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

实验报告

开课学期：2020 秋季

课程名称：数字逻辑设计（实验）

实验名称：记忆游戏

实验性质：综合设计型

实验学时：6 地点：T2614

学生班级：计科 5 班

学生学号：190110509

学生姓名：王铭

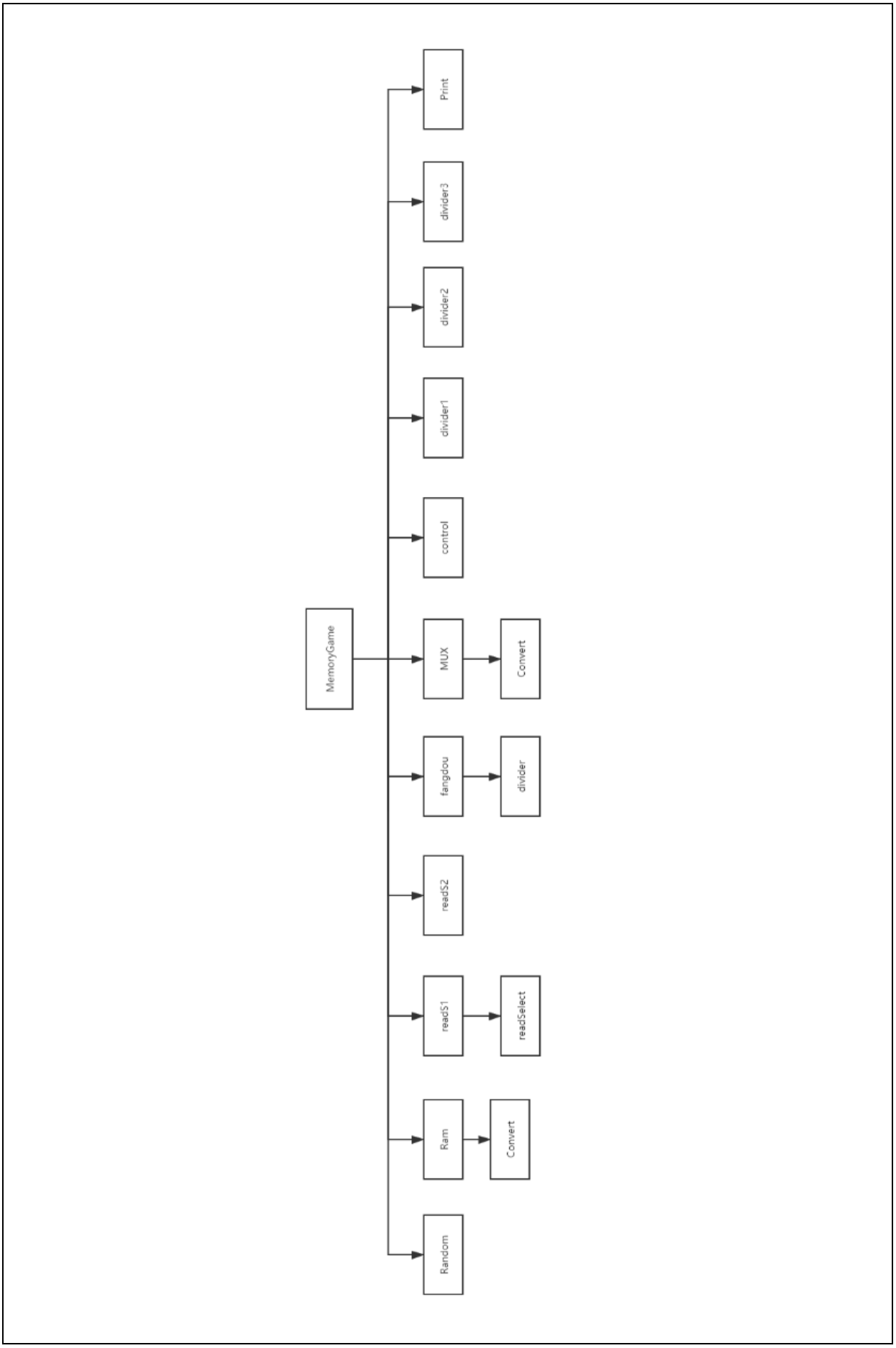
评阅教师：

报告成绩：

实验与创新实践教育中心制

2020 年 12 月

设计的功能描述
基础功能：当玩家按下 S0 键后，能够自动产生五个随机数并显示到数码管上，按下 S1 后，玩家选择想要匹配的随机数地址，按下 S2 确定选择。后用户需要输入五次一个八进制数，每次按下 S3 确认输入，当输入够五次后，若用户输入的数字与选择的随机数相同，则数码管显示对应的“随机数地址-随机数”，否则，数码管显示浮动的零，表示匹配失败。若用户想要重新开始游戏，需按下 S4 复位键，则游戏重新恢复初始状态，等待用户按下 S0 开始产生新一轮的随机数。
系统功能详细设计
主要设计分块： 分频器，随机数，消抖，存储，选择，匹配，数据选择，输出八个模块。 主要功能： 分频功能：divider，divider1，divider2，divider3 消抖：fangdou 随机数：Random 存储：ram 选择：MUX 数据选择：readS1、readSelect 匹配：readS2 输出：convert 和 Print 相互关系图：

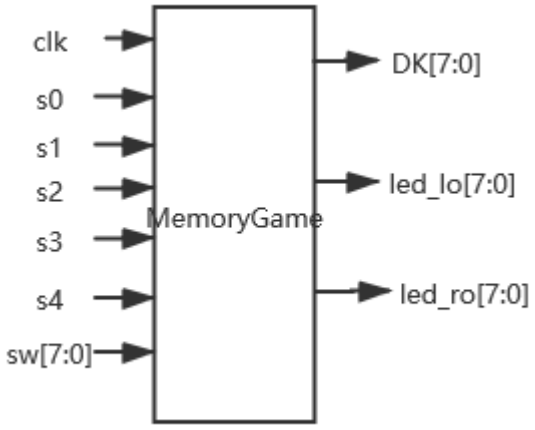

MemoryGame
Print
divider3
divider2
divider1
control
MUX
Convert
fangdou
divider
readS2
readS1
readSelect
Ram
Convert
Random

状态描述：

本设计未采用状态机表示按键的状态，选择用 select、res、flag、i 等标记标明程序执行到的步骤。

模块描述

主模块：



MemoryGame 模块为文件顶层模块，输入 5 个按键信号 s0~s5，时钟信号 clk，以及拨码 sw，输出控制信号 DK，左端 7 段数码管和右端 7 段数码管的信号编码 led_lo 和 led_ro。

主要设计：

MemoryGame 模块通过调用消抖模块，将按键信号过滤。S0 按下时调用随机数模块，生成 5 个随机数，同时调用 Ram 模块，存储生成的随机数。S1 或 S2 按下时，调用 readS1 完成用户的输入选择功能，同时对 Ram 中存储的随机数进行读取。按下 S3 时，由 ReadS2 模块进行匹配，输出匹配的结果。每次输出时用 control 模块控制打印（DK、led_lo、led_ro 变化）的频率，并用 MUX 模块选择当前游戏进行到的模式，选择出应该显示的内容。最后将相关内容通过 Print 模块显示在 7 段数码管中，从而完成该游戏

的设计。

该模块的关键变量有：

rand:当前随机数生成器正在生成的随机数

rand1~rand5:生成的 5 个随机数

set_0~set_5:消抖后的 s0~s5

i:控制生成随机数的变量，i 为 1~5 代表生成第 1~5 个随机数

select:用户按下 S1、S2 后的选择，由 readS1 决定

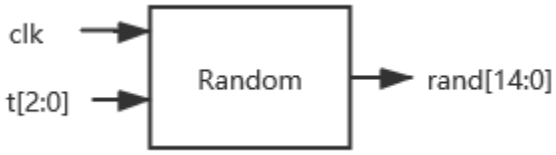
select_num:用户选择的随机数

res:用户匹配之后的结果，1 表示失败，2 表示成功，由 readS2 选择

buffer:要输出到数码管上显示的内容，由 MUX 选择

注：上述变量 i、res、select 均从 1 开始表示有效，0 为初始值，程序中不会用到。

Random 模块：



输入变量有时钟信号 clk，顶层模块的 i，输出为 15 位的随机数。

该模块的设计思路：

采用给移位寄存器增加反馈，完成状态转移，同时取状态机的最后一位作为随机数的第 i 位输出。

关键变量：

i:控制状态机的最后一位赋值给生成随机数的第几位，初值为 0，在生成随

机数阶段，随时钟的上升沿每次加一，直到 f。

flag:1 表示随机数模块正常工作

x:存储上一个状态的输入变量 t

输入的 t 变量，由于 t（即顶层文件的 i）为 0 表示初始状态，故该模块判断随机数是否开始生成的标准为，x< t。当用户在未生成 5 个随机数时再次按下 S0 时，x 会大于 t，此时将 i 变成 0，表示重新生成。

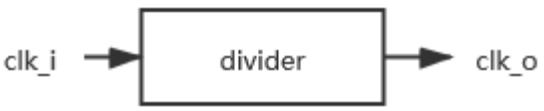
now_state:状态机的现态

next_state:状态机的次态

状态机转换的逻辑关系如下：

```
next_state[7] <= now_state[7]^now_state[5]^now_state[4]^now_state[3];
                next_state[6] <= now_state[5]^now_state[4];
                next_state[5] <= now_state[4]^now_state[3];
                next_state[4] <= now_state[3]^now_state[2];
                next_state[3] <= now_state[2]^now_state[1];
                next_state[2] <= now_state[3];
                next_state[1] <= now_state[2];
                next_state[0] <= now_state[1];
```

divider：



输入要分频的时钟信号 clk_i,输出分频为 1ms 的时钟信号 clk_o，采用计数

器分频，较为简单

divider1:

输入要分频的时钟信号 clk_i,输出分频为 1s 的时钟信号 clk_o，采用计数器分频，用于控制 1s 生成 1 个随机数（clk_o 每次上升沿时，顶层模块的 i 加 1）

divider2:

输入要分频的时钟信号 clk_i,输出分频为 1ms 的时钟信号 clk_o，采用计数器分频，用于利用人眼的特性，在短暂时间内人眼分辨不出来数码管显示时发生的快速变化，可以用来控制 DK 的输出。

divider3:

输入要分频的时钟信号 clk_i,输出分频为 0.25s 的时钟信号 clk_o，采用计数器分频，主要用于在玩家匹配失败时，输出浮动的 0。

control:

输入信号：

clk_dk1 为 1ms 的时钟信号

clk_dk2 为 0.25s 的时钟信号

res 为匹配的结果

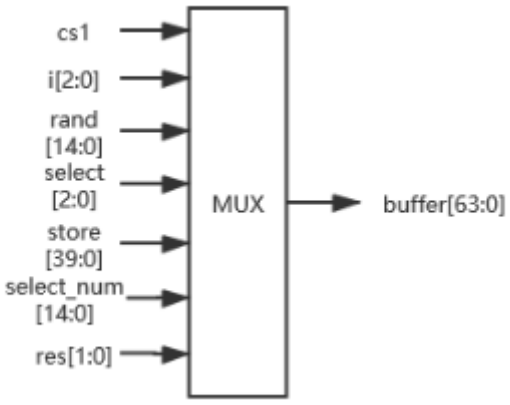
set_4 为消抖过滤后的复位键。

输出 cnt 为控制 DK 变化的计数器

主要设计：

根据 res 判断 cnt 变化的频率，当 res 不是 1（即匹配未失败时），控制 cnt 变化的应该为 clk_dk1,否则为 clk_dk2

MUX：



输入信号：

cs1 为顶层文件中的 flag，表示生成随机数的信号

i 为生成的随机数序号

rand 为正在生成的随机数

select 为用户的选择

store 为存储第五个随机数的 7 段数码管表示

res 为匹配的结果

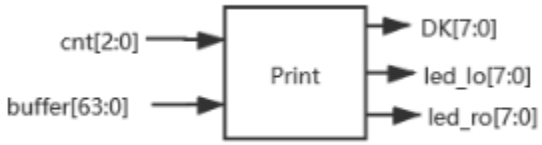
输出信号：

buffer 为 8 个数码管该输出的内容

主要设计：

分 3 个大的情况，①res 为 0，此时根据 select 是否为 0 判断是否处于随机数的生成阶段，输出相应的随机数编码，否则输出选择。②res 为 1，匹配失败，输出 8 个 f 的编码。③res 为 2，匹配成功，输出“地址-随机数”的编码。

Print：



输入信号：

cnt 为控制显示的频率的计数器

buffer[63:0]对应 8 个数码管的输出

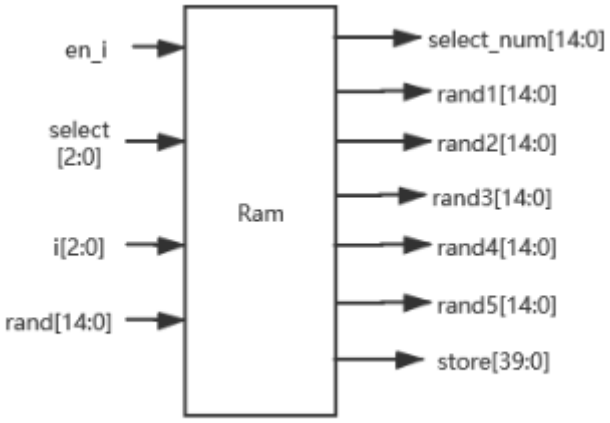
输出信号：

DK 为选择信号，led_lo 为左四个的输出，led_ro 为右四个的输出

主要设计：

根据 cnt 的值，DK 的值和 led_lo、led_ro 的值。

Ram：



输入信号:

en_i 为写使能信号，为顶层模块的 flag（当处于生成随机数阶段时为 1）

select 为用户的选择，根据 select 进行读随机数

i 为当前生成的随机数次序

rand 为当前生成的随机数

输出信号：

rand1~rand5 为生成的 5 个随机数

select_num 为用户选择的随机数

store 用于向数码管打印第 5 个随机数

主要设计：

当使能端有效时，ram 进行写功能，根据 i 的值，将当前生成的随机数赋值给 randi。当 i 为 5 时，由于数码管要保留显示第 5 个随机数，故用 store

保存这 5 位随机数转换为 7 段数码管输出时的值。

否则，判断 select 的值，确定用户选择的数字，赋值给 select_num

fangdou:

输入信号：

clk 为时钟信号，s0 为要消抖的信号

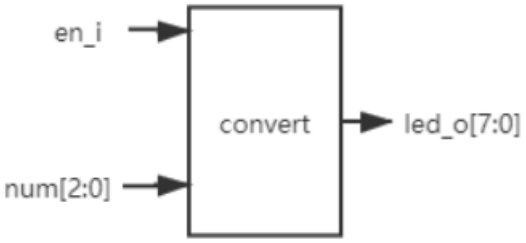
输出信号：

s1 为消抖后的信号

主要设计：

用一个 1ms 的时钟信号，在检测到 s0 变化后的第 5ms 对信号采样，若信号仍发生了变化，则此次按键有效，输出 s1，否则认为此次按键无效。

convert:



输入信号:

en_i 为使能信号

num 为要转换成 7 段数码管输出的数字

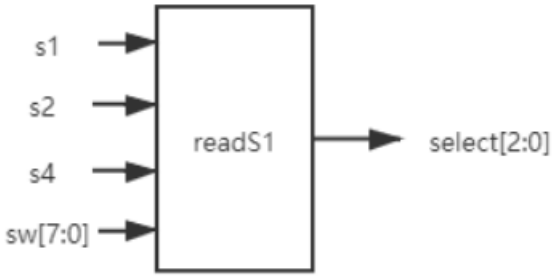
输出信号：

输出该数字转换为 7 段数码管输出后的结果。

主要设计：

主要用“查表法”，判断输入的数字，选择相应输出，较简单。

readS1:



输入信号：

s1,s2,s4 为消抖后的按键，sw 为选择时的输入

输出信号：

select 为做出的选择

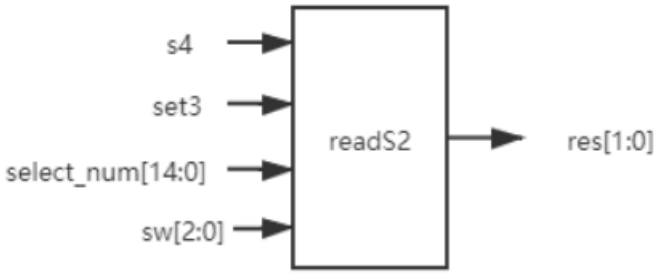
关键变量：

flag：为 0 表示未按下，为 1 表示 s1 按下，为 2 表示 s2 按下

主要设计：

根据按键情况设置 flag 的值，后调用 readSelect 模块，选择 select 的输出值。

readS2:



输入信号：

s4、s3 为消抖后的按键 s4、s3,select_num 为选择的数字，sw 为每次输入的一位 8 进制数。

输出信号：

匹配的结果，成功为 2，失败为 1，否则为 0

主要设计：

用 s3 为时钟，控制一个计数器，5 次输入后，进行匹配，返回匹配结果 res

readSelect:

输入信号：

sw 为输入的选择，flag 为记录按键的标记

输出信号：

select 为做出的选择。

主要设计：

根据 flag 的值判断 s2 是否按下，按下后 select 不变，否则判断 s1 是否按下，按下后 select 若小于 4 则输出选择，否则输出 6 表示此时应该输出 8 个 f，若未按下 s1、s2，flag 为 0 则 select 为 0，否则 select 不变。

管脚分配表

clk	s0	s1	s2	s3	s4
P17	R11	R17	R15	V1	U4
sw[0]	sw[1]	sw[2]	sw[3]	sw[4]	sw[5]
R1	N4	M4	R2	P2	P3
sw[6]	sw[7]	DK[0]	DK[1]	DK[2]	DK[3]
P4	P5	G6	E1	F1	G1
DK[4]	DK[5]	DK[6]	DK[7]	led_lo[0]	led_lo[1]
H1	C1	C2	G2	B4	A4
led_lo[2]	led_lo[3]	led_lo[4]	led_lo[5]	led_lo[6]	led_lo[7]
A3	B1	A1	B3	B2	D5
led_ro[0]	led_ro[1]	led_ro[2]	led_ro[3]	led_ro[4]	led_ro[5]
D4	E3	D3	F4	F3	E2
led_ro[6]	led_ro[7]				
D2	H2				

调试报告

仿真代码：

```
`timescale 1ms / 1ns

module MemoryGame_sim();

reg clk = 0;

reg set_0 = 0;

reg set_1 = 0;

reg set_2 = 0;
```

```
reg set_3 = 0;
reg set_4 = 0;
reg [7:0]sw;
wire [7:0]DK;
wire [7:0]led_lo;
wire [7:0]led_ro;
MemoryGame
game(clk,set_0,set_1,set_2,set_3,set_4,sw[7:0],DK[7:0],led_lo[7:0],led_ro[7:0]);
always #0.000005 clk = ~clk;
initial
begin
    #0 set_0 = 0;
    #6 set_0 = 1;
    #6 set_0 = 0;//模拟按下 s0
    #50 sw[7:0] = 8'b0000_0001;//生成随机数并输入选择
    #5000 set_1 = 1;
    #6 set_1 = 0;//按下 set1
    #6 set_2 = 1;
    #6 set_2 = 0;//按下 S2 确定选择
    #6 sw[2:0] = 3'b011;//输入 5 次 8 进制数，匹配不成功
    #6 set_3 = 1;
    #6 set_3 = 0;
```

```
#6 sw[2:0] = 3'b100;
#6 set_3 = 1;
#6 set_3 = 0;
#6 sw[2:0] = 3'b101;
#6 set_3 = 1;
#6 set_3 = 0;
#6 sw[2:0] = 3'b110;
#6 set_3 = 1;
#6 set_3 = 0;
#6 sw[2:0] = 3'b010;
#6 set_3 = 1;
#6 set_3 = 0;
#20 set_4 = 1;
#6 set_4 = 0;
#10 set_0 = 0;
#6 set_0 = 1;
#6 set_0 = 0;//按下 s0
#50 sw[7:0] = 8'b0000_0011;//生成随机数并输入选择
#5000 set_1 = 1;
#6 set_1 = 0;//按下 set1
#6 set_2 = 1;
#6 set_2 = 0;//按下 S2 确定选择
```

```
        #6 sw[2:0] = 3'b001;//输入 5 次 8 进制数，匹配成功
        #6 set_3 = 1;
        #6 set_3 = 0;
        #6 sw[2:0] = 3'b110;
        #6 set_3 = 1;
        #6 set_3 = 0;
        #6 sw[2:0] = 3'b100;
        #6 set_3 = 1;
        #6 set_3 = 0;
        #6 sw[2:0] = 3'b111;
        #6 set_3 = 1;
        #6 set_3 = 0;
        #6 sw[2:0] = 3'b010;
        #6 set_3 = 1;
        #6 set_3 = 0;
        end
endmodule
```

仿真分析：

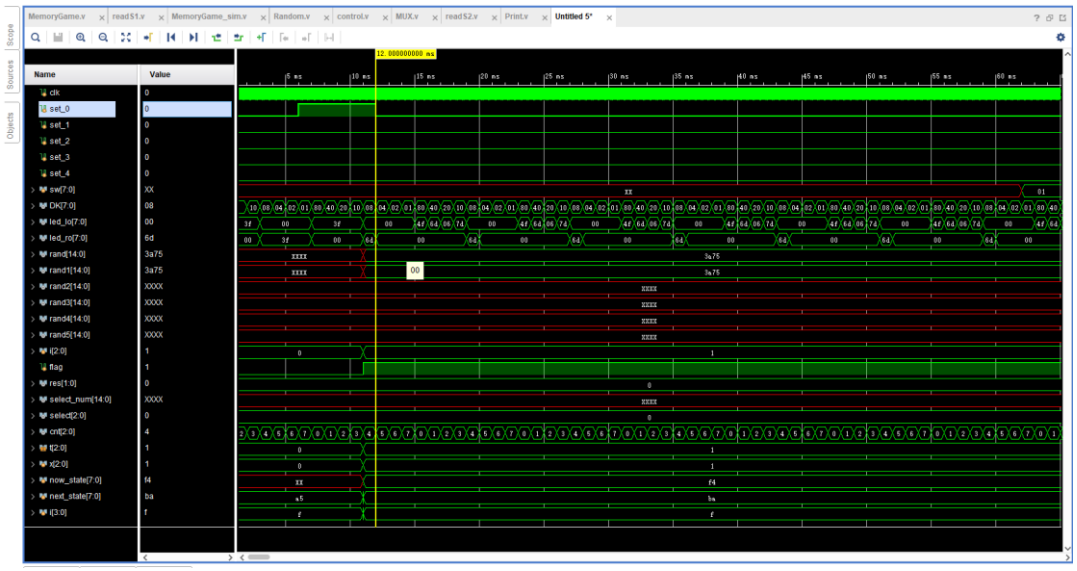


图 1

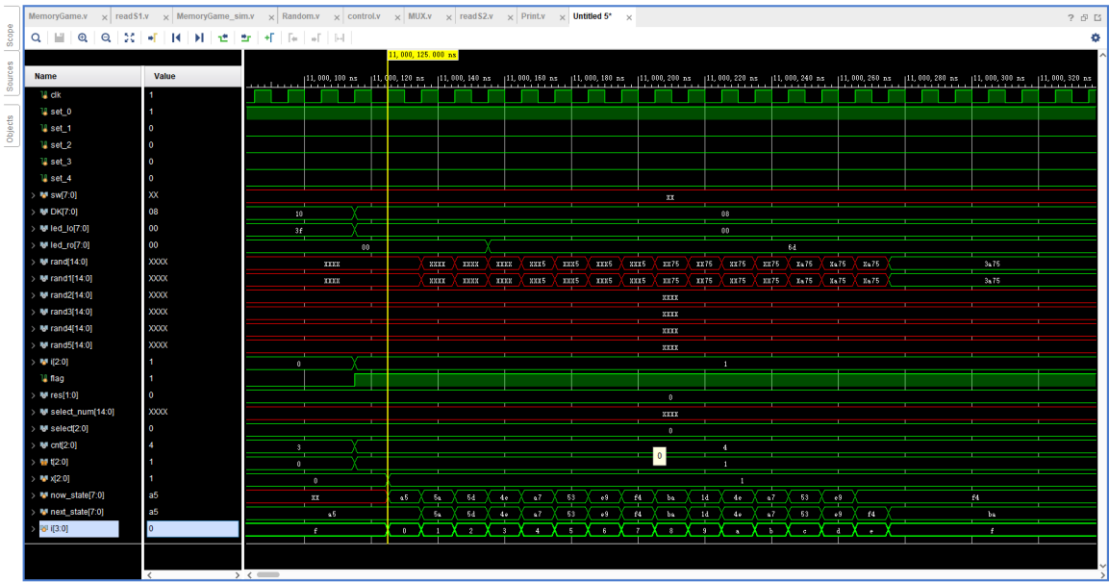


图 2

如图 1 所示，12ms 时，模拟 S0 按下后，随机数模块开始生成随机数。随机数的生成选择使用状态机生成，按照表达式给出的逻辑，控制现态转换成次态的方式，每次时钟的上升沿会导致状态机的一次变化。同个上升沿时，取状态机现态的末位赋值给随机数的一位，当开始生成一个随机数时，取状态机的状态的前 15 次变化（用 i 反映），当 i 处于[0,14]区间时，

把状态机的现态末位赋给随机数的第 i 位，当 i 取 15（f）时，不再赋值。图 2 中，now_state 和 next_state 分别表示现态和次态，每个时钟上升沿时，状态发生状态转移，其中 i 用来控制赋值，结果正确。

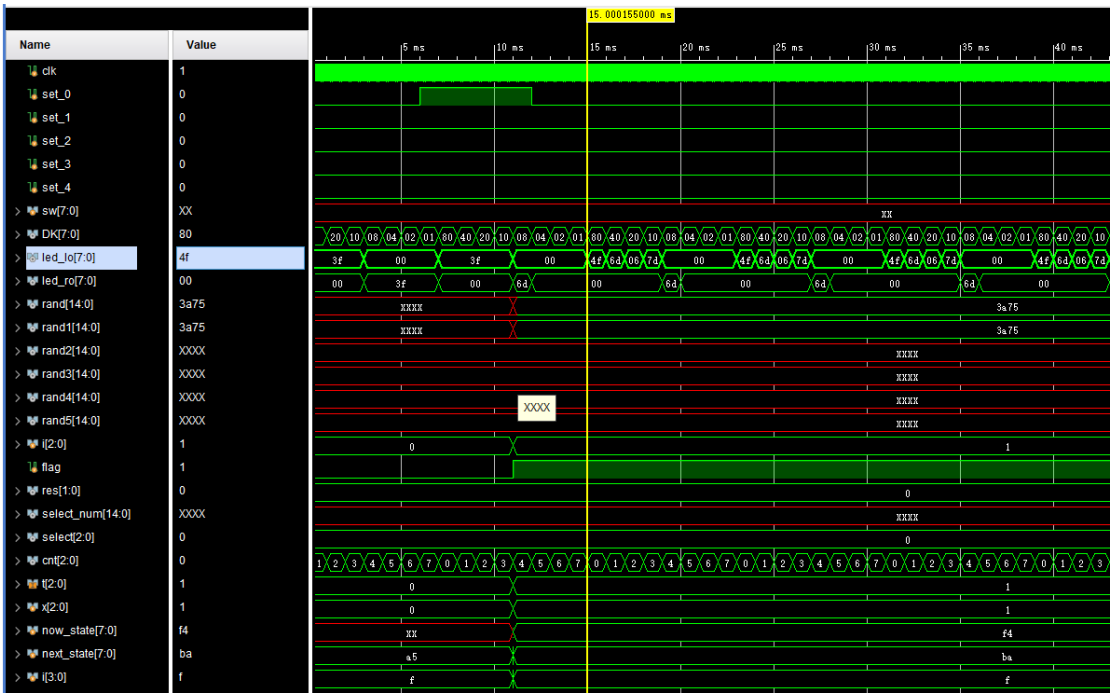


图 3

如图 3 所示，在生成随机数的同时，将此时生成的随机数转换成 7 段数码管的表示形式，并打印到数码管上，其中 DK[7:0]控制选择数码管。led_lo 和 led_ro 显示 7 段数码管输出的内容，其中根据设计，DK 为 80,40,20,10 时，led_lo 显示生成随机数的前 4 位，DK 为 08 时，led_ro 显示生成随机数的第 5 位，而数码管的后 3 位不亮，即 DK 为 04,02,01 时，led_ro 为 0 仿真结果均正确。

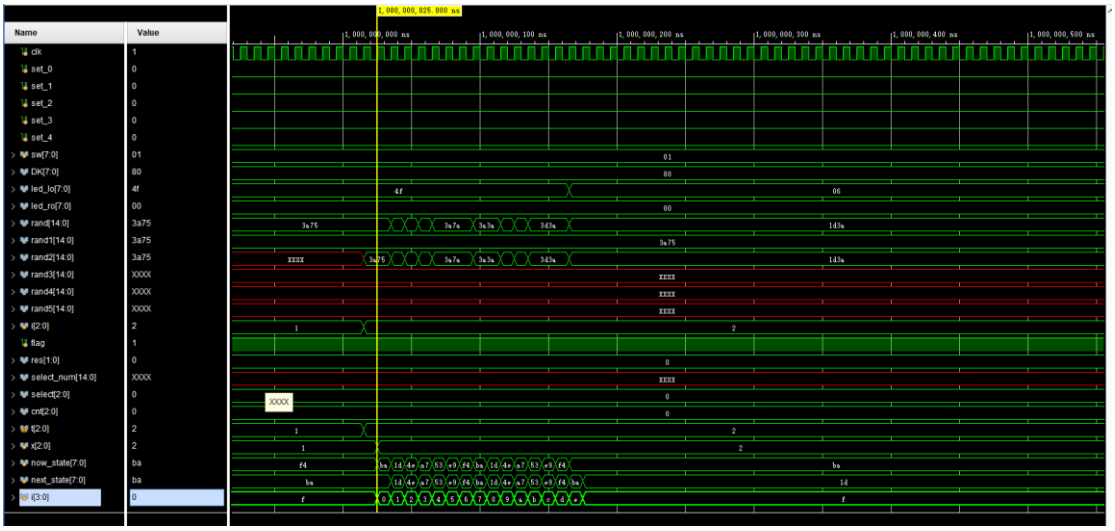


图 4

如图 4 所示，在 1s 左右，当下一个随机数要生成时，i 会自动变 0。

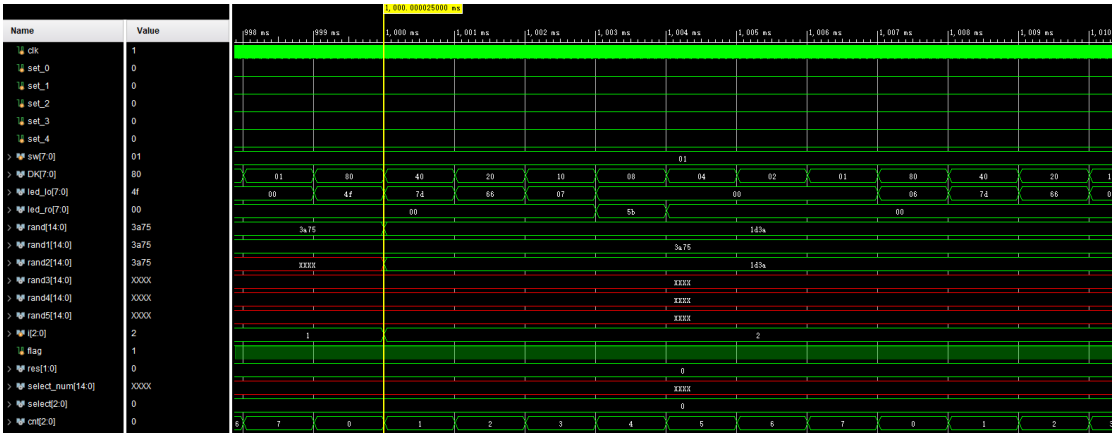


图 5

DK 及 led_lo,led_ro 的控制信号由 cnt 的值控制，即 DK 变化的 cnt 的频率相同，如图 5 所示。

重复上述过程，直至 5s 后生成 5 个随机数。

图 6

如图 6 所示，生成随机数时，由顶层文件的 i 信号和 flag（写使能信号）控制 Ram 模块将随机数写入 rand1~rand5。

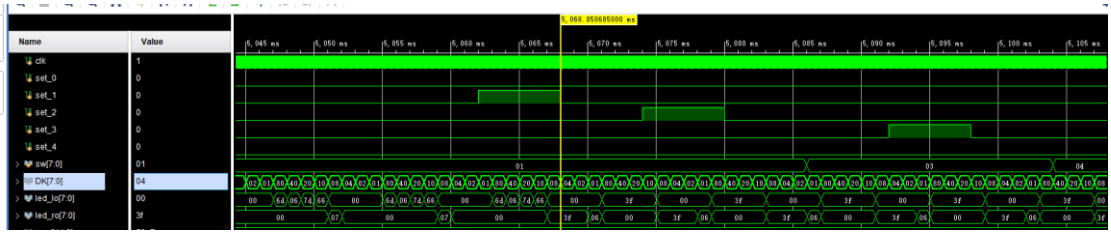


图 7

如图 7 所示，在生成 5 个随机数后，模拟 S1 按键，当按下 S1 后（5068ms），数码管显示 SW[7:0]的选择,select[3:0]表示选择，仿真的 led_lo,led_ro 显示均正确。

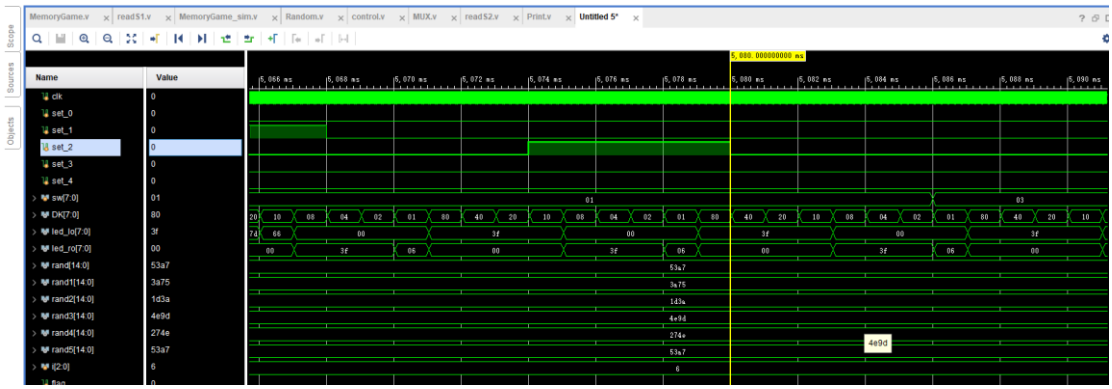


图 8

如图 8 所示，按下 S2 后确认选择，此时 select 和数码管的显示均保持不变，仿真正确。

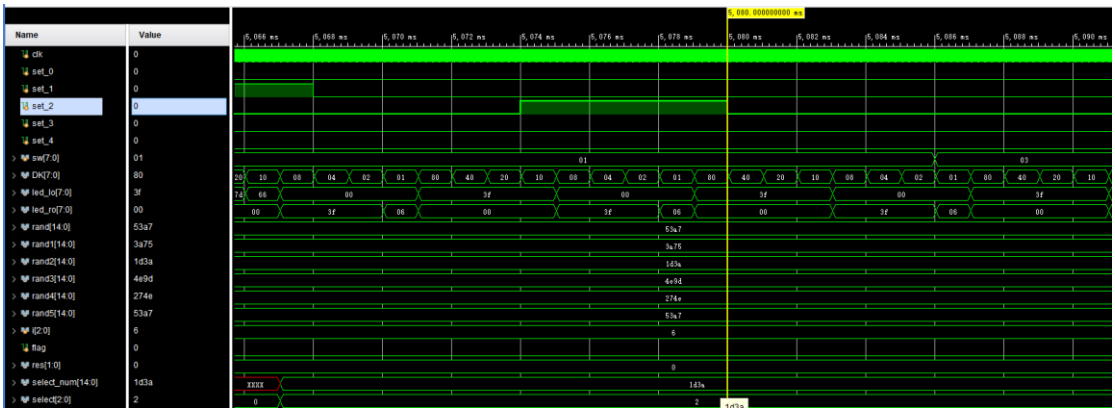


图 9

如图 9 所示，按下 S2 后，根据 select（即读信号）的值，在 ram 中读取选择的随机数到 select_num 中，select 为 1，则选择的是第二位随机数，故 select_num 取 rand2（1d3a）以供匹配。

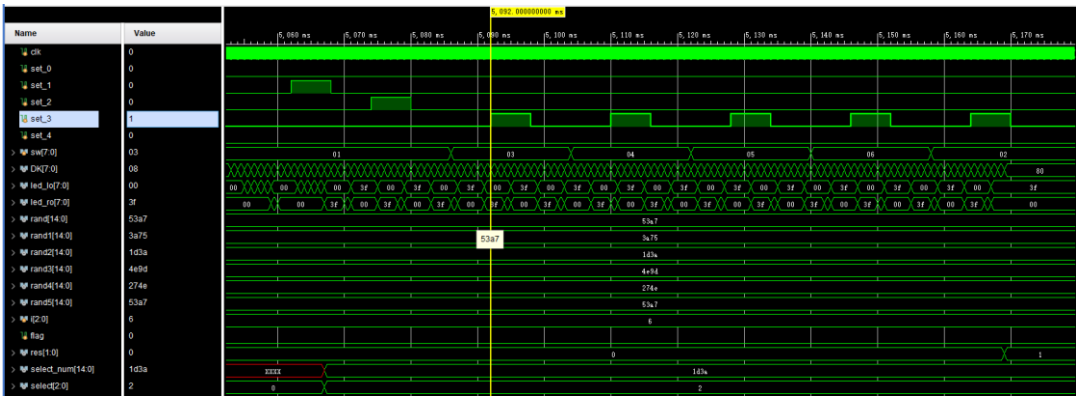


图 10

接着，如图 10 所示，模拟按下 5 次 S3，每次由 SW[2:0]输入一位八进制数，与上一步选择的随机数进行匹配。

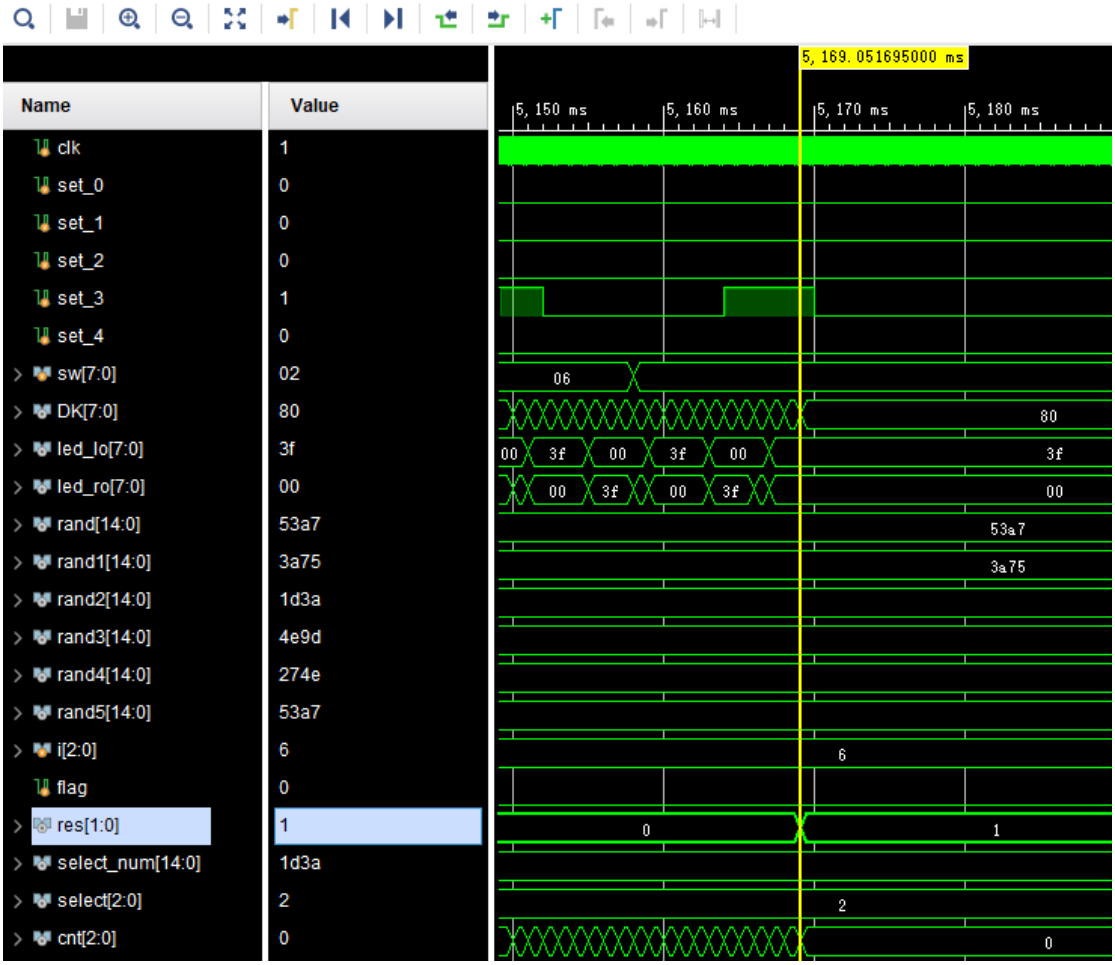


图 11

如图 11 所示，在第一次匹配时，由于输入的 5 个八进制数组成的 5 位数与选择的随机数不同，res 变为 1，表示匹配失败。此时，cnt 以及 DK、led_lo,led_ro 的显示频率变慢，显示浮动的 0，结果正确。

图 12

如图 12，按下复位键 S4（5190ms），开始第二次游戏，顶层文件的 i 置 0，select 置 0

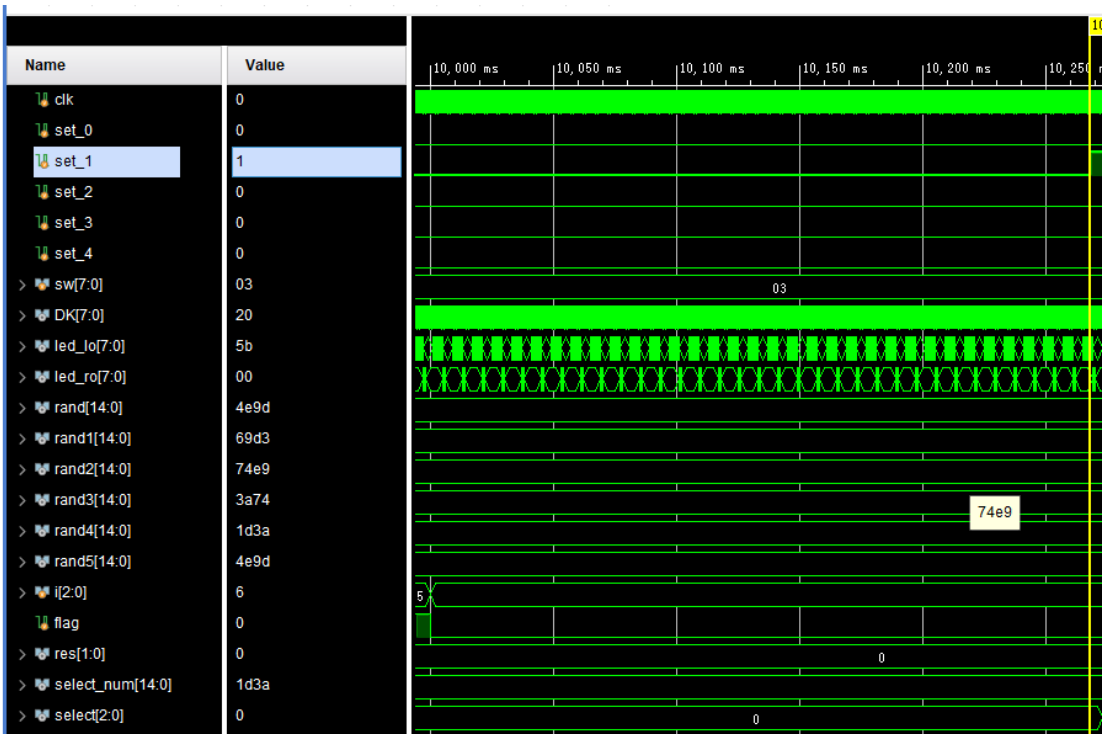


图 13

如图 13 所示，5s 后生成新一轮随机数。

图 14

如图 14 所示，选择第四个随机数，仿真模拟按下 S1，S2，select 变为 4，根据 select 从 Ram 中读取第 4 个随机数赋给 select_num(1d3a)，仿真正确。

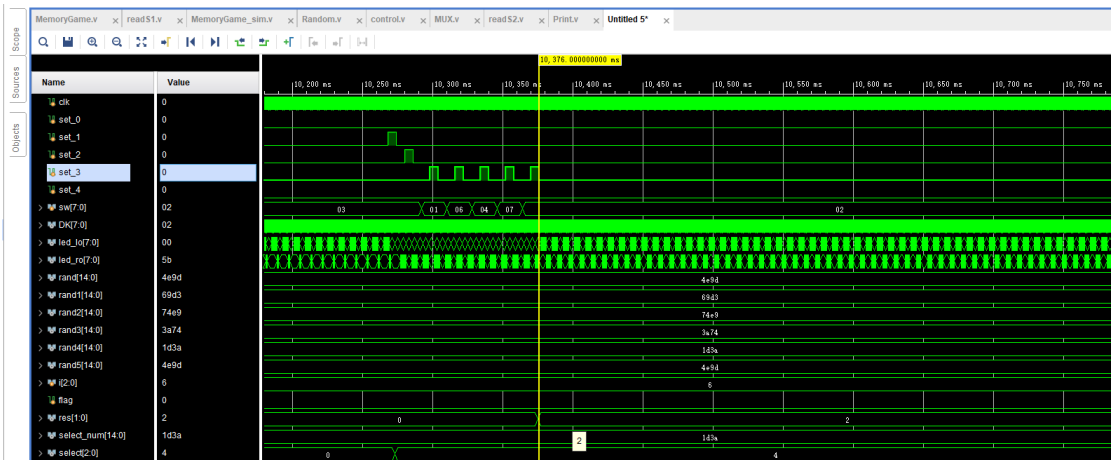


图 15

如图 15 所示，同样地，输入 5 位八进制数，与选择的随机数进行匹配，此次匹配成功，输出 res 为 2。

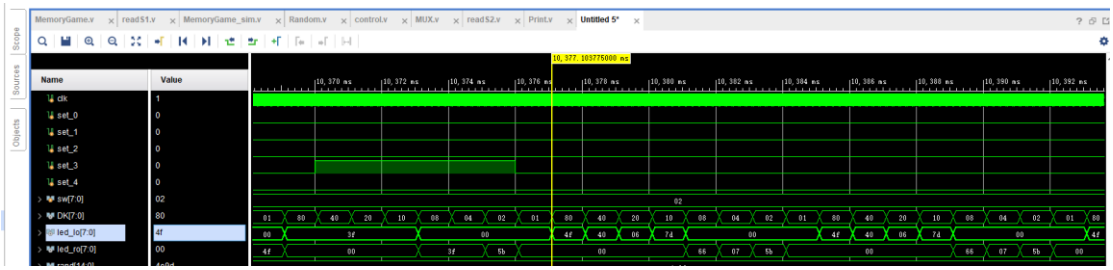


图 16

如图 16 所示，此时数码管显示为“选择-随机数”，仿真结果正确。

设计过程中遇到的问题及解决方法

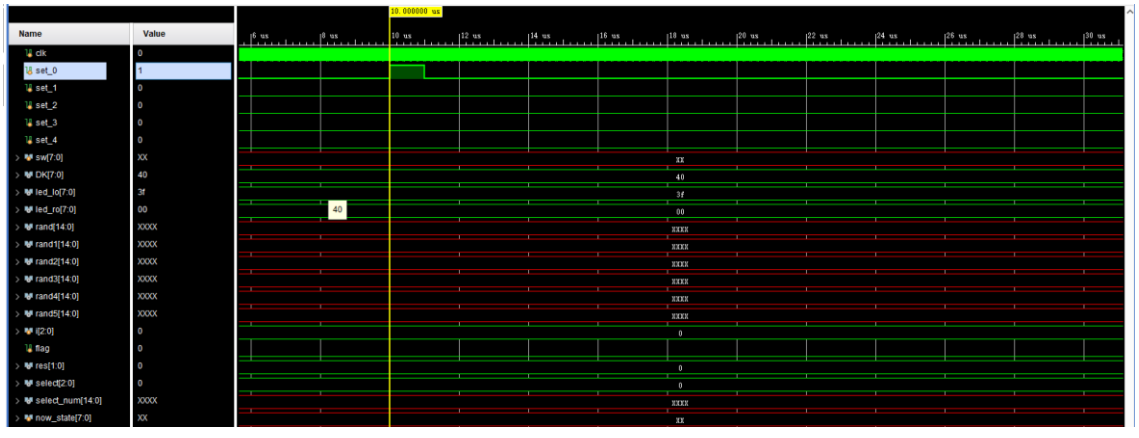
仿真中遇到的问题：

未考虑消抖功能的影响，模拟按键的时间太短，导致按键被消抖，即模拟按键失败，随机数无法生成。

解决方法：

消抖模块采用延时消抖的方法，其中取样的时间为 5ms，则在仿真文件中模拟按键按下时，信号按下的时间置为 6ms

错误的仿真截图：



仿真较为顺利，除出现此错误外并未出现其他问题。

课程设计总结

包括设计的总结和还需改进的内容以及收获

设计总结：

本次设计采用的思想偏编程化，在完成设计时选用了标志表示状态，初版设计时仿真、上板均较为顺利，初版的耦合度较高，故我选择优化程序结构，增加多个模块使该设计更偏模块化。

还需改进的内容：

由于设了许多标志位，使得代码看起来更为冗杂，可以考虑用状态机表示按键的各个状态。同时，在分频模块时，并未集成一个模块，在需要不同频率的时钟信号时，选择用了多个模块，可以改进。在随机数的生成模块中，我选择用带反馈的移位状态机生成随机数，这种随机数的伪随机的程度太高，可以进一步改进。最后，模块的命名规范上仍待改进，如 readS1 可以改成 Select，readS2 可改为 Match。

收获：

本门实验课程对我的理论学习有很大的帮助，通过实验上班，将较为抽象的理论知识变得具体化，使我掌握的更牢固。同时，提高了自己的解决难题的能力，锻炼了自己的编程与硬件相结合的思维模式。