哈尔滨工业大学
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2023 秋季

课程名称：数字逻辑设计（实验）

实验名称：简易计算器

实验性质：综合设计型

实验学时：6 地点：T2615

学生班级：计科 5 班

学生学号：220110515

学生姓名：金正达

评阅教师：

报告成绩：

实验与创新实践教育中心制

2023 年 11 月

设计的功能描述

该简易计算器实现了

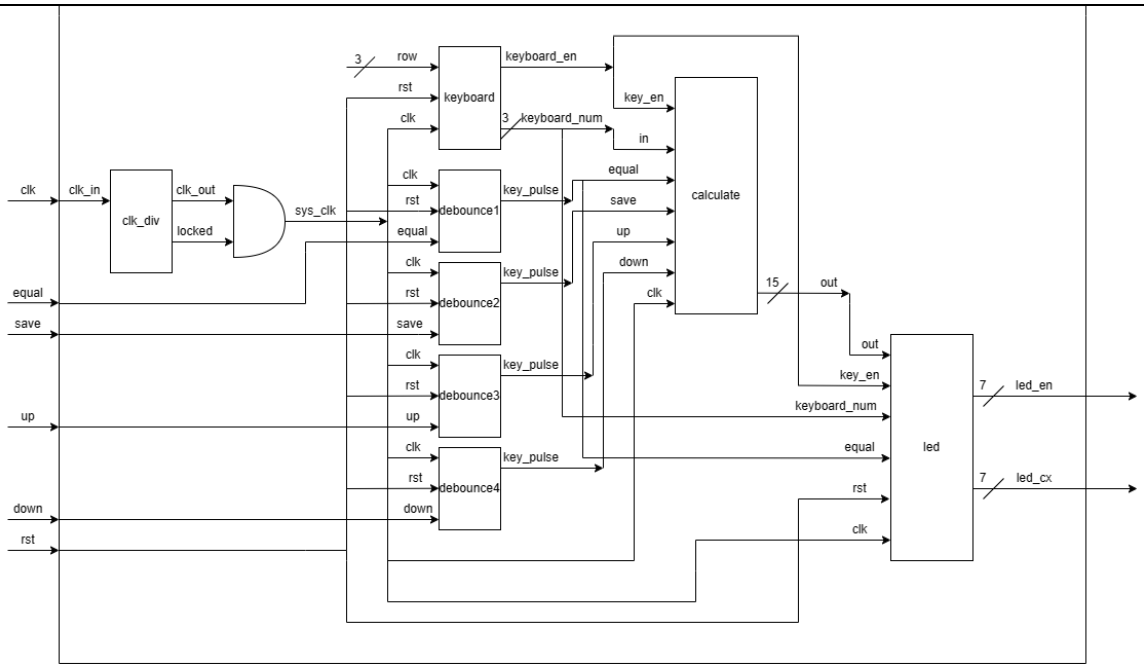
（1）两位十进制正整数的加、减、乘、除运算，可以进行连续计算；

（2）实时在数码管上显示四位键盘输入，在数码管上显示计算结果；

（3）保存当前计算结果，至多存储上四个计算结果；

（4）通过按键进行上翻上一个计算结果和下翻下一个计算结果，上翻下翻的结果可以进一步用于连续计算。

系统功能详细设计



读取输入键盘模块功能集中，计算模块专注于计算出结果，数码管控制模块主要处理数码管显示，按键消抖模块只处理按键抖动，这样划分有利于各个部分的抽象，对外部仅暴露其接口，以便于更高效的开发，同时也利于模块的复用。

clk_div 模块为每个模块提供时钟信号，其输出 sys_clk 作为每个模块的时钟信号输入；debounce 模块为各个按键处理抖动，将消抖后的按键

信号输入 calculate 和 led 模块；keyboard 模块读取键盘输入，提供按键使能信号 keyboard_en 和输入按键信号 keyboard_num 给 calculate 模块和 led 模块；calculate 模块根据输入计算出结果 out 提供给 led 模块；led 模块根据提供的按键信号和计算结果，输出数码管使能信号 led_en 和数码管驱动信号 led_cx。

模块设计与实现

（1）clk_div 模块

功能：该模块用于内置时钟降频，实现将 100MHz 时钟降频为 20MHz 时钟。

输入信号：clk_in1（内置 100MHz 时钟信号）

输出信号：clk_out1（20MHz 时钟信号），locked（PLL 锁定信号）

（2）debounce_button 模块

功能：实现按键消抖

输入信号：clk（时钟信号），rst（复位信号），key（需要消抖的按键信号）

输出信号：key_pulse（消抖后的按键信号）

设计思路：采用计时消抖法。

实现逻辑：检测到按键被按下之后，开始计时，一段时间之后开始采样按键信号，即可消除按键前抖动的影响。

关键代码：

```
14          always @(posedge clk or posedge rst) begin
15              if (rst) begin  //复位
16                  key_last <= 1'b0;
17                  key_now <= 1'b0;
18                  key_pulse <= 1'b0;
19              end
20              else begin
21                  if (key_last == key_now) begin //实时检测按键状态
22                      key_now <= key;
23                      key_last <= key_now;
24                      key_pulse <= 0;
25                      cnt <= 32'd1;
26                  end
27                  else begin
28                      if (cnt >= 32'd15000) begin  //计时完成，开始采样信号
29                          key_pulse <= key_now & ~key_last;
30                          key_now <= key;
31                          key_last <= key_now;
32                      end
33                      else //计时
34                          cnt <= cnt + 32'd1;
35                  end
36              end
37          end
```

（3）keyboard 模块

功能：读取键盘按键信号

输入信号：clk（时钟信号），rst（复位信号），row（行扫描信号）

输出信号：col（行扫描信号），keyboard_en（键盘按键使能号），keyboard_num（键盘输入）

设计思路：采用列扫描法

实现逻辑：采用计数器，计数一段时间后进行列扫描信号的移位。

关键代码：

```
27  always @(posedge clk or posedge rst) begin
28      if (rst == 1)           col <= 4'b1111;
29      else if (col ==  4'b1111) col <= 4'b1110;
30      else if (cnt_end)         col <= {col[2:0], col[3]};
31  end
```

产生列扫描信号。

（4）calculate 模块

功能：实现二位十进制正整数的加、减、乘、除及连续运算，其内置 save_nums 模块实现了计算结果的存储和上翻下翻结果。

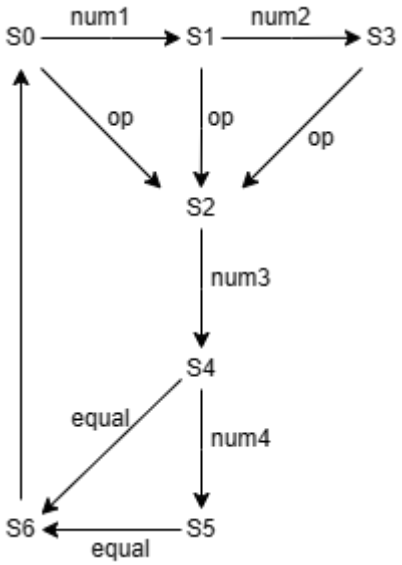
输入信号：clk（时钟信号），rst（复位信号），key_en（键盘按键使能信号），equal（等于按键信号），save（保存按键信号），up（上翻按键信号），down（下翻按键信号）

输出信号：in（键盘输入），out（计算结果输出）

设计思路：采用状态机实现

实现逻辑：S0 为初始状态，编码为 3’d0；S1 为读取第一个数的一位数状态，编码为 3’d1；S2 为读取操作符状态，编码为 3’d2；S3 为读取到第一个数的两位数状态，编码为 3’d3；S4 为读取到第二个数一位数的状态，编码为 3’d4；S5 为读取到第二个数两位数的状态，编码为 3’d5；S6 为计算状态，编码为 3’d6。

状态转移图：



状态转移判断主要代码如下：

```
48      always @(*) begin
49          case (current_state)
50              S0: if(key_en) begin
51                      if (in <= 9) next_state = S1;
52                      else next_state = S2;
53                  end
54                  else next_state = S0;
55              S1: if (key_en) begin
56                      if (in <= 9 ) next_state = S3;
57                      else next_state = S2;
58                  end
59                  else next_state = S1;
60              S2: if(key_en && in <= 9) next_state = S4;
61                  else next_state = S2;
62              S3: if (key_en) next_state = S2;
63                  else next_state = S3;
64              S4: if(key_en) begin
65                      if (in <= 9) next_state = S5;
66                  end
67                  else if (equal) next_state = S6;
68                  else next_state = S4;
69              S5: if(equal) next_state = S6;
70                  else next_state = S5;
71              S6: if (cnt_end) next_state = S0;
72                  else next_state = S6;
73              default : next_state = S0;
74          endcase
75      end
```

例如，在 S0 状态，判断是否有输入，若有按键被按下，则判断输入是否小于等于 9，是则转 S1，否则转 S2，若无按键被按下，则保持 S0。其余状态转移类似。

状态输出主要代码如下：

```
        case(current_state)
            S0: if(key_en) begin
                    if (in <= 9) num1 <= in;
                    else begin
                        op <= in;
                        num1 <= last_out;
                    end
                end
            S1: if (key_en) begin
                    if (in <= 9 ) num1 <= num1 * 10 + in;
                    else op <= in;
                end
            S2: if(key_en && in <= 9) num2 <= in;
            S3: if(key_en) op <= in;
            S4: if(key_en && in <= 9) num2 <= num2 * 10 + in;
            S6: begin
                case (op)
                    4'ha: result <= num1 + num2;
                    4'hb: result <= num1 - num2;
                    4'hc: result <= num1 * num2;
                    4'hd: result <= num1 / num2;
                    default : result <= result;
                endcase
            end
            default: result <= result;
        endcase
```

例如，在 S0 状态，若按键被按下，判断按键输入是否小于等于 9，是则读取第一个操作数，否则读取操作符且将第一个操作数置为上一次的计算结果。在 S6 状态，根据所读取的操作符进行相应的计算。其余状态类似。

（5）save_nums 模块

功能：实现至多四个计算结果的储存，并且可以上翻下翻之前的计算结果。

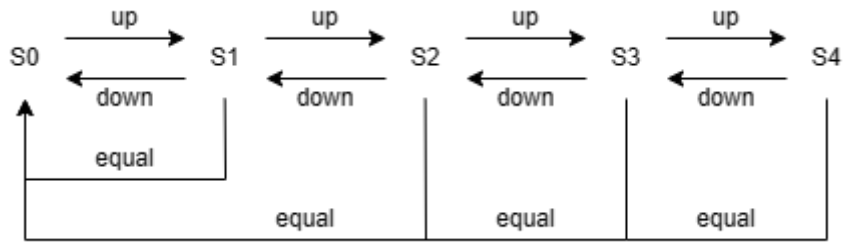
输入信号：clk（时钟信号），rst（复位信号），equal（等于按键信号），save（保按键信号），up（上翻按键信号），down（下翻按键信号），resut（计算结果）

输出信号：last_out（输出结果）

设计思路：采用状态机实现

实现逻辑：S0 为初始状态，表示显示当前结果，编码为 3’d0；S1 表示显示上一个计算结果，编码为 3’d1；S2 表示显示上上一个计算结果，编码为 3’d2；S3 表示显示上上上一个计算结果，编码为 3’d3；S4 表示显示上上上上一个计算结果，编码为 3’d4。

状态转移图：



保存计算结果部分代码如下：

```
24      always @(posedge clk or posedge rst) begin
25          if (rst) num_saved <= 0;
26          else if (save) num_saved <= {num_saved[47: 0], result};
27          else num_saved <= num_saved;
28      end
```

使用 64 位寄存器变量来保存结果，当保存按键被按下时进行移位拼接，来保存至多三个计算结果。

状态转移主要代码如下：

```
35      always @(*) begin
36          case(current_state)
37              S0: if (up) next_state = S1;
38                  else next_state = S0;
39              S1: if (equal) next_state = S0;
40                  else if (up) next_state = S2;
41                  else if (down) next_state = S0;
42                  else next_state = S1;
43              S2: if (equal) next_state = S0;
44                  else if (up) next_state = S3;
45                  else if (down) next_state = S1;
46                  else next_state = S2;
47              S3: if (equal) next_state = S0;
48                  else if (up) next_state = S4;
49                  else if (down) next_state = S2;
50                  else next_state = S3;
51              S4: if (equal) next_state = S0;
52                  else if (down) next_state = S3;
53                  else next_state = S4;
54          endcase
55      end
```

例如，在 S0 状态，若按下上翻按键，则跳转 S1 状态，否则保持 S0 状态；在 S1 状态，若按下上翻按键，则跳转 S2 状态，若按下下翻按键，则跳转 S0 状态，否则保持 S1 状态。其余状态转移类似。

状态输出主要代码如下：

```
57      always @(posedge clk or posedge rst) begin
58          if (rst) last_out <= 0;
59          else begin
60              case(current_state)
61                  S0: last_out <= result;
62                  S1: last_out <= num_saved[15: 0];
63                  S2: last_out <= num_saved[31: 16];
64                  S3: last_out <= num_saved[47: 32];
65                  S4: last_out <= num_saved[63: 48];
66                  default : last_out <= 0;
67              endcase
68          end
69      end
```

根据状态来输出此时的计算结果。例如，S0 时输出当前计算结果，S1 时输出上一个计算结果。

（6）led 模块

功能：实时在数码管显示按键输入和输出。

输入信号：clk（时钟信号），rst（复位信号），key_en（按键使能信号），equal（等于按键信号），keyboard_num（按键输入），out（计算结果）

输出信号：led_en（数码管使能信号），led_cx（数码管驱动信号）

设计思路：每隔一段时间数码管使能信号移位一次，相应的显示数字变化一次，频率超过人眼的分辨范围即可看到常亮的数码管显示。

实现逻辑：使用计数器来控制间隔时间，若达到计数截止，则将数码管使能信号移位一次，否则保持原有使能信号。对于不同的使能信号，确定该使能信号下应该显示的数字，再由数字来确定数码管驱动信号。

关键代码：

```
    always @(posedge clk or posedge rst) begin
        if (rst)
            led_en <= 8'b11111110;
        else if (cnt_end)
            led_en <= {led_en[0], led_en[7: 1]};
        else
            led_en <= led_en;
    end
```

用于移位数码管使能信号。

```
always@(*) begin
        case (led_en)
            8'b01111111: num = mid_num1[15: 12];
            8'b10111111: num = mid_num1[11: 8];
            8'b11011111: num = mid_num1[7: 4];
            8'b11101111: num = mid_num1[3: 0];
            8'b11110111: num = mid_num2[15: 12];
            8'b11111011: num = mid_num2[11: 8];
            8'b11111101: num = mid_num2[7: 4];
            8'b11111110: num = mid_num2[3: 0];
            default num = num;
    endcase
end
```

用于确定数码管当前显示数字。

（7）counter 模块

功能：计数器模块，计数达到设定值输出 1.

输入信号：clk（时钟信号），rst（复位信号），cnt_inc（计数使能）

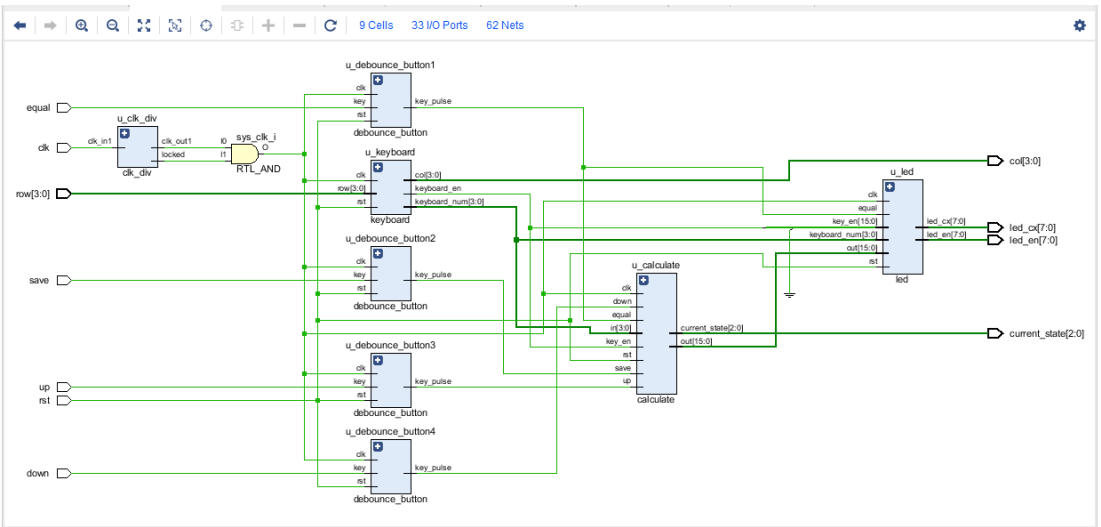
输出信号：cnt_end（计数周期性截止信号），cnt（当前计数数值）

设计思路：在每个时钟周期计数一次，计数结束复位。

实现逻辑：简单的条件语句实现（关键代码）：

```
16   always @(posedge clk, posedge rst) begin
17       if (rst) cnt <= 1'b0;
18       else if (cnt_end) cnt <= 1'b0;
19       else if (cnt_inc) cnt <= cnt + 1'b1;
20   end
```

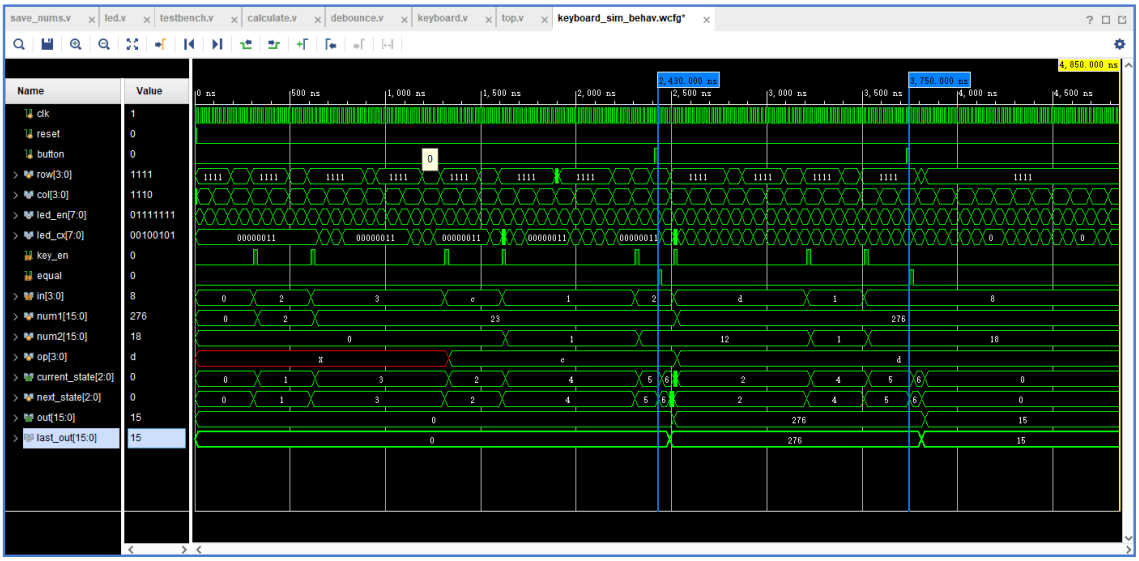
（8）顶层模块的 RTL 分析原理图：



从左到右分别为时钟降频模块、键盘输入模块、按键消抖模块、计算模块和数码管控制模块。

调试报告

仿真波形截图及仿真分析



关键信号：

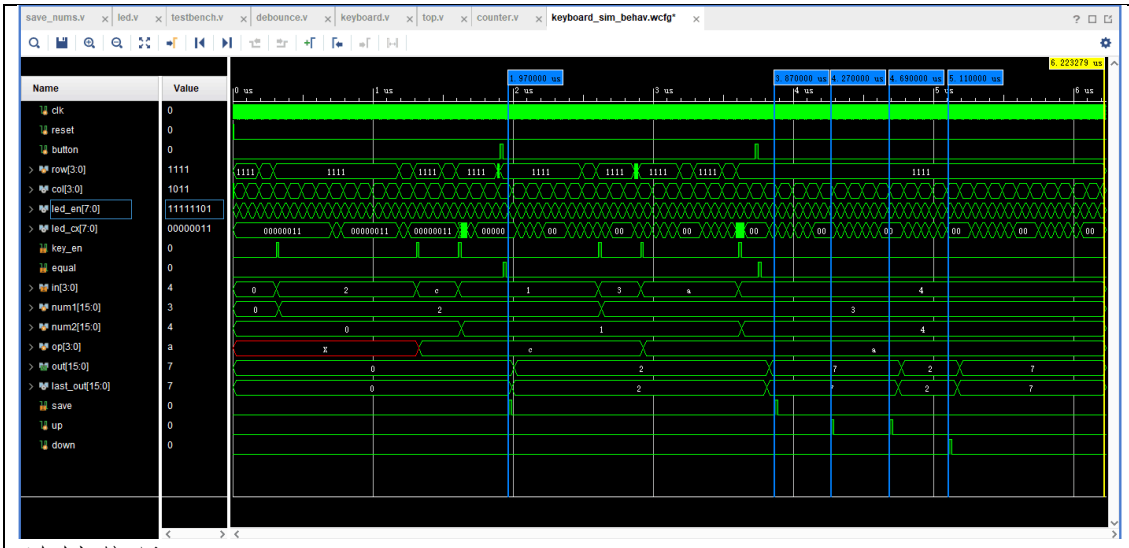
clk 为时钟信号，reset 为复位信号，button 为计算信号，row 为行扫描信号，col 为列扫描信号，led_en 为数码管使能信号，led_cx 为数码管驱动信号，key_en 为按键使能信号，equal 为内部计算按键信号，in 为计算模块输入信号，num1 为第一个操作数，num2 为第二个操作数，op 为操作符，out 为计算输出，last_out 为输出存储。

（1）在 2,430,000ns 之前，计算模块读入 2、3、c、1、2，转化为 23 储存在 num1 之中，c 储存在 op 之中，12 储存在 num2 之中。

（2）2,430,000ns 时，equal 信号使能，执行计算步骤，即 23*12，out 变为 276，符合预期。

（3）2,430,000ns 到 3,750,000 之间，计算模块输入 d、1、8，num1 读入上一次计算结果 276，op 储存 d，num2 储存 18，符合预期。

（4）3,750,000ns 时，equal 使能，执行计算步骤，即 276 / 18，out 变为 15，符合预期。



关键信号：

clk 为时钟信号，reset 为复位信号，button 为计算信号，row 为行扫描信号，col 为列扫描信号，led_en 为数码管使能信号，led_cx 为数码管驱动信号，key_en 为按键使能信号，equal 为内部计算按键信号，in 为计算模块输入信号，num1 为第一个操作数，num2 为第二个操作数，op 为操作符，out 为计算输出，last_out 为输出存储，save 为计算结果保存按键信号，up 为上翻计算结果信号，down 为下翻计算结果信号。

（1）1,970,000ns 时计算出 2，save 使能，结果被保存。

（2）3,870,000ns 时计算出 7，save 使能，结果被保存。

（3）4,270,000ns 时，up 使能，上翻上一个计算结果，即 7，out 输出 7，符合预期。

（4）4,690,000ns 时，up 使能，再上翻上一个计算结果，即 2，out 输出 2，符合预期。

（5）5,110,000ns 时，down 使能，下翻一个计算结果，即 2，out 输出 2，符合预期。

综上所述，该简易计算器实现了预期功能。

设计过程中遇到的问题及解决方法
（1）calculate 模块实现状态机的过程中，将状态转移逻辑和输出逻辑写在同一个 always 块之中，导致计算输出始终错误，状态机状态跳转也不符合预期。解决方法：拆分状态机状态转移逻辑和输出逻辑，写在两个 always 块之中，问题就得到了解决。 （2）led 模块，实现数码管显示，将移位操作写为组合逻辑，导致移位冲突，移位结果不符合预期，数码管显示不符合预期。解决方法：采用时序逻辑进行移位逻辑表达，采用非阻塞赋值，即可获得预期显示效果。 （3）save_nums 模块，每次执行计算按下 equal，上翻下翻结果就会出现错误，只能正确上翻或者下翻结果一次。解决方法：在状态转移逻辑中加入新的状态转移，即在每个状态下按下 equal 均会回到 S0 初始状态，即可实现上翻下翻的正确显示。
课程设计总结
（1）在设计过程中，采用“自顶向下”的设计理念，逐步将复杂的功能划分为各个模块，分模块实现各个功能，组合在一起即实现了该简易计算器的基本功能。在实现过程中，根据不同的操作需要、数据类型选用组合逻辑或者时序逻辑来实现，按需设计状态机来完成较复杂功能的实现。 （2）希望实验课的文档能够更详细一些，比如，在介绍数码管的原理时，可以使用更多的图和标注来说明数码管引脚的控制，也可以给出简单的驱动案例来说明。 （3）这一系列的实验课锻炼了我数字电路设计的能力、verilog 的使用和调试能力、仿真波形分析能力，加深了我对数字逻辑设计和 verilog 硬件描述语言的理解。