖
                    begin
                        if(size > 0)
                            nextstate <= POP;
                        else
                            nextstate <= NOFOOD;
                    end
                default:
                    nextstate <= WAIT;
            endcase

            waitled = 1;
            overloadled = 0;
            nofoodled = 0;
            keyerrorled = 0;
            pushled = 0;
            popled = 0;
        end

    PUSH://存入外卖
        begin
            //选取出一个空柜子
            for(i = 0; i <= 3 && flag == 0; i = i+1)
                begin
                    if(isuse[i] == 0)
                        flag = 1;
                    else
                        flag = 0;
                end
            flag = 0;

            //设置密码
            for(j = 0; j <= 3; j = j+1)
                boxkey[j+i*4] = key[j];

            //标记该箱子被占用
            isuse[i] = 1;

            i = 0;
            j = 0;

            nextstate <= WAIT;

            waitled = 0;
            overloadled = 0;
            nofoodled = 0;
            keyerrorled = 0;
            pushled = 1;
            popled = 0;
```

```
        end

    POP://提取外卖
        begin
            if(iscorrect_wire == 1)//输入的密码正确
                begin
                    nextstate <= WAIT;
                    size = size - 1;
                    isuse[i] = 0;
                    keyerrorled = 0;
                end
            else //输入密码错误
                begin
                    if(getinfo == 1)
                        nextstate <= WAIT;
                    else
                        nextstate <= POP;

                    keyerrorled = 1;
                end

            waitled = 0;
            overloadled = 0;
            nofoodled = 0;
            pushled = 0;
            popled = 1;
        end

    OVERLOAD:
        begin
            if(getinfo)
                nextstate <= WAIT;
            else
                nextstate <= OVERLOAD;

            waitled = 0;
            overloadled = 1;
            nofoodled = 0;
            keyerrorled = 0;
            pushled = 0;
            popled = 0;
        end

    NOFOOD:
        begin
            if(getinfo)
                nextstate <= WAIT;
            else
                nextstate <= NOFOOD;

            waitled = 0;
            overloadled = 0;
            nofoodled = 1;
            keyerrorled = 0;
            pushled = 0;
            popled = 0;
        end
```

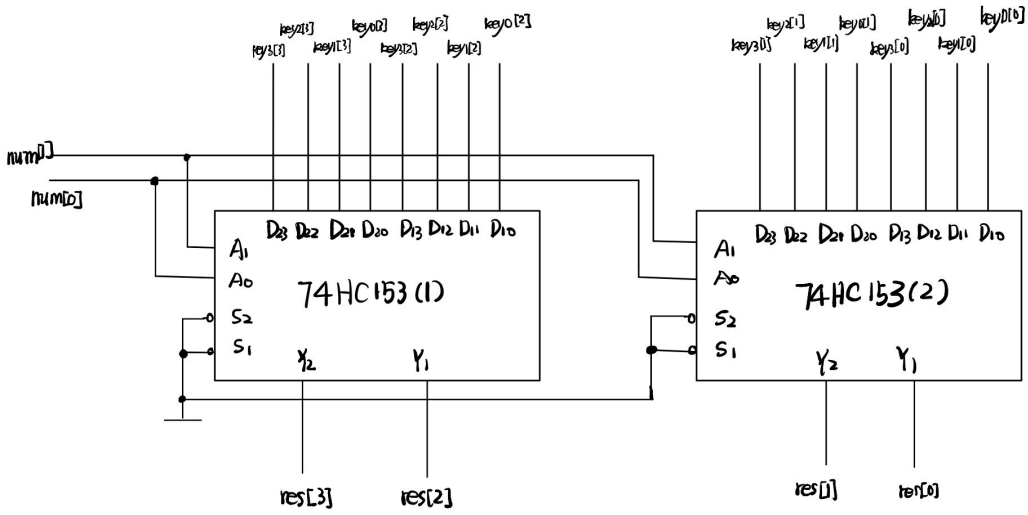
```
        default:
            nextstate <= WAIT;

    endcase
end
```

根据软件QUARTUS的仿真，可以生成以下状态图：



2. 四选一的数据选择器

   参考了课本“双四选一”的数据选择器74HC153。更具体来说，本模块需要实现的是输入四个四位的二进制信号和两位的地址信号，输出被选中的电路信号。也就是说，可以将两个74HC153芯片串联。（如下图）

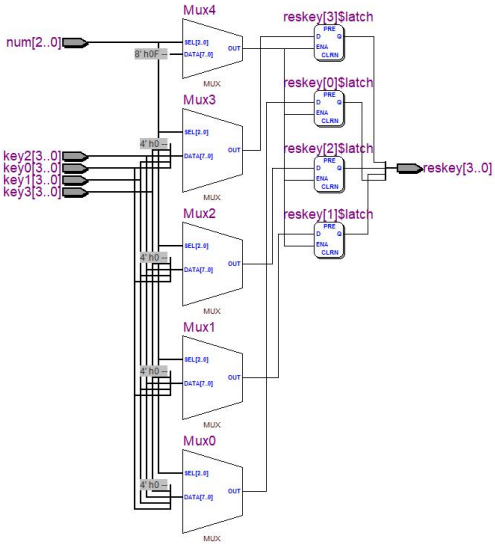


连接上述电路图略显繁琐，可以使用verilog进行设计，代码如下：

```
module select4_1(input [3:0] key0,
                 input [3:0] key1,
                 input [3:0] key2,
                 input [3:0] key3,
                 input [2:0] num,
                 output reg [3:0] reskey);

always@(num)
begin
```

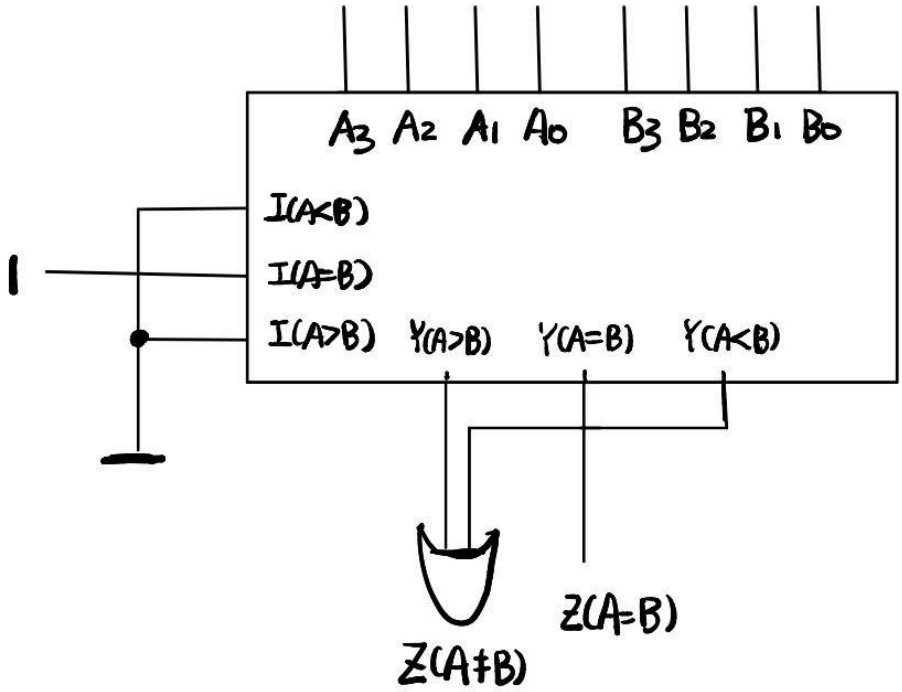
```
    case(num)
    3'b000:
        reskey = key0;
    3'b001:
        reskey = key1;
    3'b010:
        reskey = key2;
    3'b011:
        reskey = key3;
    default:
        reskey = reskey;
    endcase
end

endmodule
```

下图为QUARTUS的生成电路图以及封装好的模块：



3. 4位的数值比较器

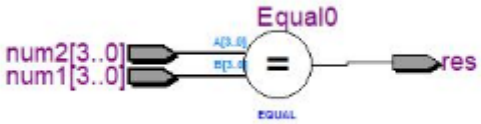
   可以直接采用课本上介绍的74HC85，完成4位数值比较。根据本设计的要求，将Y(A>B)和Y(A<B)合并为输出Y(A!=B)。

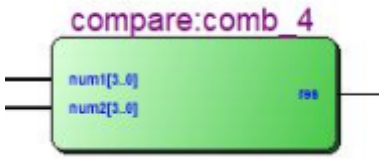


为了和前面实现保持一致，同样用verilog实现了以下的模块：

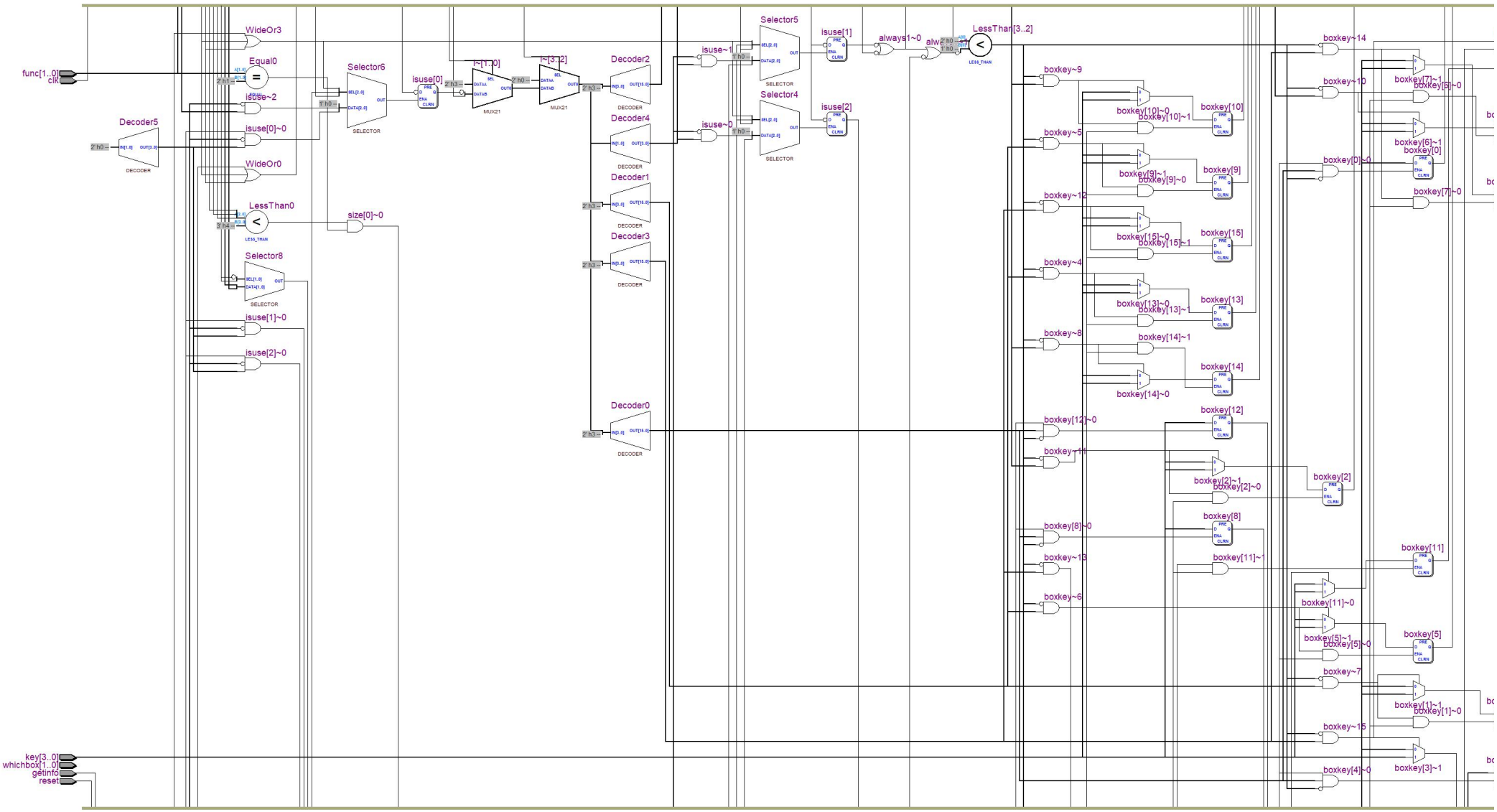
```
module compare(input [3:0] num1,
               input [3:0] num2,
               output reg res);

always@(num1 or num2)
begin
    if(num1 == num2)
        res = 1;
    else
        res = 0;
end

endmodule
```

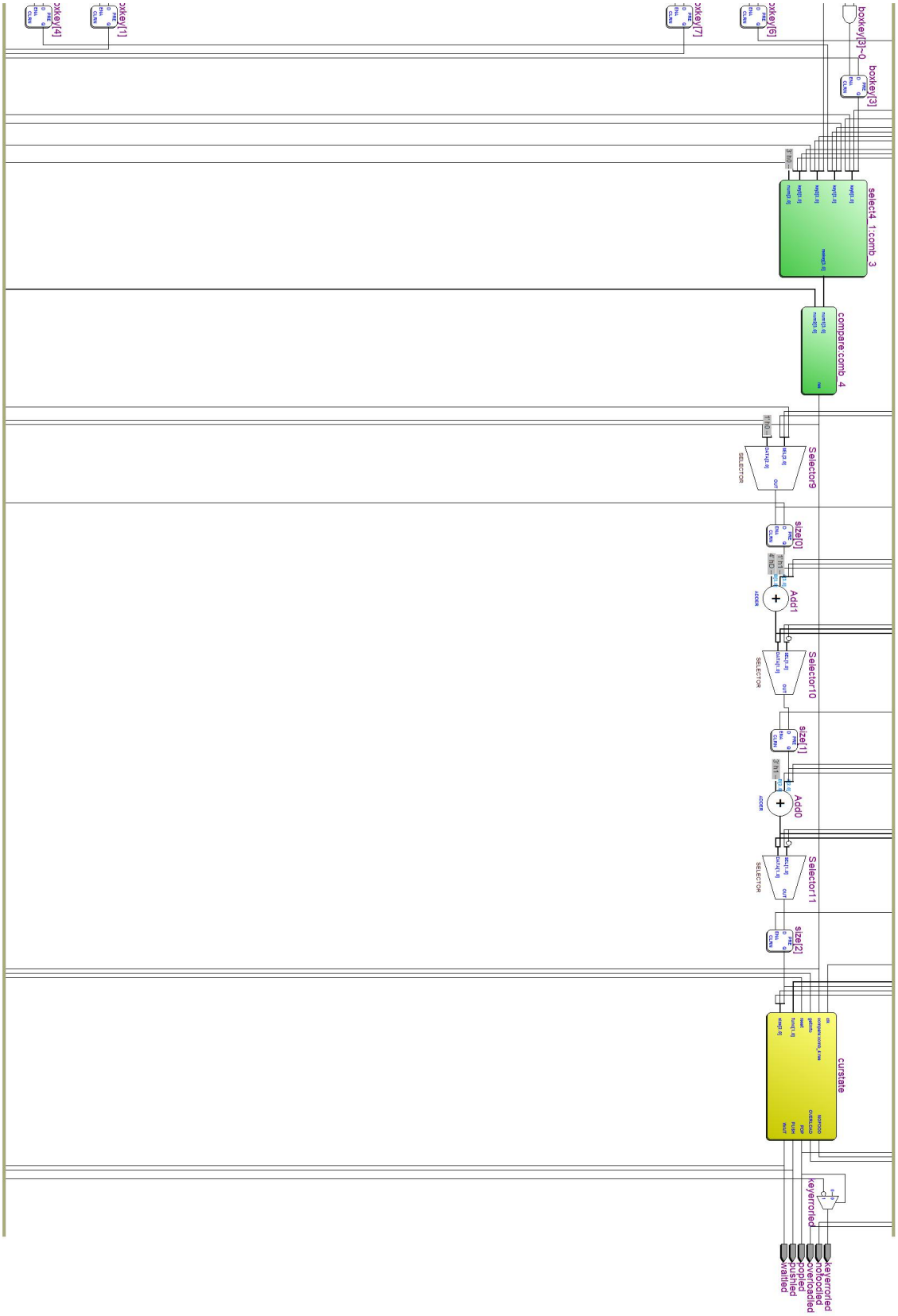
并生成了电路图以及封装好的模块。





综合以上三个模块生成的电路图为：


func[1..0]
clk
key[3..0]
whichbox[1..0]
getinfo
reset
WideOr3
Equal0
isuse~2
Decoder5
isuse[0]~0
WideOr0
LessThan0
Selector8
isuse[1]~0
isuse[2]~0
Selector6
size[0]~0
isuse[0]
i~[1..0]
i~[3..2]
MUX21
MUX21
Decoder2
Decoder4
Decoder1
Decoder3
Decoder0
isuse~1
isuse~0
Selector5
Selector4
isuse[1]
isuse[2]
always1~0
LessThan[3..2]
boxkey~9
boxkey~5
boxkey~12
boxkey~4
boxkey~8
boxkey[12]~0
boxkey~11
boxkey[8]~0
boxkey~13
boxkey~6
boxkey~14
boxkey~10
boxkey[10]~0
boxkey[10]~1
boxkey[10]
boxkey[9]~1
boxkey[9]~0
boxkey[9]
boxkey[15]~0
boxkey[15]~1
boxkey[15]
boxkey[13]~0
boxkey[13]~1
boxkey[13]
boxkey[14]~1
boxkey[14]~0
boxkey[14]
boxkey[12]
boxkey[2]~1
boxkey[2]~0
boxkey[2]
boxkey[8]
boxkey[11]~1
boxkey[11]~0
boxkey[11]
boxkey[5]~1
boxkey[5]~0
boxkey[5]
boxkey~7
boxkey[1]~1
boxkey[1]~0
boxkey~15
boxkey[4]~0
boxkey[3]~1
boxkey[7]~1
boxkey[5]~0
boxkey[6]~1
boxkey[0]
boxkey[0]~0
boxkey[7]~0



详细的代码和图片可以点击下面的链接:

链接(https://cloud.tsinghua.edu.cn/d/e54baecb508b4c878a3e/)

总结

将以上各部分通过电路搭接起来，再经过一定的电子工艺，经过简化、浓缩制成一个简单的电路结构， 并搭配上合适的外部器具，就可以生成一个自助外卖寄存机了。

受限于期末的时间和精力，本设计功能上仍然有两方面的遗憾。一是应当为自主外卖寄存机添上一个温度调节装置以保证外卖的口感；二是应当设计每个寄存柜输入密码的次数上限，比如连续输入错误3次后冻结该寄存柜并呼叫管理员。希望日后可以不断改善。