摘要

本文将介绍如何用 ISE 去设计一个打地鼠游戏。如何用画原理图和写 verilog 代码的方式去完成需要的模块，对于每一个模块如何去设计仿真代码来验证该模块是否达到自己想要的结果。以及在设计时遇到错误应如何通过看仿真代码来更正错误。

关键词：打地鼠 模块 ISE Verilog

目录

第 1 章 绪论.....4
1.1 打地鼠设计背景.....4
1.2 主要内容和难点.....4
第 2 章 打地鼠设计原理.....4
2.1 设计内容.....4
2.2 设计方案.....4
2.3 硬件设计.....5
2.3.1 顶层模块.....5
2.3.2 Time_.....5
2.3.3 counter_32bit_rev.....6
2.3.4 counter_1s.....6
2.3.5 Random.....7
2.3.6 Press、Calculate 和 delay.....7
2.3.7 hamster_1 和 hamster_4.....8
2.3.8 Score_.....9
2.3.9 Bin2BCD_Add3 和 Bin2BCD16.....10
第 3 章 打地鼠设计实现.....12
3.1 实现方法.....12
3.2 实现过程.....13
3.3 仿真与调试.....13
3.3.1 Time_.....13
3.3.2 Bin2BCD16.....14
3.3.3 Press.....15
3.3.4 Calculate.....16
3.3.5 hamster_1.....17
第 4 章 系统测试验证与结果分析.....18
4.1 功能测试.....18
4.2 技术参数测试.....19
4.3 结果分析.....19
4.4 系统演示与操作说明.....19
第 5 章 结论.....20

第1章　绪论

1.1 打地鼠设计原因

当初想做一个打地鼠的游戏是因为这款游戏是平常我们最常看见的游戏，而打地鼠正可以代表我们 90 后一款相当有代表性的街机游戏，因此在这次的大程我就想说透过这学期所学的 ISE 软件用 Verilog 语言以及数字逻辑图来实现这个游戏。

1.2 主要内容和难点

任务是要设计输入、计时、计分、地鼠和输出。我认为整个大程设计的难点是要如何把各自不同功用的模块统整起来，当然其中各自模块有各自的难点，因此没做出一个功能的模块后都要先下板子进行验证等，等到所有子模块验证完后再全部合起来变成一个完整的大型游戏。

第 2 章 打地鼠设计原理

2.1 设计内容

这次的大程设计通过 ISE 软件进行设计，并在最后去实验室吓到 SWORD 板子上进行运行验证。在整个程序的模块中，除了用 verilog 语言来代码设计之外，还使用逻辑电路图的方式来设计，所以对于这些知识点都需要深入了解。

2.2 设计方案

整个大程主要分为五个部份，输入部份、地鼠部份、计时部份、计分部份和输出部份。

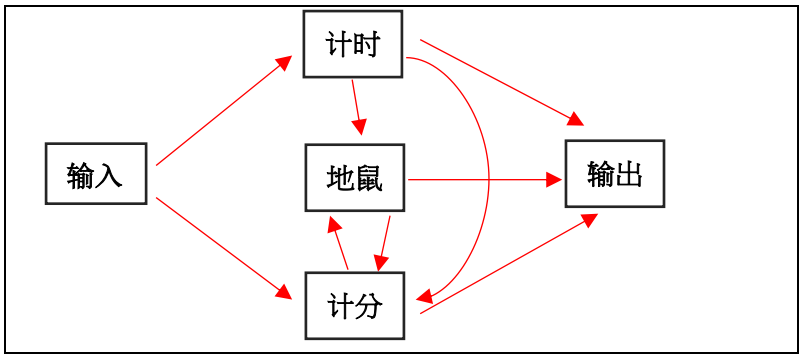

图 1 系统设计图

2.3 硬件设计

2.3.1 顶层模块

输入部分包括 SAnti_jitter、SEnter_2_32 和 clkdiv，输出部分包括 Display、Seg7_Dev 和 PIO，计时部分包括 counter_1s、counter_32bit_rev、Time_和 Bin2BCD16，计分和地鼠部分包括 Random、hamster_4、Bin2BCD16 和 Score。

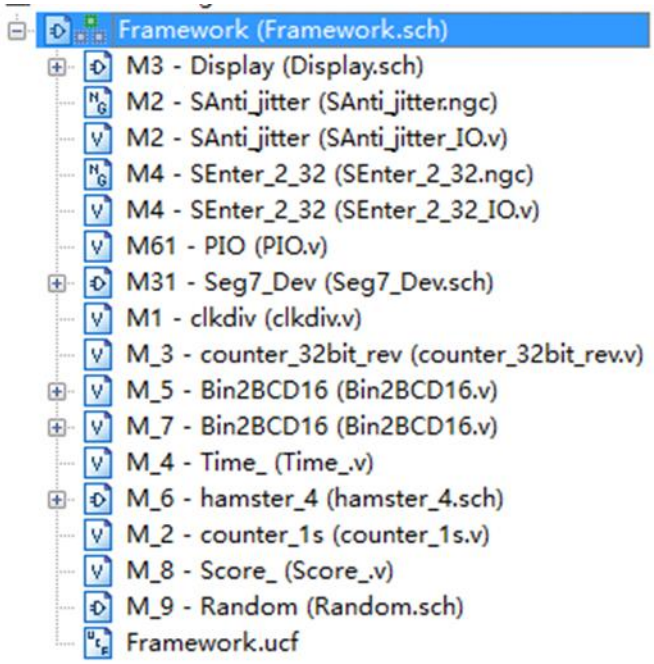

图 2 模块图

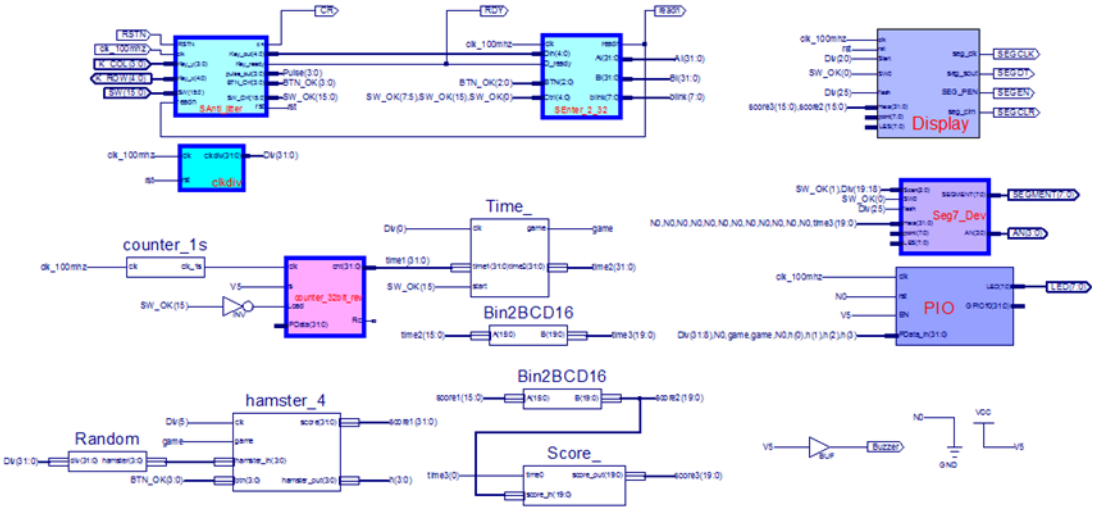

图 3 顶层模块图

2.3.2 Time_

这个模块的功能是把前一个模块的自增改为倒数。

```
`timescale 1ns / 1ps
module Time_(
    input clk,
    input [31:0]time1,
    input start,
    output game,
    output reg [31:0]time2
    );
    wire [31:0]time0;
    wire flag;
    assign time0=32'b00000000000000000000000000111100;//游戏时间为 60 秒
    assign flag=time0>time1?1:0;//flag 标记游戏时间是否到
    AND2 and2_0(.I0(start),.I1(flag),.O(game));//game 表示游戏是否结束
    always@(posedge clk)//如果游戏没结束，进行倒数
        if(game)time2<=time0-time1;else time2<={32{1'b0}};
endmodule
```

2.3.3 counter_32bit_rev

这个模块的功能是每经过 clk 时间，如果 s=1 则做自增，否则做自减，这是实验课上做的模块。

```
`timescale 1ns / 1ps
module counter_32bit_rev(
    input clk,
    input s,
    input Load,
    input[31:0]PData,
    output reg[31:0]cnt,
    output reg Rc
    );
    always@(posedge clk)begin//当时间是上升沿时做自增或自减
        if(Load)cnt<=PData;else begin
            if(s)cnt<=cnt+1;else cnt<=cnt-1;
            if((|cnt)|(&cnt))Rc<=1;else Rc<=0;
        end
    end
endmodule
```

2.3.4 counter_1s

这个模块的主要功能是每一秒输出一个脉冲。

```
`timescale 1ns / 1ps
module counter_1s(clk, clk_1s);
    input wire clk;
    output reg clk_1s;
    reg [31:0] cnt;
    always@(posedge clk)//用 cnt 记录 clk 过了多少次上升沿
        if(cnt<50_000_000)cnt<=cnt+1;else begin
            cnt<=0;
            clk_1s<=~clk_1s;//每过半秒更改 clk_1s 的值
        end
endmodule
```

2.3.5 Random

这个模块是通过用时间为种子来生成 4 个伪随机数来代表地鼠。

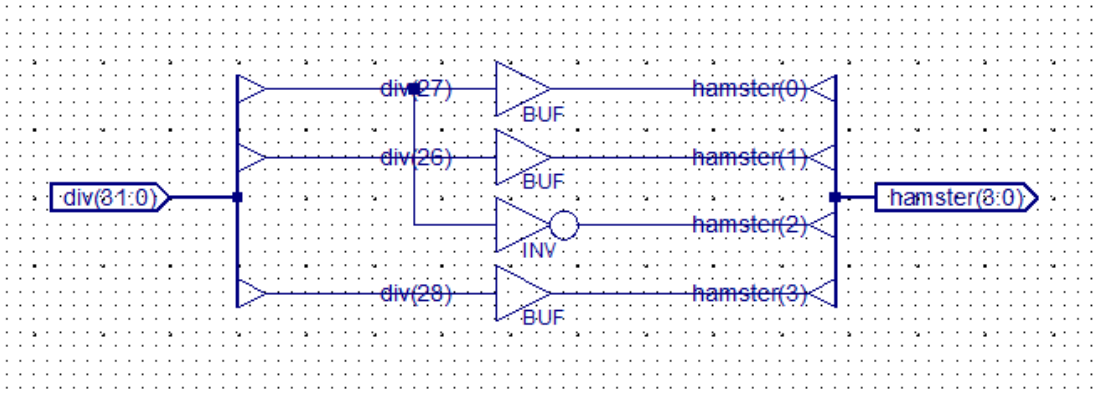

图 4 Random 模块的原理图

2.3.6 Press、Calculate 和 delay

Press 是当按键按下时判断是否打中地鼠，如果打中地鼠，地鼠灯将会灭掉。Calculate 是计算正确打到地鼠时加分，打不中地鼠时减分。Delay 是让 Calculate 先计算加减分，再改变地鼠的状态。

```
`timescale 1ns / 1ps
module Press(
    input hamster_in,
    input btn,
    output reg hamster_out
    );
    always@(posedge hamster_in or negedge hamster_in or posedge btn)
        if(btn)hamster_out<=1'b0;else//如果打中地鼠，地鼠灯将会灭掉
            if(hamster_in)hamster_out<=1'b1;else hamster_out<=1'b0;//如果地鼠重新出现，地鼠灯亮
```

```
endmodule
```

```
`timescale 1ns / 1ps
module Calculate(
    input btn,
    input hamster,
    input game,
    output reg[31:0]score
    );
    always@(posedge btn or negedge game)
        if(!game)score<=32'b00000000000000000000000000000100;else//如果游戏重置，每只地鼠的底分是 4 分
        if(hamster)score<=score+1'b1;else score<=score-2'b10;//如果正确打到地鼠时加 1 分，打不中地鼠时减 1 分
endmodule
```

```
`timescale 1ns / 1ps
module delay(
    input clk,
    input hamster_in,
    output reg hamster_out
    );
    always@(posedge clk)//每当 clk 上升沿时改变地鼠状态，起延迟作用
        hamster_out<=hamster_in;
endmodule
```

2.3.7 hamster_1 和 hamster_4

hamster_1 是输出每一只地鼠的状态以及在这只地鼠上得到的分数。hamster_4 是把 4 个 hamster_1 的地鼠状态综合起来，以及计算在各地鼠位置得分的总和。

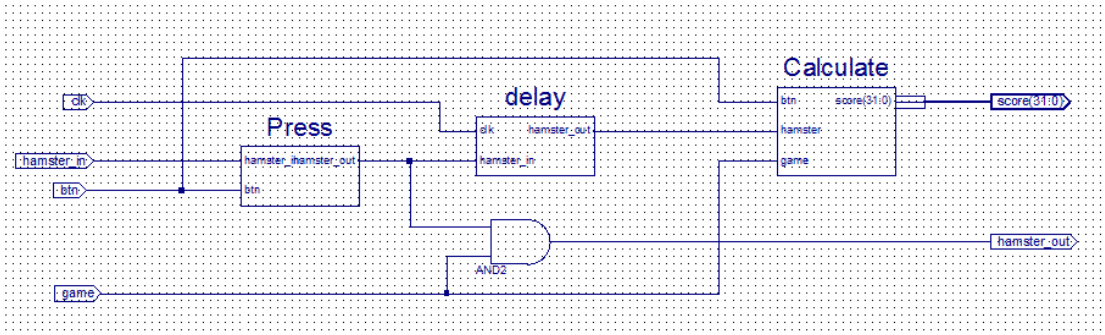

图 5 hamster_1 模块的原理图

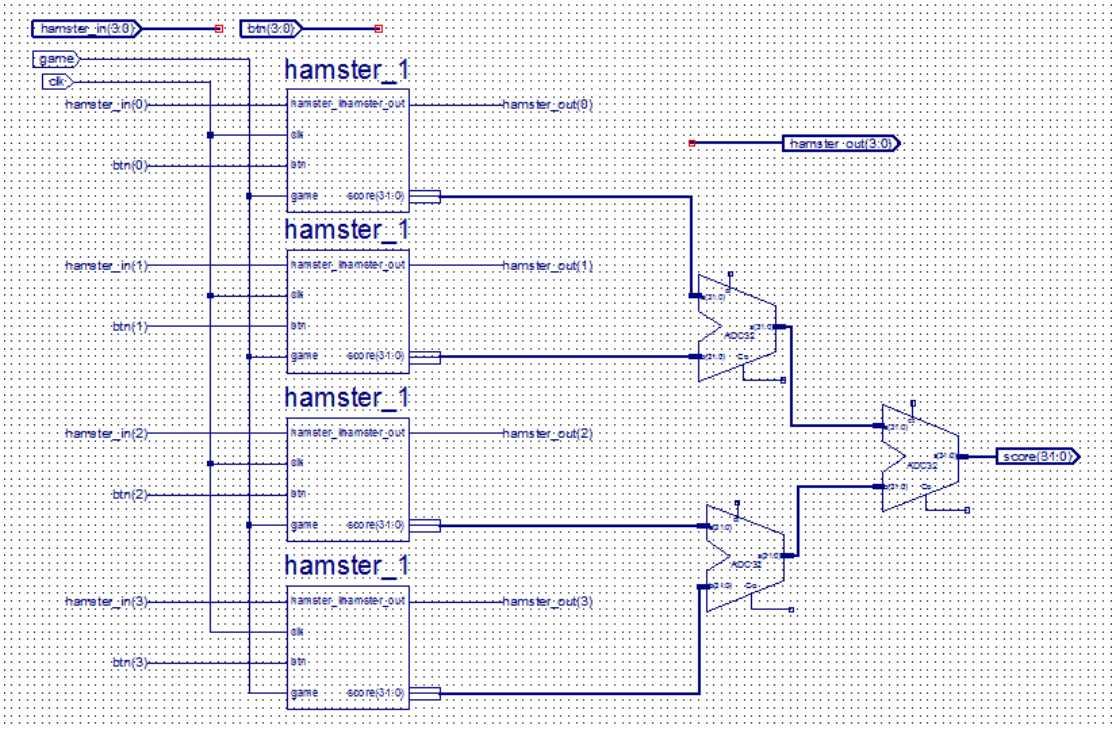

图 6 hamster_4 模块的原理图

2.3.8 Score_

Score_是记录上一次游戏所得到的分数。

```
`timescale 1ns / 1ps
module Score_(
    input [19:0]score_in,
    input time0,
    output reg [19:0]score_out
    );
    always@(posedge time0 or negedge time0)begin//每当时间改为零时记录分数
        if(score_in[0])score_out[0]<=1'b1;else score_out[0]<=1'b0;
        if(score_in[1])score_out[1]<=1'b1;else score_out[1]<=1'b0;
        if(score_in[2])score_out[2]<=1'b1;else score_out[2]<=1'b0;
        if(score_in[3])score_out[3]<=1'b1;else score_out[3]<=1'b0;
        if(score_in[4])score_out[4]<=1'b1;else score_out[4]<=1'b0;
        if(score_in[5])score_out[5]<=1'b1;else score_out[5]<=1'b0;
        if(score_in[6])score_out[6]<=1'b1;else score_out[6]<=1'b0;
        if(score_in[7])score_out[7]<=1'b1;else score_out[7]<=1'b0;
        if(score_in[8])score_out[8]<=1'b1;else score_out[8]<=1'b0;
        if(score_in[9])score_out[9]<=1'b1;else score_out[9]<=1'b0;
        if(score_in[10])score_out[10]<=1'b1;else score_out[10]<=1'b0;
        if(score_in[11])score_out[11]<=1'b1;else score_out[11]<=1'b0;
    end
endmodule
```

2.3.9 Bin2BCD_Add3 和 Bin2BCD16

Bin2BCD_Add 是把 4 位的二进制数转成 BCD 码，Bin2BCD16 是调用 Bin2BCD_Add3 把 16 位二进制数转成 BCD 码。

```
`timescale 1ns / 1ps
module Bin2BCD_Add3(in4, out4);
	input [3:0] in4;
	output reg [3:0] out4;
	always @(in4)
		case(in4)
			4'b0000:out4<=4'b0000;
			4'b0001:out4<=4'b0001;
		4'b0010:out4<=4'b0010;
		4'b0011:out4<=4'b0011;
		4'b0100:out4<=4'b0100;
		4'b0101:out4<=4'b1000;
		4'b0110:out4<=4'b1001;
		4'b0111:out4<=4'b1010;
		4'b1000:out4<=4'b1011;
		4'b1001:out4<=4'b1100;
		default:out4<=4'b0000;
		endcase
endmodule
```

```
`timescale 1ns / 1ps
module Bin2BCD16(A, B);
	input [15:0] A;
	output [19:0] B;
	wire [3:0] c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13;
	wire [3:0] c14, c15, c16, c17, c18, c19, c20, c21, c22, c23;
	wire [3:0] c24, c25, c26, c27, c28, c29, c30;
	wire [3:0] c31, c32, c33, c34;
	wire [3:0] d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13;
	wire [3:0] d14, d15, d16, d17, d18, d19, d20, d21, d22, d23;
	wire [3:0] d24, d25, d26, d27, d28, d29, d30;
	wire [3:0] d31, d32, d33, d34;
	assign d1 = {1'b0, A[15:13]};
	assign d2 = {c1[2:0], A[12]};
	assign d3 = {c2[2:0], A[11]};
	assign d4 = {c3[2:0], A[10]};
	assign d5 = {c4[2:0], A[9]};
	assign d6 = {c5[2:0], A[8]};
```

```
    assign d7 = {c6[2:0], A[7]};
    assign d8 = {c7[2:0], A[6]};
    assign d9 = {c8[2:0], A[5]};
    assign d10 = {c9[2:0], A[4]};
    assign d11 = {c10[2:0], A[3]};
    assign d12 = {c11[2:0], A[2]};
    assign d13 = {c12[2:0], A[1]};
    assign d14 = {1'b0, c1[3], c2[3], c3[3]};
    assign d15 = {c14[2:0], c4[3]};
    assign d16 = {c15[2:0], c5[3]};
    assign d17 = {c16[2:0], c6[3]};
    assign d18 = {c17[2:0], c7[3]};
    assign d19 = {c18[2:0], c8[3]};
    assign d20 = {c19[2:0], c9[3]};
    assign d21 = {c20[2:0], c10[3]};
    assign d22 = {c21[2:0], c11[3]};
    assign d23 = {c22[2:0], c12[3]};
    assign d24 = {1'b0, c14[3], c15[3], c16[3]};
    assign d25 = {c24[2:0], c17[3]};
    assign d26 = {c25[2:0], c18[3]};
    assign d27 = {c26[2:0], c19[3]};
    assign d28 = {c27[2:0], c20[3]};
    assign d29 = {c28[2:0], c21[3]};
    assign d30 = {c29[2:0], c22[3]};
    assign d31 = {1'b0, c24[3], c25[3], c26[3]};
    assign d32 = {c31[2:0], c27[3]};
    assign d33 = {c32[2:0], c28[3]};
    assign d34 = {c33[2:0], c29[3]};
    Bin2BCD_Add3 m1(d1, c1);
    Bin2BCD_Add3 m2(d2, c2);
    Bin2BCD_Add3 m3(d3, c3);
    Bin2BCD_Add3 m4(d4, c4);
    Bin2BCD_Add3 m5(d5, c5);
    Bin2BCD_Add3 m6(d6, c6);
    Bin2BCD_Add3 m7(d7, c7);
    Bin2BCD_Add3 m8(d8, c8);
    Bin2BCD_Add3 m9(d9, c9);
    Bin2BCD_Add3 m10(d10, c10);
    Bin2BCD_Add3 m11(d11, c11);
    Bin2BCD_Add3 m12(d12, c12);
    Bin2BCD_Add3 m13(d13, c13);
    Bin2BCD_Add3 m14(d14, c14);
    Bin2BCD_Add3 m15(d15, c15);
    Bin2BCD_Add3 m16(d16, c16);
```

```
    Bin2BCD_Add3 m17(d17, c17);
    Bin2BCD_Add3 m18(d18, c18);
    Bin2BCD_Add3 m19(d19, c19);
    Bin2BCD_Add3 m20(d20, c20);
    Bin2BCD_Add3 m21(d21, c21);
    Bin2BCD_Add3 m22(d22, c22);
    Bin2BCD_Add3 m23(d23, c23);
    Bin2BCD_Add3 m24(d24, c24);
    Bin2BCD_Add3 m25(d25, c25);
    Bin2BCD_Add3 m26(d26, c26);
    Bin2BCD_Add3 m27(d27, c27);
    Bin2BCD_Add3 m28(d28, c28);
    Bin2BCD_Add3 m29(d29, c29);
    Bin2BCD_Add3 m30(d30, c30);
    Bin2BCD_Add3 m31(d31, c31);
    Bin2BCD_Add3 m32(d32, c32);
    Bin2BCD_Add3 m33(d33, c33);
    Bin2BCD_Add3 m34(d34, c34);
    assign B = {c31[3], c32[3], c33[3], c34[3:0], c30[3:0], c23[3:0], c13[3:0], A[0]};
endmodule
```

第 3 章 打地鼠设计实现

3.1 实现方法

实验模块时不要都采用画原理图的方式设计模块，适合时要用代码的方式设计模块，因为有时写代码可能会比画原理图简单。写代码则需要对 verilog 的语法有深入的理解，以及能灵运用，这是一个难点所在的地方。

3.2 实现过程

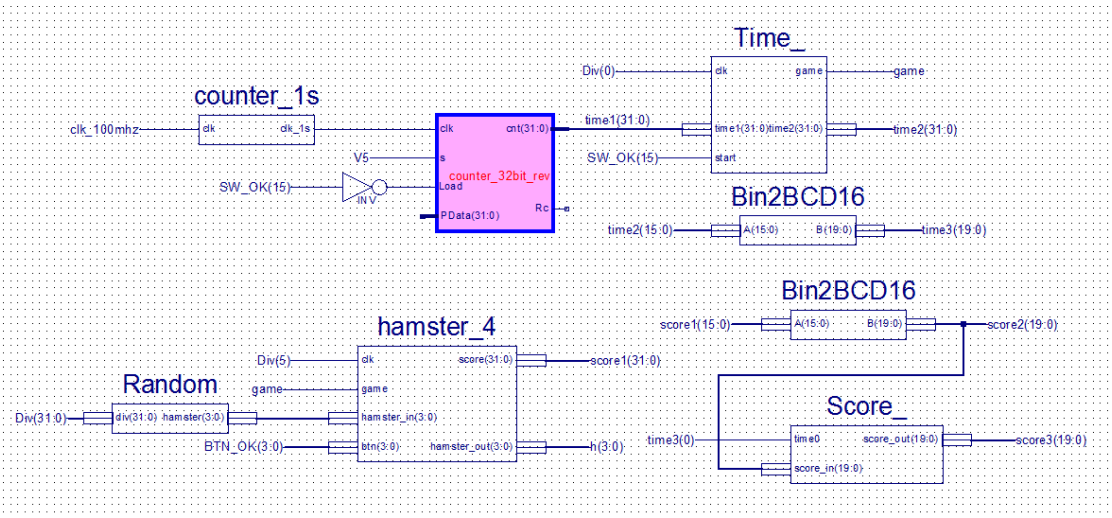

图 7 主要模块图

实现过程是先把每模块做好，再放到顶模块当中去。counter_1s 输出的一秒脉冲输入到 counter_32bit_rev， SW(15) 代表游戏开始输入到 counter_32bit_rev 的使能端，counter_32bit_rev 的自增输入到 Time_里改能倒计时，倒计时的输出是二进制数输入到 Bin2BCD16 中转成 BCD 码。随机数模块输出和按钮输入到 hamster_4 里计算得分和当前地鼠状态，hamster_4 的计分输入到 Bin2BCD16 中转成 BCD 码，得分输入到 Score 里存着。

3.3 仿真与调试

3.3.1 Time_

```
`timescale 1ns / 1ps
module Time__sim;
    reg clk;
    reg [31:0] time1;
    reg start;
    wire game;
    wire [31:0] time2;
    Time_ uut (
        .clk(clk),
        .time1(time1),
        .start(start),
        .game(game),
        .time2(time2)
    );
    initial begin
        clk<=0;
```

```
            start=0;
            #20;
            start=1;
            for(time1=0;time1<70;time1=time1+1)#10;//验证超过 30 秒时正否会停止游戏
            for(time1=0;time1<10;time1=time1+1)#10;
            start=0;//验证 SW=0 时正否会停止游戏
            for(time1=10;time1<20;time1=time1+1)#10;
        end
        integer i;
      always@*for(i=0;i<100;i=i+1)begin
            #1;
            clk<=~clk;
        end
endmodule
```

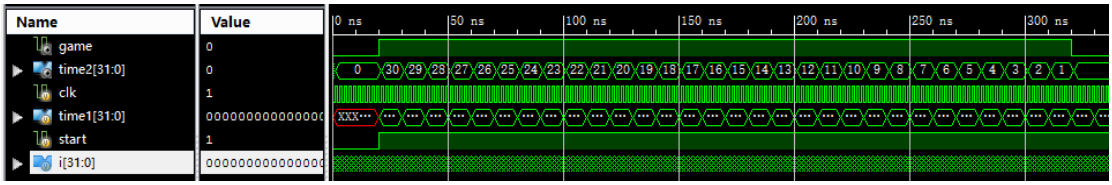

图 8 Time_模块的仿真图 1

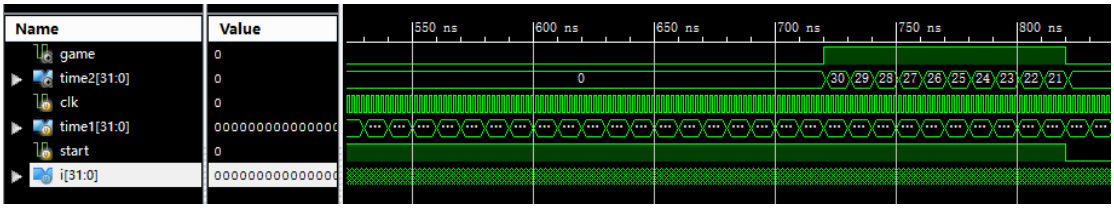

图 9 Time_模块的仿真图 2

3.3.2 Bin2BCD16

```
`timescale 1ns / 1ps
module Bin2BCD16_sim;
        reg [15:0] A;
        wire [19:0] B;
        Bin2BCD16 uut (
            .A(A),
            .B(B)
        );
        initial begin
            for(A=0;A<7'b1100100;A=A+1'b1)#50;//验证能不能把二进制转成 BCD 码
        end
endmodule
```

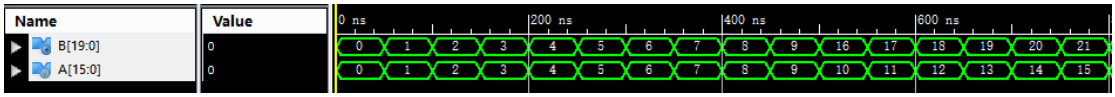

图 10 Bin2BCD16 模块的仿真图

3.3.3 Press

```
`timescale 1ns / 1ps
module Press_sim;
	reg hamster_in;
	reg btn;
	wire hamster_out;
	Press uut (
		.hamster_in(hamster_in),
		.btn(btn),
		.hamster_out(hamster_out)
	);
	initial begin//验证打中地鼠后地鼠是否会消失
		hamster_in=0;
		btn=0;
		#50;
		btn=1;
		#50;
		btn=0;
		#50;
		btn=1;
		#50;
		btn=0;
		#50;
		btn=1;
		#50;
		btn=0;
		#50;
		btn=1;
		#50;
	end
	integer i;
	always@*for(i=0;i<5;i=i+1)begin
		#100;
		hamster_in<=~hamster_in;
	end
endmodule
```

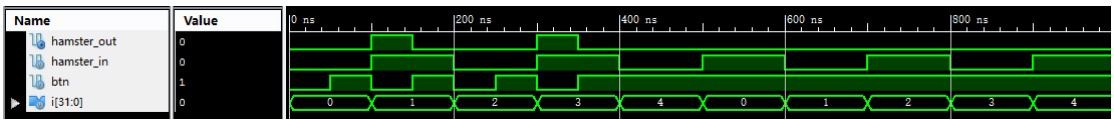

图 11 Press 模块的仿真图

3.3.4 Calculate

```
`timescale 1ns / 1ps
module Calculate_sim;
	reg btn;
	reg hamster;
	reg game;
	wire [31:0] score;
	Calculate uut (
		.btn(btn),
		.hamster(hamster),
		.game(game),
		.score(score)
	);
	initial begin//验证正确打中地鼠是否加分，打不中地鼠是否会减分
		hamster=0;
		game=0;
		btn=0;
		#25;
		btn=1;
		#25;
		btn=0;
		#25;
		btn=1;
		#25;
		game=1;
		btn=0;
		#20;
		btn=1;
		#20;
		btn=0;
		#20;
		btn=1;
	end
    integer i;
	always@*for(i=0;i<20;i=i+1)begin
		#50;
		hamster<=~hamster;
	end
endmodule
```

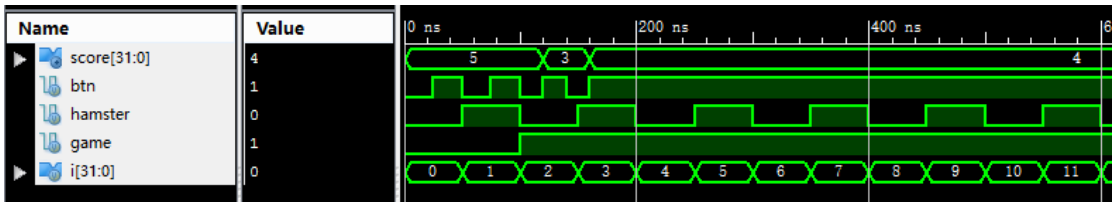

图 12 Calculate 模块的仿真图

3.3.5 hamster_1

```
`timescale 1ns / 1ps
module hamster_1_hamster_1_sch_tb();
    reg hamster_in;
    reg clk;
    reg btn;
    reg game;
    wire hamster_out;
    wire [31:0] score;
    hamster_1 UUT (
            .hamster_in(hamster_in),
            .clk(clk),
            .hamster_out(hamster_out),
            .score(score),
            .btn(btn),
            .game(game)
    );
     initial begin//验证 1 个地鼠时游戏是否正常
            clk=0;
            hamster_in=0;
            game=0;
            btn=0;
            #20;
            btn=1;
            #20;
            btn=0;
            #20;
            btn=1;
            #20;
            btn=0;
            #20;
            game=1;
            #20;
            btn=1;
            #20;
            btn=0;
```

```
            #20;
            btn=1;
            #20;
            btn=0;
            #40;
            btn=1;
            #20;
            btn=0;
            #20;
            btn=1;
            #20;
            btn=0;
        end
        integer i,j;
        always@*for(i=0;i<500;i=i+1)begin
            #1;
            clk<=~clk;
        end
        always@*for(j=0;j<10;j=j+1)begin
            #50;
            hamster_in<=~hamster_in;
        end
endmodule
```

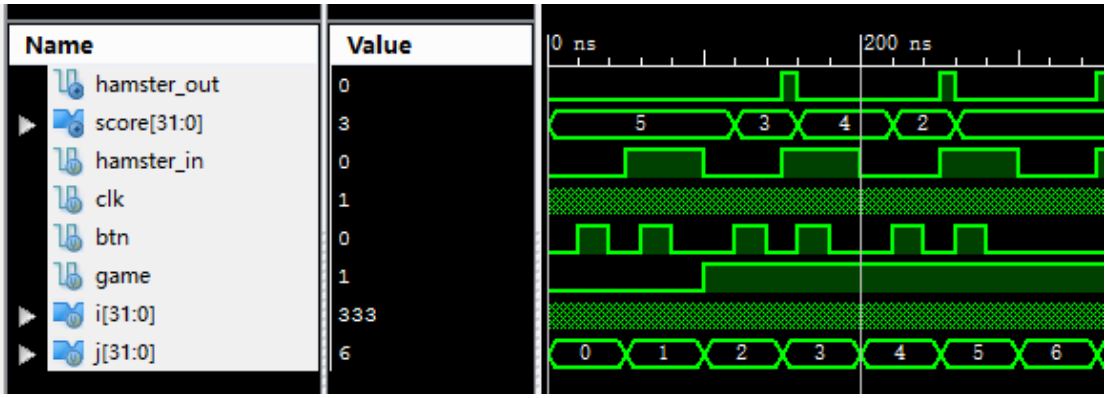

图 13 hamster_1 模块的仿真图

第 4 章 系统测试验证与结果分析

4.1 功能测试

上面的仿真代码的输入包括了输入变量的代表性情况，通过仿真图可见地鼠会随机地出现，然后过一段时间地鼠会消失，而且显示开始游戏的 2 个灯也能正常工作。所以模块的设计功能达到其目标。

4.2 技术参数测试

上面的仿真代码的输入包括了输入变量的代表性情况，通过仿真图可见当正确打到地鼠时会加 1 分，没打中地鼠会扣掉 1 分，而且计时模块也正常的倒计时。所以参数的设计功能达到其目标。

4.3 结果分析

存在的问题是原本我们设计时有一个功能是用来存储上一次游戏的分数，但是我们在实际验证的时候，却发现了错误，导致原本能存储上衣次游戏分数的功能无法实现，因为时间不够我没有找出最后 bug 在哪里，但我认为有可能是中间某些模块没有正常运作。

4.4 系统演示与操作说明

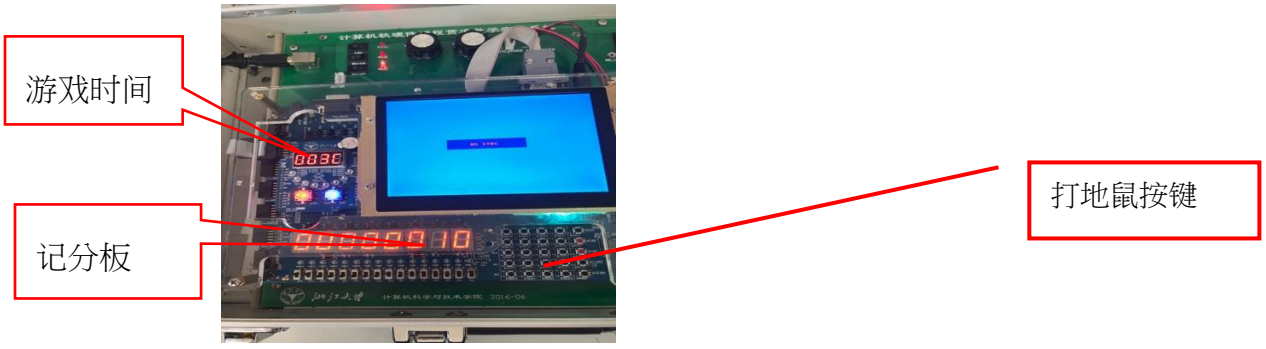

图 13 游戏面板介紹

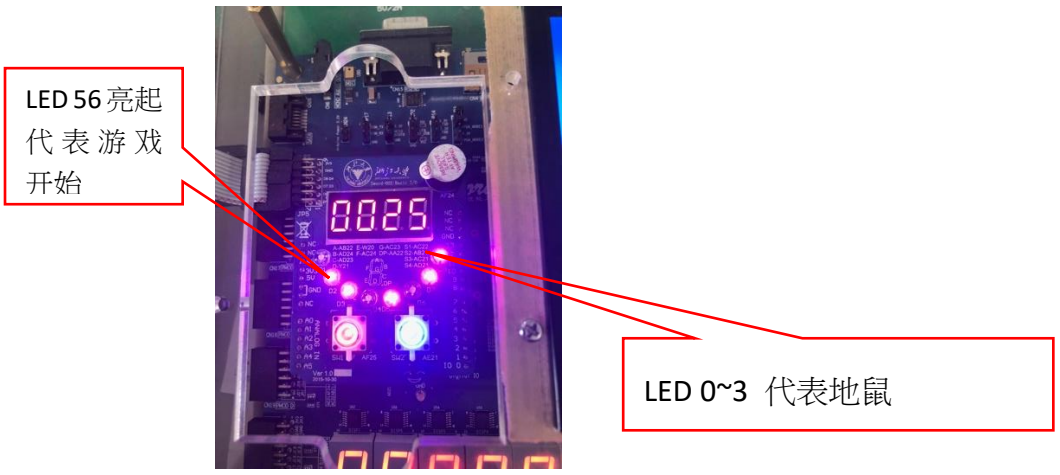

图 14 小板子的数码管和灯

小板子上的第 0~3 个灯代表地鼠，灯亮代表地鼠出现，灯灭代表地鼠消失。如果第 5 和 6 个灯亮，代表游戏开始，否则代表游戏结束。小板子上的 4 个 7 段数码管显示剩余时间（游戏时长为 60 秒）。

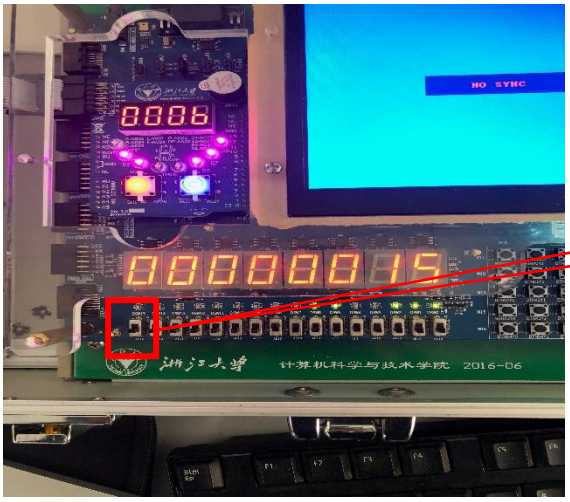

图 15 大板子上的数码管和游戏开关

大板子上的左边 4 个 7 段数码管显示上一局游戏的得分，大板子上的右边 4 个 7 段数码管显示当前游戏的得分。当开关 SW 变为 1 时，游戏重置；当开关 SW 变为 0 时，开始游戏。

图 16 对应地鼠的 4 排按钮

BTN0~BTN3 分别对应着小板子上的第 3 到 0 灯的地鼠，当成功打死一只地鼠时会得 1 分，打错失误扣 1 分，游戏开始底分为 16 分。

第 5 章 结论

这次的大程我主要做的事一个打地鼠的游戏，每次游戏 60 秒，保底分为 16 分，地鼠的出现用时间为种子来生成 4 个伪随机数来代表地鼠。地鼠的出现用 LED 灯来显示，地鼠出现则点亮，当打中一支地鼠则加一分并且 LED 灯熄灭，失误打错则扣去一分。这次大程我还是遇到了一些困难，像是我原本想要把这个打地鼠的游戏用 VGA 屏幕来显示，但是在实现过程中对于屏幕的使用出现许多 Bug，像是地鼠画面出现混乱，影像重叠等等，所以后来因为

时间原因决定以 LED 以及七段数码管来实现。但是以 LED 灯以及七段数码管的实现过程中也遇到一些问题，像是地鼠的随机出现有时候会意外当机，还有打中地鼠时 LED 灯有时会出现无预警闪烁，这些问题在过程中的 Debug 都相当困难。还有一个较为大的问题，在整个程序的设计过程中，原本有一个功能是要能在计分板的左侧纪录少一次游戏的分数，也就是寄存器的功能，代码里面也有这个寄存器模块，但是在实验室下版子验证实却一直出现问题，有时后能成功寄存，有时后出现错误。对于整个游戏我还是相当满意，打地鼠的游戏还是可以顺利执行，再规划，评估，代码，除错，不断的除错，这些过程中我也学到了许多，如何自行设计，写代码的能力也有所提升。