哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2019 秋季

课程名称：数字逻辑设计（实验）

实验名称：快递柜设计

实验性质：综合设计型

实验学时：6 地点：T2614

学生班级：1801105

学生学号：180110527

学生姓名：李秋阳

评阅教师：

报告成绩：

实验与创新实践教育中心制

2019 年 12 月

一、 实验项目

快递柜设计

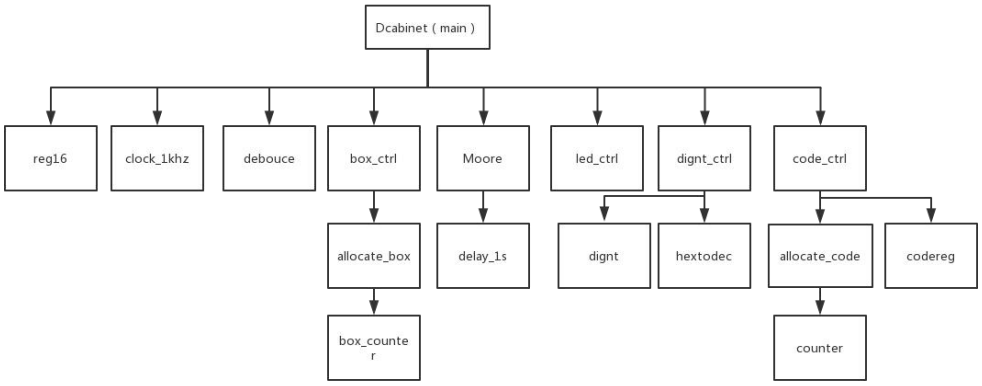
二、 系统功能详细设计及实现

1. 系统设计描述和硬件框图

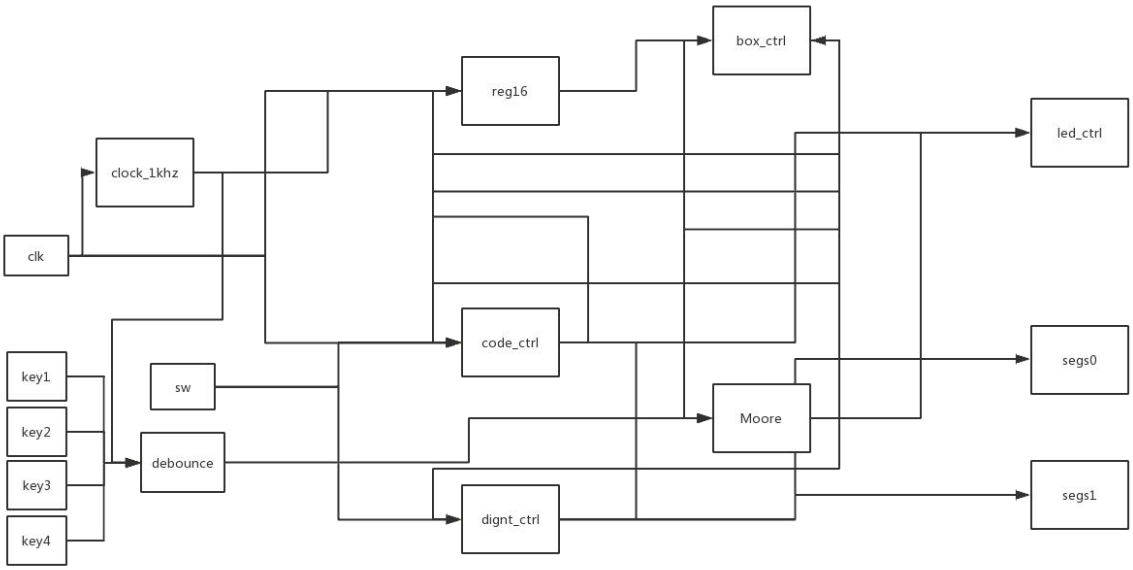
此系统描述了简易快递柜的工作过程。系统连接后进入欢迎界面，显示空快递柜的数量（快递柜总数量为 16）。选择存柜后，随机分配空箱号并显示随机生成的密码，确认后返回欢迎界面。选择取柜后，通过拨码开关输入密码，按确认后显示密码。若密码正确，箱号对应 led 灯闪烁；若密码错误，数码管显示“88888888”。按确认后返回欢迎界面。

系统总共分为 8 个模块进行设计，分别是 clock_1khz——时钟分频模块，reg16——快递箱寄存器模块，debounce——消抖模块，Moore——状态机模块，box_ctrl——快递箱控制模块，led_ctrl——led 灯控制模块，code_ctrl——密码控制模块，dignt_ctrl——数码管控制模块。其下包含实现其功能的小模块。

各模块关系如下：

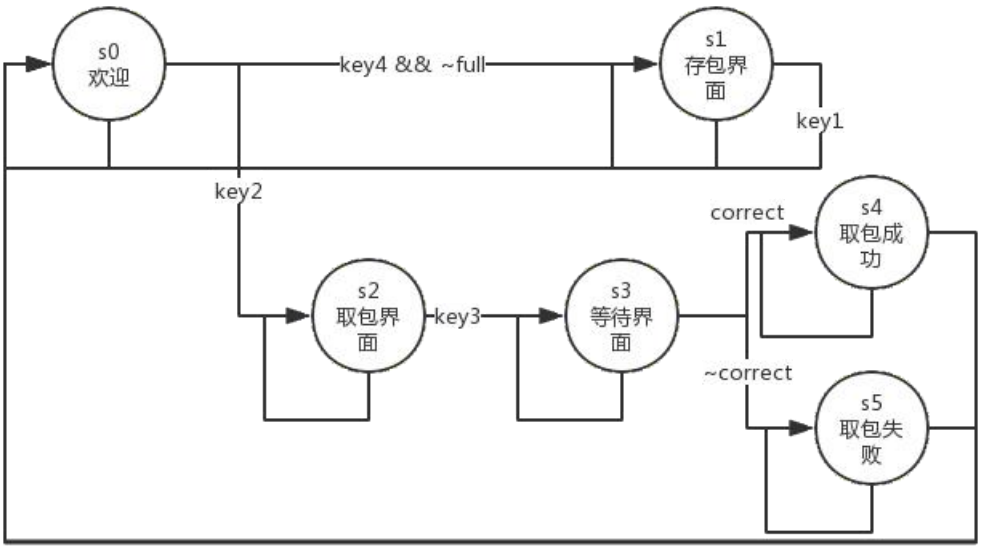


方框图如下：



2. 状态描述及状态转换图

```
parameter s0 = 3'b000;// 欢迎界面
parameter s1 = 3'b001;// 存包界面
parameter s2 = 3'b100; // 取包界面
parameter s3 = 3'b101; // 密码输入后的等待界面
parameter s4 = 3'b111;// 取包成功界面
parameter s5 = 3'b110;// 取包失败界面
```



3. 模块描述

main 主模块

```
module main(
    input clk, input[15:0] sw, input key1,
    input key2, input key3, input key4,
    output[15:0] led, output[7:0] len,
    output[7:0] segs0, output[7:0] segs1
    );
    wire clk1;
    wire key1_de,key2_de,key3_de,key4_de;
    wire [15:0] B_in;
    wire [15:0] B_out;
    wire [4:0] E_boxes;
    wire full;
    wire [3:0] B_num;
    wire [3:0] B_clear;
    wire [2:0] state;
    wire [15:0] code;
    wire correct;
    wire B_ready;
```

clk 为时钟信号，sw 为拨码开关，key1 为确认，key2 为取包，key3 为输入，key4 为存包，led 为 led 灯，len 为数码管使能开关，segs0 为第一个数码管管脚，segs1 为第二个数码管管脚，clk1 为分频后时钟，key1_de 到 key4_de 为消抖后按键信号，B_in 为快递柜寄存器输入，B_out 为快递柜寄存器输出，full 为快递柜是否满的标志，state 为状态寄存器，B_num 为分配的柜号，B_clear 为要清除的柜号，code 为密码，B_ready 为延时完毕的标志。

clock_1khz 分频模块

```
`timescale 1ns / 1ps
module clock_1khz(input clk, output reg clk1 = 0);
    reg[15:0] divcount = 0;
    always @ (posedge clk) begin
        if (divcount == 16'd49999)
        begin
            divcount <= 0;
            clk1 <= ~clk1;
        end
        else
            divcount <= divcount + 1;
    end
endmodule
```

获得 1khz 的分频时钟

reg16 快递柜寄存器模块

```
`timescale 1ns / 1ps
module reg16(input clk, input[15:0] d, output reg[15:0] q = 0, output[4:0] E_count);
    assign E_count = ~d[0] + ~d[1] + ~d[2] + ~d[3] + ~d[4] + ~d[5] + ~d[6] + ~d[7]
    + ~d[8] + ~d[9] + ~d[10] + ~d[11] + ~d[12] + ~d[13] + ~d[14] + ~d[15];
    always @(posedge clk) begin
        q <= d;
    end
endmodule
```

d 为输入端，q 为输出端，E_count 为空箱子个数

debounce 按键消抖模块

```
`timescale 1ns / 1ps
module debounce(input clk, input key, output reg result = 0);
reg [3:0] count = 0;
always @(posedge clk)
   begin
       if(count >= 4'd10 && result == 0)
           begin
               result <= 1;
           end
       if(key == 0)
           begin
               count <= 0;
               result <= 0;
           end
       else if(count < 4'd10)
           begin
               count <= count + 1;
           end
   end
endmodule
```

clk 为时钟信号，key 为需要消抖的按键，result 为消抖后的按键信号

Moore 状态机模块

实现状态转换

```
`timescale 1ns / 1ps
module delay_1s(input clk, input en, output reg D_delay = 0);
    reg[26:0] count = 0;
    always @(posedge clk) begin
        if (en == 0) begin
            count <= 0;
            D_delay <= 0;
        end
        else if (count == 28'd99999999) begin
            D_delay <= 1;
        end
        else begin
            count <= count + 1;
        end
    end
endmodule
```

其中包含 delay_1s 模块，en 为使能端，D_ready 为延时完成标志

box_ctrl 快递柜控制模块

alloc_ready 为快递柜分配完成的标志，实现随机分配箱号，其中包含 allocate_box 模块

```
`timescale 1ns / 1ps
module allocate_box(input clk, input[2:0] state, input[15:0] box,
output reg[3:0] B_num, output reg D_delay = 0);
    parameter s1 = 3'b001;
    reg find = 0;
    reg[3:0] allocated;
    integer i;
    wire[3:0] rand;
    box_counter bc(clk, rand);
    always @(*) begin
        allocated = rand;
        find = 0;
        for (i=0; i<16; i=i+1) begin
            if (box[(rand+i) % 16] == 0 && ~find) begin
                allocated = rand + i;
                find = 1;
            end
        end
    end
    always @(posedge clk) begin
        if (state == s1 && ~D_delay) begin
            B_num <= allocated;
            D_delay <= 1;
        end
        if (state != s1) begin
            D_delay <= 0;
        end
    end
endmodule
```

box 为快递箱寄存器输出，其中包含随机生成箱号的计数器模块 box_counter

```
`timescale 1ns / 1ps
module box_counter(input clk, output[3:0] randbox);
    reg[7:0] count = 8'h1B;
    assign randbox[3:0] = count[3:0];
    always @(posedge clk)
        begin
            count <= (count == 8'hFF ? 8'h1B : count + 1);
        end
endmodule
```

led_ctrl led 灯控制模块

根据分配的箱号确定 led 的亮灭，起到类似译码器的作用

code_ctrl 密码控制模块

作为随机生成密码的寄存器，randcode 为随机生成的密码，input 为输入的密码，box_to_write 为写入的密码，box_matched 为随机分配的快递柜号，其下的随机

分配模块及计数器与 box_ctrl 类似。

```
`timescale 1ns / 1ps
module counter(input clk, output reg[15:0] count);
    reg[19:0] count1 = 0;
    always @(posedge clk) begin
        count1 <= count1 + 1;
        count <= count1[19:4];
    end

endmodule
```

dignt_ctrl 数码管控制模块

```
`timescale 1ns / 1ps
module dignt_ctrl(input clk, input[2:0] state, input[4:0] E_boxes,
                  input[3:0] B_num, input[15:0] code, input[15:0] sw,
                  output[7:0] len, output[7:0] segs0, output[7:0] segs1);
    parameter
    s0 = 3'b000,
    s1 = 3'b001,
    s2 = 3'b100,
    s3 = 3'b101,
    s4 = 3'b111,
    s5 = 3'b110;
    reg[7:0] en;
    reg[1:0] sel0 = 0;
    reg[1:0] sel1 = 0;
    reg[3:0] hexad0;
    reg[3:0] hexad1;
    wire E_tens;
    wire[3:0] E_ones;
    reg[4:0] B_num_2;
    wire B_num_tens;
    wire[3:0] B_num_ones;
    reg[15:0] inputcode;
    dignt dig1(.en(en[3:0]), .sel(sel0), .hexad(hexad0), .ce(len[3:0]), .seg(segs0));
    dignt dig2(.en(en[7:4]), .sel(sel1), .hexad(hexad1), .ce(len[7:4]), .seg(segs1));
    hextodec htd1(.hex(E_boxes), .tens(E_tens), .ones(E_ones));
    hextodec htd2(.hex(B_num_2), .tens(B_num_tens), .ones(B_num_ones));
```

E_tens 为空快递柜数十进制表示的十位，E_ones 为空快递柜数十进制表示的个位，B_num_2 为快递柜号，B_tens 为快递柜数十进制表示的十位，B_ones 为快递柜数十进制表示的个位

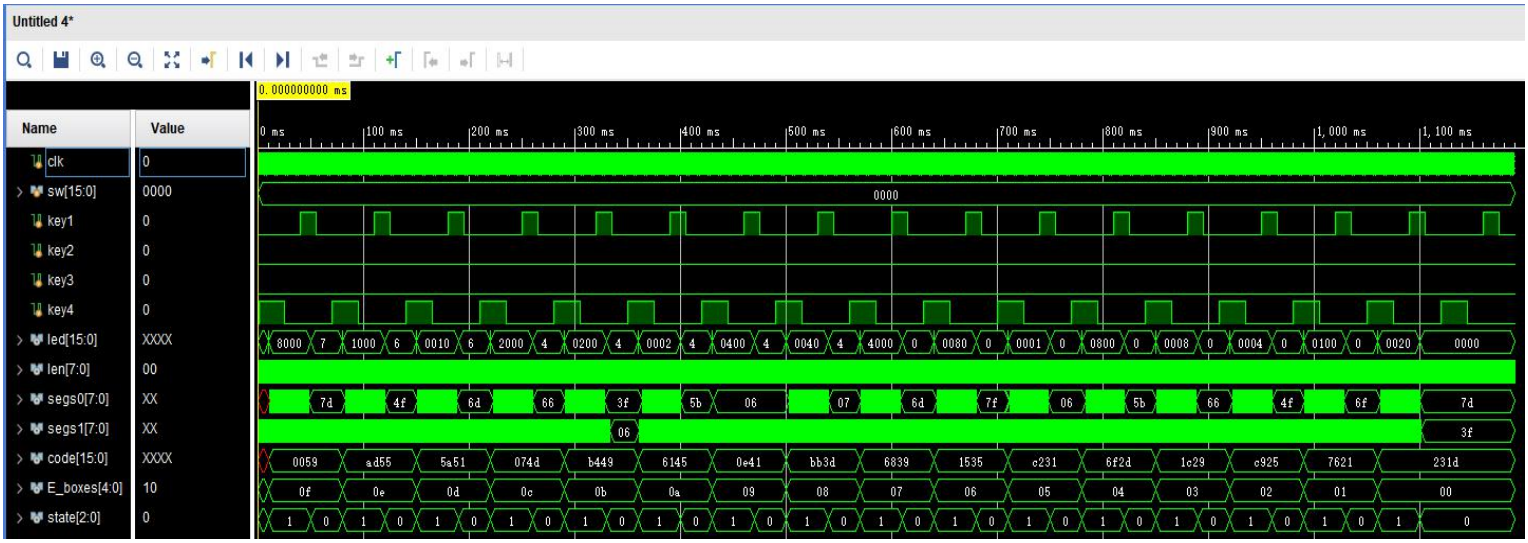
控制数码管显示的时序以及数码管的使能

4. 管脚分配表

信号	管脚	信号	管脚	信号	管脚	信号	管脚
sw[15]	P5	sw[7]	U3	led[15]	K3	led[7]	K2
sw[14]	P4	sw[6]	U2	led[14]	M1	led[6]	J2
sw[13]	P3	sw[5]	V2	led[13]	L1	led[5]	J3
sw[12]	P2	sw[4]	V5	led[12]	K6	led[4]	H4
sw[11]	R2	sw[3]	V4	led[11]	J5	led[3]	J4
sw[10]	M4	sw[2]	R3	led[10]	H5	led[2]	G3
sw[9]	N4	sw[1]	T3	led[9]	H6	led[1]	G4
sw[8]	R1	sw[0]	T5	led[8]	K1	led[0]	F6
len[7]	G6	len[3]	H1	segs0[7]	D5	segs0[3]	B1
len[6]	E1	len[2]	C1	segs0[6]	B2	segs0[2]	A3
len[5]	F1	len[1]	C2	segs0[5]	B3	segs0[1]	A4
len[4]	G1	len[0]	H1	segs0[4]	A1	segs0[0]	B4
segs1[7]	H2	segs1[3]	F4	key1	R11		
segs1[6]	D2	segs1[2]	D3	key2	R17		
segs1[5]	E2	segs1[1]	E3	key3	V1		
segs1[4]	F3	segs1[0]	D4	key4	U4		

三、 调试报告

1. 仿真界面截图及仿真数据分析



led[15:0]为 led 灯状态，key1 为取包信号，key4 为存包信号，code[15:0]为随机生成的四位十六进制密码，state[2:0]为主模块状态机的状态，E_boxes[4:0]为剩余的快递柜数量。

可以看到，随着 key4，key1 信号的交替输入，状态机状态在 0 和 1 之间切换，每次状态变为 1 时 E_boxes 中的数值减一，生成新的随机密码，led 灯状态随之改变。直到 key4、key1 信号发出 16 次以后，E_boxes 中的数值减为 0，不再生成新的密码，状态机状态不再跳转。

2. 出现的问题、分析原因及解决方案

在设计完成后发现仿真和上板都显示状态机状态没有跳转，经检查发现 Moore 状态机模块逻辑设计存在问题，是由于对三段式状态机理解不深，组合逻辑与时序逻辑混乱，在严格按照三段式状态机设计逻辑对代码进行修正后现象基本正确。多次尝试后发现随机生成的密码第一位一直为“1”，检查后发现是因为先给第一位赋了初值，没有受计数器影响，修正后现象完全正确。

四、 总结及实验课程感想

痛苦的修炼终于结束了。Verilog 这样的硬件描述语言带来比数据结构更深痛苦的原因不在于它有多难，而是你以为自己的代码完全没有问题，仿真结果也完全正确的时候，各种各样的连接错误、现象错误会接二连三地摧残你的心智。而且由于要综合、实现、生成比特流文件，一次调试需要的时间往往很长，一个小问题找不到原因可能就是好几个小时的调试。最后一次大实验体验还算不错，在机房连续 9 个小时的奋战后顺利通关。虽然过程很痛苦，但是完成后的快感是难以言喻的。

本门课程的实验不仅让我对于如今基于硬件编程有了一定的了解，也让我学到了模块化编程的一些知识，收获是显而易见。老师和助教们对课程的精心设计和实验中的悉心指导也给了我很大的帮助，非常感谢！