设计的功能描述

基本功能：

1. 系统在开机上电或按下复位键 S4 时，数码管显示 8 个“0”，等待按键 S0 被按下；
2. 按下按键 S0 后，系统随机生成 5 个互不相同的 5 位八进制数，并以 1s 为时间间隔播放这五个数一次，播放完毕后数码管保持显示最后一个 5 位数，等待按键 S1 被按下；
3. 按下按键 S1 后，系统读取拨码开关的值并显示在数码管上，若所输入的值大于 4，数码管显示 8 个“F”；
4. 按下按键 S2 后，系统开始根据拨码开关的值对上一步所输入的地址中存放的 5 位八进制数进行匹配。每按下一次按键 S3，读取拨码开关 sw[2:0]作为输入数据，当最新输入的 5 位数字等于需要进行匹配的数字时，数码管显示“地址数-匹配成功的 5 位八进制数”；匹配不成功时显示 8 个以 4Hz 不断闪烁的“0”；
5. 匹配成功后，再次按下 S3 可进行下一轮匹配；按下复位键 S4 可重新生成一组 5 位八进制数。

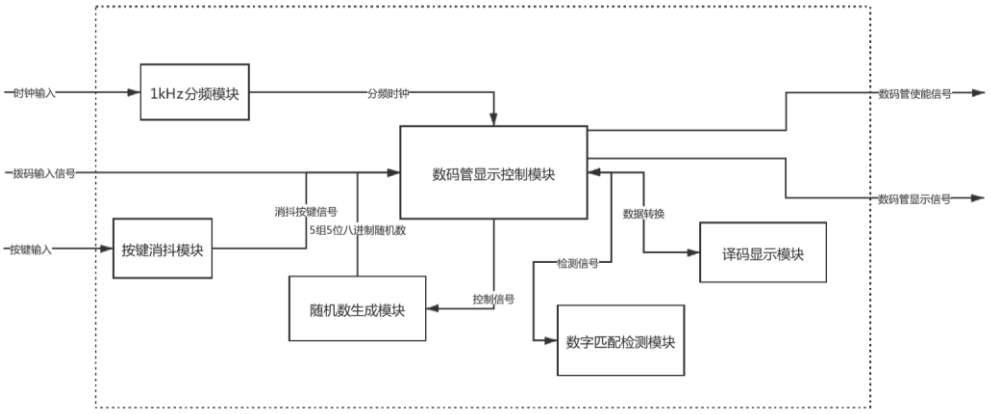
系统功能详细设计

1. 系统主要功能

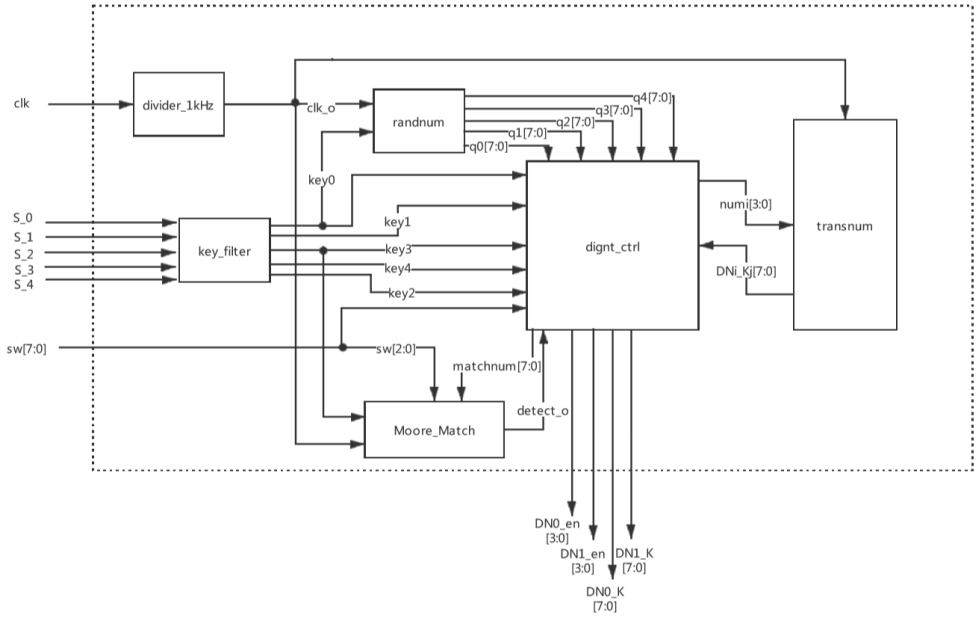
   本系统主要包含以下几个模块：

   a) 1kHz 分频模块：将 100MHz 的输入时钟分频至 1kHz；
   b) 按键消抖模块：用于对输入的 S0、S1、S2、S3、S4 按键进行消除抖动；
   c) 随机数生成模块：当使能信号 en 置高时，系统开始生成一组（5 个）5 位八进制数，系统中使能信号为 S0；
   d) 数字匹配检测模块：用于检测拨码输入的数字是否符合需要匹配的随机数，并输出检测信号；
   e) 译码显示模块：每一位数码管最多需要显示一个 4 位宽数，译码显示模块用于将要显示的 4 位宽数转换为匹配单个数码管的 8 位二进制数；
   f) 数码管显示控制模块（顶层模块）：用于控制键入相应按键后数码管的显示情况，包括开机上电或按下复位键 S4 时需要显示 8 个“0”、按下 S1 按键时播放随机数字、按下 S2 时进入匹配状态显示匹配结果。

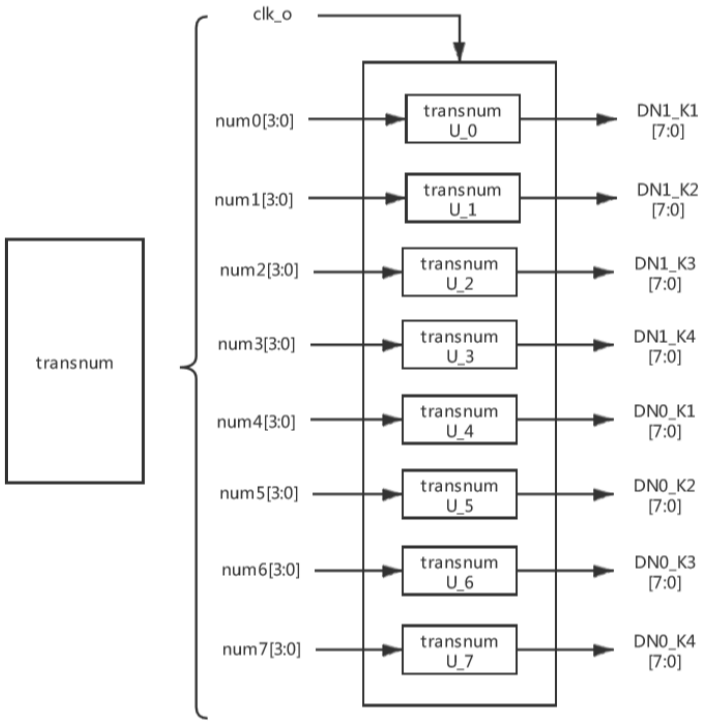
   简要系统框图如下：



使用模块名并表达模块中变量传递关系的框图如下：



其中 transnum.v 模块进行了多次实例化，具体为：



2. 状态分配、状态编码和状态图

在 Moore_match.v 中使用 Moore 型状态机实现了检测数字是否匹配的功能，状态分配为：

a) A0 为初始状态或失配状态；
b) A1 为接收到一位与待匹配数相符的数字；
c) A2 为接收到两位与待匹配数相符的数字；

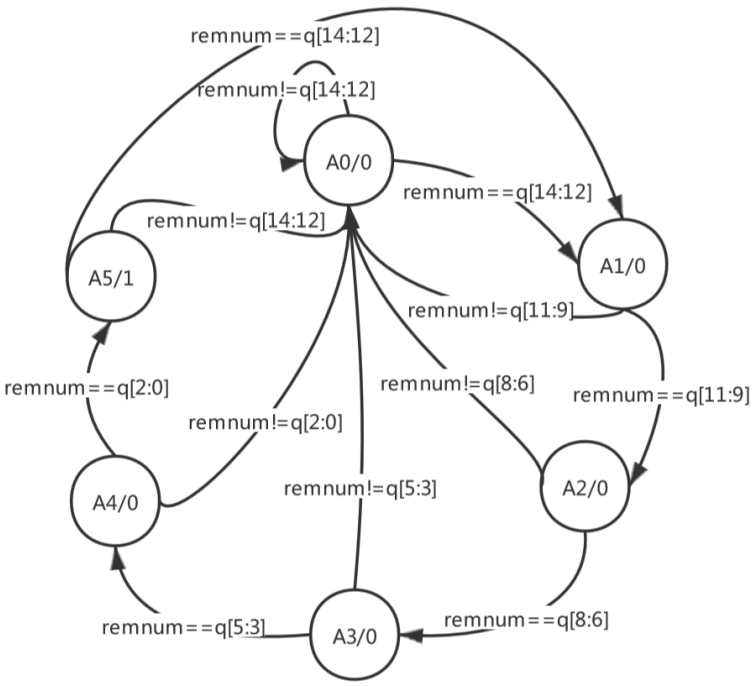
d) A3 为接收到三位与待匹配数相符的数字；
e) A4 为接收到四位与待匹配数相符的数字；
f) A5 为接收到五位与待匹配数相符的数字；

当现态为 A5 时，检测信号 detect_o 输出高电平，否则表现为低电平。

状态编码为：

A0=1'd0; A1=1'd1; A2=2'd2; A3=2'd3; A4=3'd4; A5=3'd5;

状态图为：



模块描述

1. 1kHz 分频模块 divider_1kHz.v

信号名	位宽	属性	功能
clk_i	1	输入	100MHz 的输入时钟
clk_o	1	输出	1kHz 的输出时钟

主要使用一个 27 位宽的 cnt 变量，cnt 在每个时钟上升沿到来之时进行自增操作，当 cnt 计数至 50000 时，输出时钟进行电平变化；当 cnt 计数至 100000 时，cnt 清零进行下一轮计数。

2. 按键消抖模块 key_filter.v

信号名	位宽	属性	功能
clk	1	输入	1kHz 的输入时钟
key	1	输入	按键输入信号
result	1	输出	消抖后的按键信号

使用 count 变量进行计数，延迟时间为 10 个输入时钟。

3. 随机数生成模块 randnum.v

信号名	位宽	属性	功能
clk	1	输入	1kHz 的输入时钟
en	1	输入	使能信号，高有效

q0	15	输出	5 位八进制随机数
q1	15	输出	5 位八进制随机数
q2	15	输出	5 位八进制随机数
q3	15	输出	5 位八进制随机数
q4	15	输出	5 位八进制随机数

采用线性反馈移位寄存器的方法生成随机数，初始 seed 设置为 0。当使能信号为高电平时（即按下按键 S0 时），将寄存的 1 组（5 个）5 位八进制数输出到 q0、q1、q2、q3、q4 中。

4. 译码显示模块 transnum.v

信号名	位宽	属性	功能
num	4	输入	需要进行转换输出的数字
clk	1	输入	1kHz 的输入时钟
led	8	输出	对应数码管显示的二进制数

由于系统中所需要显示的数为八进制数，为了代码简洁考虑，在 0~7 均显示为对应数字的基础上，设置输入为 4’d8 时数码管不显示，4’d9 时数码管显示“F”，4’d10 时数码管显示为“-”（地址-随机数连接符）。

5. 数字匹配检测模块 Moore_match.v

信号名	位宽	属性	功能
clk	1	输入	1kHz 时钟输入信号
key3	1	输入	经消抖后的按键使能信号，高有效
remnum	3	输入	拨码输入数字
q	15	输入	待匹配数字
detect_o	1	输出	检测是否匹配成功，1 为匹配成功

使用 Moore 型状态机进行数字匹配，当 key3 为高电平（按下 S3）时，进行状态检测。

6. 数码管显示控制模块 dignt_ctrl.v（顶层模块）

信号名	位宽	属性	功能
clk	1	输入	100MHz 的输入时钟
S_0	1	输入	按键信号 S0
S_1	1	输入	按键信号 S1
S_2	1	输入	按键信号 S2
S_3	1	输入	按键信号 S3
S_4	1	输入	按键信号 S4
sw	8	输入	拨码开关输入
DN0_en	4	输出	DN0 组数码管使能信号
DN1_en	4	输出	DN1 组数码管使能信号
DN0_K	8	输出	DN0 组数码管显示

			信号
DN1_K	8	输出	DN1 组数码管显示信号

数码管显示控制模块主要采用 if-else 语句进行跳转。模块中将每个七位数码管视作一个用于存储数字的 4 位宽容器(num*i*)，在每个分频时钟上升沿到来时进行判断检测，主要包含以下几种情况：

a) 按下复位键 S4 (key4==1)
此时 8 个数码管应该均显示为“0”，故将每个 num 均赋值为 0。此时，还需将上一次匹配读入的地址和匹配数清零，并对计数变量进行置零操作。

b) 按下 S0 进入随机数生成状态 (flag1==1)
此时应该以 1s 为间隔依次将五个数播放一遍，故将 num5~num7 赋值为 8（不显示数字），而 num0~num4 显示生成的随机数 q。设置计数变量 cnt_1s 对时钟上升沿进行计数，当计数至 1000 时播放下一个 5 位八进制数，当播放至 q4 时，数码管显示不再变化。

c) 按下 S1 进入选择地址状态 (key1==1)
此时 num0 读取拨码输入 sw[2:0]的值寄存于 readsel 中，若小于 5，则数码管显示选中地址，同时读取需要匹配的数寄存在 matchnum 中，否则 num0~num7 赋值为 9（显示“F”）。

d) 按下 S2 进入数字匹配状态 (flag2==1)
此时若是由数字检测匹配模块 Moore_match 反馈的输出变量为高电平，则显示“地址-随机数”，否则使用计数变量 cnt_4Hz 对时钟上升沿进行计数，以达到数码管闪烁 4Hz 的 8 个“0”的效果。

对于数码管显示，使用一个 2 位宽变量 scancnt 进行扫描显示。当扫描至对应编号的数码管时，该数码管的使能信号置 1。每组的数码管显示由数码管使能信号决定。

```
222     always@(posedge clk_o) begin
223         scancnt<=(scancnt>=2'd3)?0:scancnt+1'd1;
224     end
225
226     always@(scancnt) begin
227         case(scancnt)
228             2'd0:begin DN0_en<=4'b0001; DN1_en<=4'b0001; end
229             2'd1:begin DN0_en<=4'b0010; DN1_en<=4'b0010; end
230             2'd2:begin DN0_en<=4'b0100; DN1_en<=4'b0100; end
231             2'd3:begin DN0_en<=4'b1000; DN1_en<=4'b1000; end
232         endcase
233     end
```

管脚分配表

信号	管脚	信号	管脚	信号	管脚
sw[7]	P5	S_3	V1	DN0_K[2]	B3
sw[6]	P4	S_4	U4	DN0_K[3]	A1
sw[5]	P3	DN1_K[0]	H2	DN0_K[4]	B1
sw[4]	P2	DN1_K[1]	D2	DN0_K[5]	A3
sw[3]	R2	DN1_K[2]	E2	DN0_K[6]	A4
sw[2]	M4	DN1_K[3]	F3	DN0_K[7]	B4
sw[1]	N4	DN1_K[4]	F4	DN0_en[0]	H1
sw[0]	R1	DN1_K[5]	D3	DN0_en[1]	C1
clk	P17	DN1_K[6]	E3	DN0_en[2]	C2
S_0	R11	DN1_K[7]	D4	DN0_en[3]	G2
S_1	R17	DN0_K[0]	D5	DN1_en[0]	G6
S_2	R15	DN0_K[1]	B2	DN1_en[1]	E1
		DN0_en[3]	G1	DN1_en[2]	F1

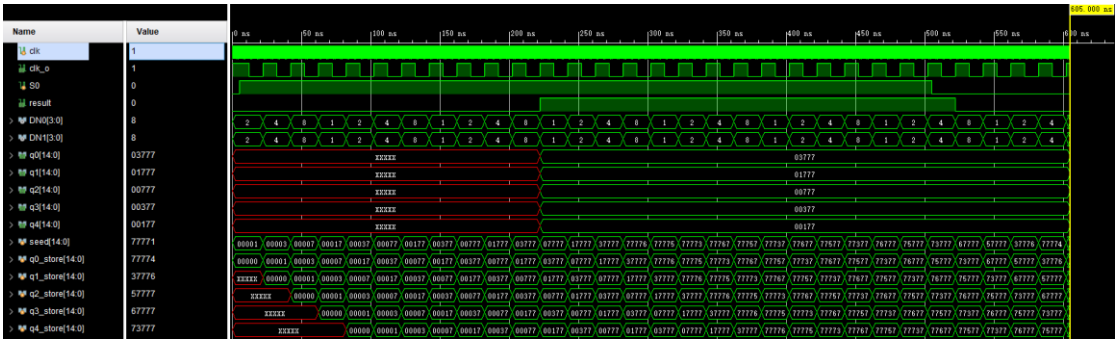
调试报告

仿真说明：仿真时减小了分频时钟计数值便于观察仿真现象。

1. 随机数生成仿真
   a) 仿真代码

```
module numgen_sim( );


    reg clk = 1;
    reg [7:0] sw;
    reg S4=0;
    reg S0=0;
    reg S1=0;
    reg S2=0;
    reg S3=0;
    wire [7:0] DN0_K;
    wire [7:0] DN1_K;
    wire [3:0] DN0;
    wire [3:0] DN1;

    dignt_ctrl U_1(
      .clk  (clk),
      .sw   (sw),
      .S_4   (S4),
      .S_0   (S0),
      .S_1   (S1),
      .S_2   (S2),
      .S_3   (S3),
      .DN0_K  (DN0_K),
      .DN1_K  (DN1_K),
      .DN0_en    (DN0),
      .DN1_en    (DN1)
    );

    always #1 clk = ~clk;

    initial begin
        #5 S0=1;     //按下S0，开始生成随机数
        #500 S0=0;
        #100 $stop;
    end


endmodule
```

b) 仿真截图



c) 仿真分析

上图中，在 5ns 时置 S0 为 1，表示按下 S0，经按键消抖后的信号为 result，在 222ns 时消抖后的按键信号置为高电平，此时存储 5 位八进制数的 q0、q1、q2、q3、q4 从线性反馈移位寄存器的寄存器变量 q0_store、q1_store、q2_store、q3_store、q4_store 中读取伪随机数数据并保持不变。伪随机数数据在时钟上升沿期间持续生成，但仅当按下 S0（result 上升沿）时存

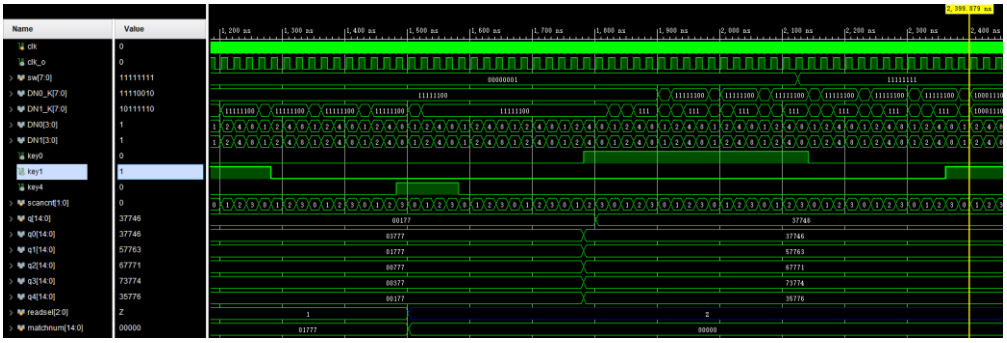
入数据。在 522ns 时，result 变为低电平，寄存器中的随机数值不发生改变。

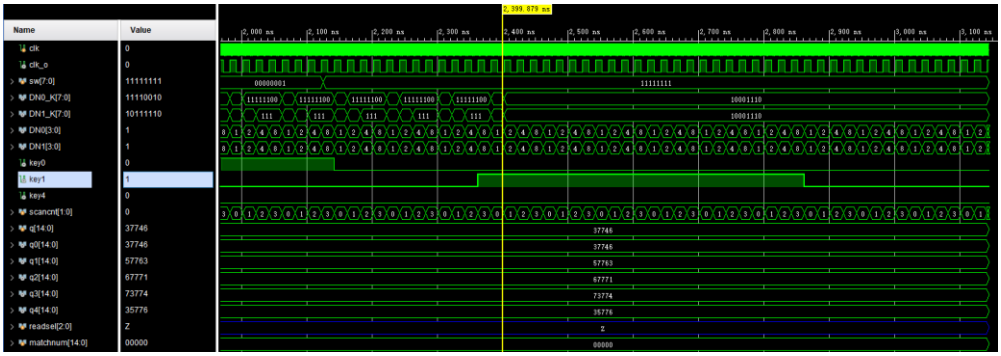
2. 数码管显示和存储器读写功能仿真

a) 仿真代码

```
module rw_sim( );

    reg clk = 1;
    reg [7:0] sw;
    reg S4=0;
    reg S0=0;
    reg S1=0;
    reg S2=0;
    reg S3=0;
    wire [7:0] DN0_K;
    wire [7:0] DN1_K;
    wire [3:0] DN0;
    wire [3:0] DN1;

    dignt_ctrl U_1(
      .clk  (clk),
      .sw   (sw),
      .S_4   (S4),
      .S_0   (S0),
      .S_1   (S1),
      .S_2   (S2),
      .S_3   (S3),
      .DN0_K  (DN0_K),
      .DN1_K  (DN1_K),
      .DN0_en    (DN0),
      .DN1_en    (DN1)
    );

    always #1 clk = ~clk;

    initial begin
        #5 S0=1;    //按下S0，开始生成随机数
        #550 begin S0=0; sw=8'b0000_0001; end//读取存储器
        #20  S1=1;  //按下S1
        #700 begin S1=0; S4=1; end  //复位
        #300 begin S0=1; S4=0; end
        #550 begin S0=0; sw=8'b1111_1111; end
        #20 S1=1;
        #700 S1=0;
        #300 $stop;


    end


endmodule
```

b) 仿真截图





c) 仿真分析

在 222ns 时，key0 置 1，由 LFSR 生成的随机数被取出至 q0、q1、q2、q3、q4 中，数码管显示由 scancnt 和数码管使能信号共同决定，如在 scancnt=3 且 DN1=4’b1000 时，数码管 DN1_K 被赋值为 8’b11111100，显示为“0”。782ns 时，key1 置 1，进入读取地址状态，sw[2:0]=3’b001，故 readsel 在下一个分频时钟上升沿寄存值为 1 并保持，同时 matchnum 中寄存选中的 5 位八进制数“01777”，此时数码管赋值为 8’h60，显示为“1”。

在 1482ns 时，置 key4 为高电平，模拟按下复位键的情况。在下一个时钟上升沿时 readsel 被置为高阻态，同时 matchnum 值被清空。1782ns 时重新置 key0 为高电平，将重新生成一组 5 位八进制数。此次拨码输入地址值置为 sw[2:0]=8’b11111111，则在 key1 按下后，无法正确匹配，8 个数码管均被赋值为 8’h8e，显示为“F”。

3. 数字匹配功能

a) 仿真代码

```
module match_sim();


    reg clk = 1;
    reg [7:0] sw;
    reg S4=0;
    reg S0=0;
    reg S1=0;
    reg S2=0;
    reg S3=0;
    wire [7:0] DN0_K;
    wire [7:0] DN1_K;
    wire [3:0] DN0;
    wire [3:0] DN1;

    dignt_ctrl U_1(
      .clk  (clk),
      .sw   (sw),
      .S_4   (S4),
      .S_0   (S0),
      .S_1   (S1),
      .S_2   (S2),
      .S_3   (S3),
      .DN0_K  (DN0_K),
      .DN1_K  (DN1_K),
      .DN0_en    (DN0),
      .DN1_en    (DN1)
    );

    always #1 clk = ~clk;

    initial begin
        #5 S0=1;    //按下S0，开始生成随机数
        #550 begin S0=0; sw=8'b0000_0001; end//读取存储器
        #20  S1=1;  //按下S1
        #300 S1=0;
        #100 sw[2:0]=3'b000;
        #30 begin S2=1; end

        #300 begin S2=0; S3=1; end
        #300 begin S3=0; sw[2:0]=3'b001; end
        #30 begin S3=1; end
        #300 begin S3=0; sw[2:0]=3'b111; end
        #30 begin S3=1; end
        #300 begin S3=0; sw[2:0]=3'b111; end
        #30 begin S3=1;end
        #300 begin S3=0;sw[2:0]=3'b110; end
        #30 begin S3=1; end
        #300 begin S3=0; end//匹配失败

        #10 begin sw[2:0]=3'b000; end
        #30 begin S3=1; end
        #300 begin S3=0; sw[2:0]=3'b001; end
        #30 begin S3=1; end
        #300 begin S3=0; sw[2:0]=3'b111; end
        #30 begin S3=1; end
        #300 begin S3=0; sw[2:0]=3'b111; end
        #30 begin S3=1;end
        #300 begin S3=0;sw[2:0]=3'b111; end
        #30 begin S3=1; end
        #300 begin S3=0; end    //匹配成功

        #300 $stop;
        //#700 $stop;

    end
endmodule
```

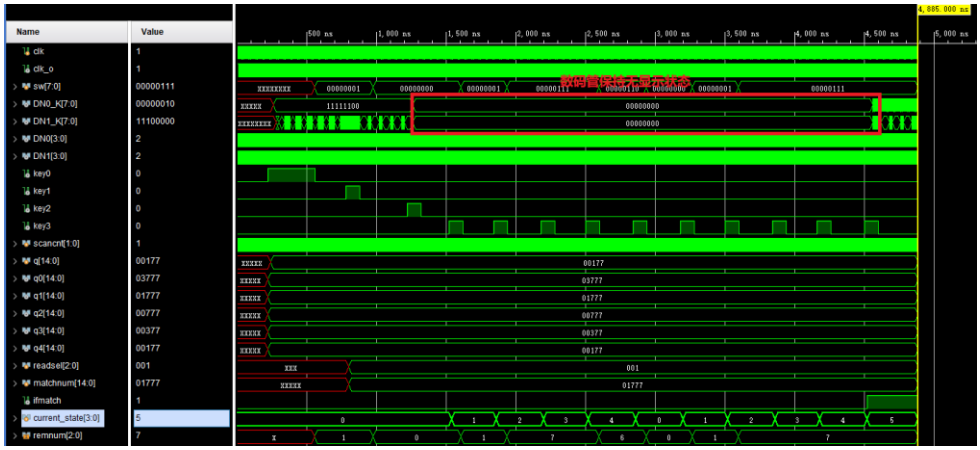
b) 仿真截图

c) 仿真分析

在 222ns 时，kye0 置为高电平，系统读取随机数存入寄存器中，置匹配地址 sw[2:0]=3’b001，待匹配数值 matchnum=15’o01777。1222ns 时，key2 置高，系统进入数值匹配状态。此后在 1522ns、1842ns、2182ns、2502ns 和 2842ns 时，系统分别读取拨码 sw[2:0]的值为“0”“1”“7”“7”“6”，最后一位失配，状态机现态 current_state 由 4 回到 0，检测输出变量 ifmatch 始终保持低电平。在此过程中，数码管保持闪烁 8 个“0”，由计数变量 cnt_4Hz 进行分频。第二轮匹配中，在 3182ns、3502ns、3842ns、4162ns、4502 时，分别读取拨码开关的值为“0”“1”“7”“7”“7”，恰好与待匹配数值一致，故在下一个分频时钟上升沿，状态机现态 current_state 变为 5，检测变量 ifmatch 置高电平，cnt_4Hz 停止计数，数码管显示“地址-匹配成功的数值”。

设计过程中遇到的问题及解决方法

1. 未对计数器变量 cnt_4Hz 赋初值，导致仿真时数码管输出错误。如下图所示：



其中数码管输出在系统进入数值匹配状态后始终保持 8’b00000000，即不显示状态，经检查发现计数变量 cnt_4Hz 始终保持不定值“X”，在给计数变量赋初值后问题解决。

2. 状态无法跳转：在操作时发现仅能进行一次匹配，但在按下复位键之后无法进行下一次匹配，匹配检测变量 ifmatch 会始终保持高电平状态，导致下一次开始匹配时会直接读取相应地址中数值并显示。检查了代码之后发现没有完全理解状态机的跳转条件。由于数字匹配时状态机的跳转条件为按下 S3，故在实际操作中需要再次按下 S3 方能使状态机回到 A0 状态。在实际操作中再次按下 S3 后问题解决，能够重复检测。

课程设计总结

在本实验设计中，我在顶层文件采取了 if-else 判断语句进行状态跳转，事实上这是在尝试状态机失败后才作出的改变。在第一版的设想中，实验中至少需要用到两个状态机，外层状态跳转使用一个状态机进行实现，内层进行数值匹配时使用状态机进行判断。但是在实际编写代码的过程中发现，由于对状态机的理解不到位不全面，出现了各种各样的问题，比如状态无法跳转等，且初版代码由于另外使用了之前实验的寄存器文件等模块，导致模块调用后十分冗余，加之对数码管显示的原理没有完全搞懂，代码过于臃肿难于调试，故推翻重来，采取了条件判断语句进行跳转。原状态机仍作为游离模块置于项目中，并且在当前顶层模块中也保留了第二版仍有问题的状态机的注释代码，今后有机会希望能够重构并完整实现系统功能。

在编写实验代码的过程中，给我带来最大困惑的就是 Verilog 硬件描述语言的特性。由于在之前学习过高级编程语言，且在 Verilog 中也有模块化的概念，故在写代码的过程中会不自觉地将“模块”和“函数”的概念混淆，妄图像调用函数那样去调用模块，而忽视了模块其实是对应着现实中某些电子元件，有着明确的管脚和电路结构，并不是完全的“黑匣子”。上面提到在初版代码中我使用了之前的寄存器文件模块，就是由于我将之视为一个可以任意调用的函数，而忽视了更简洁地描述电路的可能性。此外，起初我还希望将很多在实际运用时直接使用的变量封装起来，比如读取地址输入并确定数码管显示值，在初始设想中我将之单独列为一个模块实现读入数据和数码管赋值的功能；又比如分频这一功能，在本实验中需要分频为 1kHz、4Hz 和 1s，如果按照函数模块化设计的思想，我会倾向于将三个分频都封装为模块便于调用，并且在模块中进行模块的调用，但在设计过程中，实际上只需要分频为 1kHz，再在需要分频为更长时间时使用

计数变量进行计数即可，这样大大降低了模块调用的复杂度，使得结构更加清晰易懂。

给我印象比较深的还有数码管显示这一部分。在实验 4 的数码管控制器设计中，我采取的方法是按管脚约束将数码管分为四个一组共两组，两组中再根据题目要求分为两个一组，但是在初版设计中我却完全忽视了题目中每个数码管相对独立的特性，反而只将 8 个数码管分为了 2 组，这也进一步导致了代码极其冗余并且难于调试。此外，scancnt 变量的使用是控制数码管显示的精髓之一，我对于数码管显示的理解在本实验中也得到了很大的加深。

Verilog 的硬件描述语言特性给我带来另一个印象深刻的点是它的调试过程。虽然在高级编程语言中 debug 就容易让人沮丧，但在编写 Verilog 代码的过程中，就算仿真正确了，也容易在综合过程出现问题。例如多驱动问题，需要根据实际操作表现、代码和网表综合进行排查，而往往错误又十分隐蔽，生成比特流需要的时间对于需要反复调试的代码实在是让人心焦，调试过程使人倍感沮丧。

在如今的设计中，顶层模块仍然十分冗余，虽然实现了完整的系统功能，但在稳定性上肯定是弱于状态机的，如果能将顶层模块进一步划分，并且用状态机包裹，肯定能使系统更加清晰易懂和简洁。

本次实验可以说是集成了前几次实验的重难点进行了综合设计，且在综合的程度上又做了进一步的加深。当时老师在实验前提示可以使用寄存器文件进行随机数的存储和读写，但在实际设计中我对照搬寄存器文件的设计能否得到简洁的电路结构持怀疑态度。总体而言，这是一次非常具有挑战性的实验，作为综合实验来说可以让人得到很大的进步。而对于之前的实验，若是能够照顾到理论课进度不及实验课的情况，数字逻辑实验将会给我留下更美好的回忆。