

（深圳）

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： T2615

学生班级： 7班

学生学号： 190110716

学生姓名： 朱海峰

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 |
| 概述基本功能  生成五个五位八进制数，按时间顺序显示在数码管上。玩家记住第某个数后，在下一步输入地址（即序数）后，依次输入该五位八进制数，如果对应地址的数匹配，则提示匹配成功，否则继续匹配，知道最新输入的五个数匹配。 |
| 系统功能详细设计 |
| 用硬件框图描述系统主要功能及各模块之间的相互关系  包括状态描述、状态转换图及状态编码    图 1  状态描述：   * 状态1：初始状态，数码管显示8个0； * 状态2：数码管按照1Hz频率显示五个五位八进制数； * 状态3：玩家输入选择的地址（即上一步某个五位数中的一个的序数）； * 状态4：玩家输入一个3位二进制数，按下S3确认，在输入下一个数。   状态间的转移通过按键信号控制。    图 2 状态转换图  状态编码：  localparam[2:0]  state0 = 3'b000,  state1 = 3'b001,  state2 = 3'b010,  state3 = 3'b011;  状态编码使用3位，为远期功能预留空间，当前实验并未用到最后一位。 |

|  |
| --- |
| 模块描述 |
| 包括模块功能，输入、输出端口、变量含义及主要设计 topDesign模块： 输入：5个按键信号、时钟信号、8位开关信号  输出：左右数码管信号、数码管使能信号。   1. 模块中通过按键信号控制状态转移：  |  |  |  | | --- | --- | --- | | 现态 | 按键 | 次态 | | × | S4 |  | | 0 | S0 | 1 | | 1 | S1 | 2 | | 2 | S2 | 3 |   if(S4)  next\_state <= state0;//初始  else if(cs==state0 && S0)  next\_state <= state1;//显示  else if(cs==state1 && S1)  next\_state <= state2;//输入地址  else if(cs==state2 && S2) begin  next\_state <= state3;//读数状态  addrSel <= sw\_i[2:0];  end   1. 不同状态下，对数码管输入不同状态对应寄存器的数据，其中data1、data2、data3……data8直接输入数码管模块：   if(cs==state0)begin //状态0  data1 = data1\_0;  data2 = data2\_0;  ...  data8 = data8\_0;  end  else if(cs==state1)begin //状态1  data1 = data1\_1;  ...  end  else if(cs==state2)begin //状态2  ...   1. 扫描更新伪随机数序列：   always@(posedge clk\_i)begin  i <=(i+1) % 'd25;  if(i>='d0 && i<'d5)  temp1[i%'d5] <= Y;  else if(i>='d5 && i<'d10)  temp2[i%'d5] <= Y;  else if(i>='d10 && i<'d15)  temp3[i%'d5] <= Y;  else if(i>='d15 && i<'d20)  temp4[i%'d5] <= Y;  else if(i>='d20 && i<'d25)  temp5[i%'d5] <= Y;  end  其中Y是当前时刻随机数模块生成的4位数（实际上只有0~7），Y每个时钟沿都在更新，temp1、temp2、temp3、temp4和temp5为暂存的当前时刻随机数序列。   1. 状态为0时，按下S0，当前随机数序列temp1、temp2等存入寄存器randString1、randString2等：   always@(posedge S0)begin  if(cs ==state0)begin  randString1[0] = temp1[0];  ...  randString2[0] = temp2[0];  ...  randString3[0] = temp3[0];  ...  end  end   1. 状态1，显示5个5位八进制伪随机数：   if(cs == state1)begin  if(cnt == 'd0)begin  data1\_1 <= randString1[0];  ...  end  else if(cnt == 'd1)begin  data1\_1 <= randString2[0];  ...  end  else if(cnt == 'd2)begin  ...  end  data6\_1 <= 'he;//不显示  data7\_1 <= 'he;  data8\_1 <= cnt;  if(cnt<'b100)  cnt <= cnt + 1;//每秒钟计数增加  end  else  cnt <= 0;   1. 状态2，输入地址信号，如果不符合要求输出8个f：   if(cs==state2)begin //状态2  if(sw\_i>'d4)begin  data1 = 'hf;  data2 = 'hf;  ...  else begin  data1 = sw\_i[3:0];  data2 = ...  end;  end   1. 状态三，输入数据，数码管显示左移一位：   if(cs==state3 && S3) begin  data1\_3 = data2\_3;  data2\_3 = data3\_3;  data3\_3 = data4\_3;  data4\_3 = data5\_3;  data5\_3 = sw\_i[2:0];  end hexseg8模块： 输入：8个4位数、时钟信号  输出：左、右侧数码管信号、使能信号  功能：将八个四位数转化为7位的数码管信号，原时钟通过分频得到1000Hz扫描使能型号。  具体实现：实例化8个单个数码管信号转换模块，实例化一个分频模块。通过计数，利用1000Hz信号，输出扫描使能。 singleDisplay模块： 输入：4位数据  输出：7位数码管信号  case(data)  4'b0000: led = 7'b1111110; //0  ...  4'b1101: led = 7'b0000001; //d 用来表示-  4'b1110: led = 7'b0000000; //e 用来表示不亮  default: led = 7'b1000111; //F  endcase  由于本实验中不需要显示9~E，故将E用来表示不亮，而d用来表示“-”（dash）。 shinningO模块： 输入：系统时钟信号  输出：闪烁的0。  功能：通过分频得到4Hz时钟信号，输出4位交替的0与E。 divider1Hz模块： 输入：系统时钟  输出：1Hz时钟信号  功能：实现分频，以1Hz的模块为例。  if (rst == 1'b1)  cnt <= 51'b0;  else  cnt <= (cnt >= 'd100000000) ? 'h0 : cnt + 'h1;  clk\_o <= rst\_n\_i & (cnt < 'd50000000); randNum模块： 输入：系统时钟  输出：4位伪随机数（其中最高位为0）  功能：基于LFSR产生一个随机数，由于本实验需要用到的0~7，故输出模8即可。  begin  rand\_num[0] <= rand\_num[7];  rand\_num[1] <= rand\_num[0];  rand\_num[2] <= rand\_num[1];  rand\_num[3] <= rand\_num[2];  rand\_num[4] <= rand\_num[3]^rand\_num[7];  rand\_num[5] <= rand\_num[4]^rand\_num[7];  rand\_num[6] <= rand\_num[5]^rand\_num[7];  rand\_num[7] <= rand\_num[6];  end |
| 管脚分配表 |
| #时钟信号  P17 clk\_i  #输入信号  R1 sw\_i[0]  N4 sw\_i[1]  M4 sw\_i[2]  R2 sw\_i[3]  P2 sw\_i[4]  P3 sw\_i[5]  P4 sw\_i[6]  P5 sw\_i[7]  #数码管使能  G2 en\_seg[0]  C2 en\_seg[1]  C1 en\_seg[2]  H1 en\_seg[3]  G1 en\_seg[4]  F1 en\_seg[5]  E1 en\_seg[6]  G6 en\_seg[7]  #前四位数码管信号  B4 led\_o\_l[6]  A4 led\_o\_l[5]  A3 led\_o\_l[4]  B1 led\_o\_l[3]  A1 led\_o\_l[2]  B3 led\_o\_l[1]  B2 led\_o\_l[0]  #后四位数码管信号  D4 led\_o\_r[6]  E3 led\_o\_r[5]  D3 led\_o\_r[4]  F4 led\_o\_r[3]  F3 led\_o\_r[2]  E2 led\_o\_r[1]  D2 led\_o\_r[0]  #按键  U4 S4  V1 S3  R15 S2  R17 S1  R11 S0 |

|  |
| --- |
| 调试报告 |
| 包括仿真代码、仿真波形截图及仿真分析（需包含状态转换过程）  由于仿真过程较慢，要仿出现实中的以秒为单位的变化很费时间。于是在仿真的时候，将每个时钟分频器输出的频率增大。  对系统的伪随机数生成功能进行仿真，要求在波形中观察生成随机数时的关键信号和所生成的伪随机数   1. 单个数码管显示仿真，其中E用于表示不亮，D用于表示dash连接线。通过对数据递增，完成0、1、…、D、E每个数的数码管显示仿真。   仿真代码：  reg [3:0]data\_i = 0;  wire [6:0]led\_o;  singleDisplay uut\_singleDisplay\_1(  .data (data\_i),  .led (led\_o)  );  always #1 data\_i <= data\_i +1;    图 3   1. 八位数码管显示模块仿真   将需要显示的数据接入数码管模块，查看输出型号，主要是观察能信号。使能信号呈现出四个一组循环接替有效，说明仿真成功。  仿真代码：  reg clk=0;  reg [3:0]  data1, data2, data3, data4, data5, data6, data7, data8;  reg [7:0] sw\_i = 'b00000000;  wire [6:0] led\_o\_l;  wire [6:0] led\_o\_r;  wire [7:0] en;  wire dot;  hexseg8 uut\_hexseg3\_1(  .clk\_i (clk),  .data1 (data1),  .data2 (data2),  .data3 (data3),  .data4 (data4),  .data5 (data5),  .data6 (data6),  .data7 (data7),  .data8 (data8),  .led\_o\_l (led\_o\_l),  .led\_o\_r (led\_o\_r),  .en (en)  );  always #1 clk=~clk;  always #10000000 sw\_i = sw\_i + 1;  initial begin  data1<='b1;  data2<='d2;  data3<='d3;  data4<= 'd4;  data5<= 'd5;  data6 <= 'd6;  data7 <= 'd7;  data8 <= 'd8;  end    图 4   1. 伪随机数生成   伪随机数模块在每个时钟沿实时更新一个新0~7的随机数：out。    图 5   1. 状态转移控制   每次延时后给一个按钮按下有效信号，仿真状态表的逻辑。以下仿真代码得到信息较多，之后的波形分析都是以下仿真代码得出：  reg clk\_i = 0;  reg S1 = 0,  S2 = 0,  S3 = 0,  S4 = 0,  S0 = 0;  reg [7:0]sw\_i;  wire [6:0]led\_o\_l,led\_o\_r;  wire [7:0] en\_seg;  topDesign U\_topDesign\_sim1(  .clk\_i (clk\_i),  .S4 (S4),  .S3 (S3),  .S2 (S2),  .S1 (S1),  .S0 (S0),  .sw\_i (sw\_i),  .led\_o\_l(led\_o\_l),  .led\_o\_r(led\_o\_r),  .en\_seg (en\_seg)  );  always #1 clk\_i = ~clk\_i;  initial begin  #100 S0 = 1;  #10 S0 = 0;  #1000 S1 = 1;  #10 S1 = 0;  sw\_i[7:0] = 8'b00000011;  #100 S2 = 1;  #10 S2 = 0;  #10 sw\_i[7:0] = 8'b00000101;  #1 S3 = 1;#1 S3=0;  #10 sw\_i[7:0] = 8'b00000010;  #1 S3 = 1;#1 S3=0;  #10 sw\_i[7:0] = 8'b00000101;  #1 S3 = 1;#1 S3=0;  #1 sw\_i[7:0] = 8'b00000011;  #1 S3 = 1;#1 S3=0;  #1 sw\_i[7:0] = 8'b00000111;  #1 S3 = 1;#1 S3=0;  #1 sw\_i[7:0] = 8'b00000111;  #1 S3 = 1;#1 S3=0;  end    图 6   1. 通过计数i控制的实时更新伪随机数temp     图 7   1. randString为S0上升沿存入寄存器的随机数   S0按下时，五组五个0~7的随机数存入寄存器里面。    图 8   1. 成功匹配，data8输出从闪烁的0变为一个a   在输入五个正确的数后，数码管显示指示正确输入。    图 9   1. 成功匹配后又输入一个数，变成失败匹配状态，data8回到0与E交替，即闪烁的0的状态。     图 10 |

|  |
| --- |
| 设计过程中遇到的问题及解决方法 |
| 仿真中遇到的问题（需包括仿真截图）  在闪烁的0模块制作中，发现0并没有闪烁，在仿真中E并没有出现，发现分频器计数位数不足，输出时钟并没有出现高低电平交替。  将计数位数增加后，得到成功分频后的4Hz时钟输出：    图 11  输入正确序列后，指示灯仍然闪烁，不提示正确。    图 12  通过单步调试，缩小问题范围，发现是序列顺序搞反了，Verilog默认的顺序是从高位到地位，与输入的顺序相反。    图 13 |

|  |
| --- |
| 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  整个数字逻辑实验，我遇到了很多困难，和许多具体的问题。问题没有一一解决，但是每个实验都成功按时做出来了，很多东西是书本上得不到的。比如，某个变量在多个always中赋值，仿真没问题，上板全是问题，这个是我以前不知道的。以前不知道，就只能一组敏感信号写一个always块，但是对某个变量的赋值却非常混乱；再知道这个特性之后，编程思路一下子就清晰了：每个（组）变量的变化用一个always块，就不会出错。  每个实验的难度逐步上升，梯度十分合适，到了实验六，一个综合性的实验，怎么说呢，做出来了之后感觉也不算难。每次的困难都在于不清楚自己哪里的逻辑出问题了，或者是自己的逻辑与verilog的逻辑对不上，实现的方式不对。Verilog语言没有专门去学，老实说，基础很差，如果需要进一步用的话，最好还是从头学。  写Verilog，相信大多数同学都是从C语言这种面向过程的思维来写的，但是硬件远远不止一条线拉下来地这样运行。  实验逻辑简单，实现也不算难，调试是最繁琐的。从调试仿真到调试上板，方法区别也非常大。调试仿真，除了看波形，还能利用单步运行、断点调试，这个是我在运用过程中自己摸索出来的，仿真的调试跟C语言的调试很像，通过观测各个变量的变化，缩小问题所在的可能范围，然后改。上板的话，就复杂很多，可以看message里面的warnings，特别是critical warning，能解决一些问题。其实更多情况是这些warnings根本解决不了问题，只有凭着自己的直觉，去改一些always块的敏感信号，可能会有一些变化。  讲真，耐心最重要，老是调不出来，还得敢于放弃，看看别人怎么写的，问问老师，改一改不规范的代码。毕竟重写，是走入绝境后还想完成实验最大的可能。 |