浙江大学

课程设计报告

题目：**基于 FPGA 的炸弹人游戏**

The Bomb Man Game Based on FPGA

指导教师：刘海风

姓名学号：郝家辉 3160101827

参加成员：罗　昊 3160101822

马　存 3160101779

专　　业：计算机科学与技术

学　　院：计算机科学与技术学院

提交日期 **2017** 年 **1** 月 **16** 日

目录

摘要

炸弹人（Bomb man）是 HUDSON 公司在 1983 年推出的一款游戏。这款游戏十分经典以至于经过几十年的更新换代，它被用多种语言实现并登陆许多不同平台，游戏的机制与趣味性也更加丰富。炸弹人游戏成为了我们许多人童年的美好回忆，炸弹人小游戏利用人们的破碎时间，让人们从紧张的日常生活中得到一刻放松，让人保持良好的心态，继续投入生活工作中去。

本组自主设计的炸弹人游戏保留了游戏核心的元素，即玩家可以控制炸弹人灵活地移动躲避小鬼、小鬼能识别机器人的移动方位并追逐炸弹人、炸弹人可以在指定位置放下炸弹并消灭小鬼取得胜利，或者炸弹人被小鬼追到而游戏结束。

本课程设计由小组三名成员利用已有的 SWORD 开发板，采用 FPGA 技术和 Xilinx ISE 14.7 平台，并结合所学的知识和借鉴相关资料独立完成。

关键词：炸弹人；FPGA 平台；游戏开发

基于 FPGA 的炸弹人游戏

1. 背景介绍

1.1. 设计背景

现如今各式各样的大型游戏层出不穷，但是当我们被色彩艳丽、高度逼真的大型 3D 游戏吸引，惊叹于计算机技术和时代发展之快的同时，也不能忘记掉几十年前的游戏开发工作者的艰辛。在上世纪 80、90 年代，市面上流行的游戏大多画面粗糙、操作简单，在现在看来或许根本不值一提。在我们现在看来编写一个类似的小游戏并不是一件困难的事情，因为现在我们可以使用已有的图形库，用 java、C++等语言实现游戏的逻辑，并用 C#和一些脚本语言实现美观的界面与人性化的用户交互。但在当时，这些高级语言并没有被广泛应用到游戏编写中，这些游戏仍主要用 BASIC 和汇编等语言编写，过程较为繁琐。

除去用过去的 Basic、汇编语言和现在广泛使用的高级语言外，我们还可以用硬件描述语言开发简单的小游戏。Verilog HDL 是一种硬件描述语言，它可以用文本的形式描述数字系统硬件的结构和行为。用它可以表示逻辑电路图、逻辑表达式和数字逻辑系统所完成的逻辑功能。这种语言的优势在于它并不是对应某个硬件，而是忽略了硬件内部具体的实现方法而对系统的行为进行描述。因此，使用 Verilog HDL 和 FPGA 技术开发小游戏对我们来说是一个全新的尝试，因为我们不仅需要实现游戏的内在逻辑，而且还要同时设计一系列的控制、显示和音乐等模块。

1.2. 相关行业背景

FPGA（Field－Programmable Gate Array），即现场可编程门阵列，是在 PAL、GAL、CPLD 等可编程器件的基础上进一步发展的产物，它在上世纪 80 年代开始被应用于 PLD 中。它是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。以硬件描述语言（Verilog 或 VHDL）所完成的电路设计，可以经过简单的综合与布局，快速地烧录至 FPGA 上进行测试，是现代 IC 设计验证的技术主流。这些可编辑元件可以被用来实现一些基本的逻辑门电路（比如 AND、OR、XOR、NOT）或者更复杂一些的组合功能比如解码器或数学方程式。在大多数的 FPGA 里面，这些可编辑的元件里也包含记忆元件例如触发器（Flip－flop）或者其他更加完整的记忆块。

在过去，大部分计算机的操作都是通过键盘鼠标实现的，这种操作方式比较机械死板。而随着虚拟现实技术和传感器技术的发展，人机交互也越来越成为计算机升级换代的趋势。这种人机互动模式以更为自然性、直观性的运动、语音等摸式代替电脑的键盘和鼠标等外设。而 FPGA 往往在这类交互操作中发挥着重要的作用。目前有不少体感游戏、传感器操控器、数据手套等创新成果都是离不开 FPGA 的。因此采用 FPGA 作为主控器实现一些游戏、应用等，已经成为了当今计算机和电子领域的发展趋势之一。

1.3. 主要内容和难点

基于 FPGA 用 Verilog HDL 设计炸弹人这个游戏，我们需要实现的内容主要有以下三个：

1) 把游戏过程中可能出现的情况划分为不同的状态，实现状态机。在这部分需要清晰地描述状态如何转移、状态转移的约束条件以及每个状态的输出是什么。
2) 实现游戏内部逻辑功能。对于这个炸弹人游戏，重要的是规定炸弹人和小鬼可以走哪些路、炸弹被放置后多久会爆炸以及爆炸后哪些方向会出现火焰、小鬼追炸弹人的算法、炸弹人被追到和小鬼被炸到的输赢判定。这些逻辑功能对于游戏来说是极其重要的，直接关系到玩家的体验。
3) 控制模块和输出模块的设计。此游戏要求控制模块支持的操作尽量简单，而且更重要的是使炸弹人能够灵敏、准确地移动和放置炸弹。输出模块包括 VGA 显示输出模块和音频输出模块。我们自行添加的背景音乐可以为玩家带来愉快的游戏体验。

FPGA 的开发与传统的 PC、单片机的开发有很大不同。FPGA 以并行运算为主，以硬件描述语言来实现，与 PC 的顺序操作有较大的不同，因此对于我们这样的入门者而言，用 Verilog 语言设计时序电路来开发游戏，这对我们来说较为困难。除此之外，我们还需要自学 VGA 的扫描方式和设计方式，使图像显示在屏幕上。

2. 炸弹人游戏设计方案

2.1. 游戏设计方案

根据炸弹人游戏的游戏规则，整个游戏需要设计显示、控制以及其他模块：如音乐等模块，其中最为重要和复杂的是游戏状态的切换、炸弹人和小鬼的移动以及炸弹的释放、爆炸及消失等游戏功能。

图 1 显示了炸弹人游戏的游戏逻辑

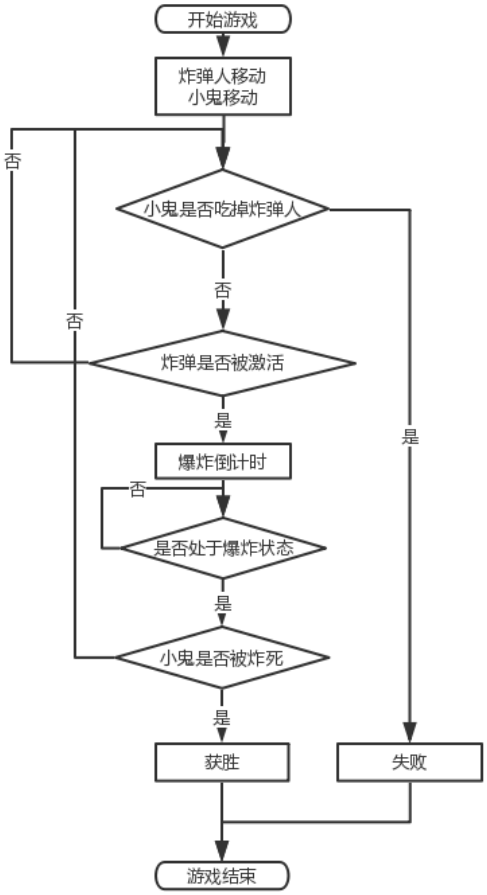

图 1 游戏逻辑流程图

通过对以上流程图的分析，分别将各个过程落实到模块上，实现各模块来完成游戏的设计。

2.2. 模块设计方案

利用自顶向下的设计方法，逐步缩小设计模块，分离各模块功能。在大型数字系统设计中，顶层模块尽量不要涉及逻辑关系，最好只进行模块调用，因此 m10，m11，m12 等较为简单的逻辑功能也单独设计了模块。

2.2.1. Top 模块

输入：

clk_100m板载 100 兆赫时钟
Key[4:0]　　5 个开关状态
Rst　　复位开关状态

输出：

VGA_HS　　VGA 水平同步信号

VGA_VS　　VGA 垂直同步信号
VAG_R[3:0] VGA 色彩 R
VAG_G[3:0] VGA 色彩 R
VAG_B[3:0] VGA 色彩 R
Speaker　　音乐
AUD_SD　　AUD 输出
Buzzer　　蜂鸣器

功能：

2.2.1.1. 调用其他模块

设置相关的中间变量进行数据传输，调用各个模块。

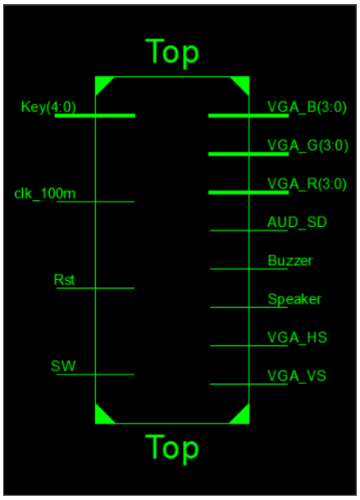

图 2 top 模块图

2.2.2. m1-controller 模块

输入：
clk_25m　　25 兆赫时钟
rst　　复位信号
vga_vs　　VGA 垂直同步信号
key[5:0]　　6 个开关状态

输出：
bg_select[1:0]　　背景选择信号
ob_ram_address[2:0]　物体写 RAM 选择信号
ob_ram_data[12:0]　物体写 RAM 数据
ob_ram_writeenable　物体写 RAM 写入使能

功能及实现：

2.2.2.1. 将开关输入转化为炸弹人控制信号

设置寄存器 last_sw[4:0]储存上一次的开关状态，根据前后两个时钟周期内的开关状态、长按计数器 Long_press_count[5:0]来判断开关是否处于长按状态，从而输出真正的开关值 Key_value[4:0]。

2.2.2.2. 炸弹人移动控制

为了减少繁复的随机数判定，炸弹人出生位置设置为固定值，在每次复位后出生。每个移动周期读取 Key_value[3:0]来判定炸弹人的移动方向，分别用 4’b1000，4’b0100，4’b0010，4’b0001 来代表上下左右四个方向。

在读取到方向信息后，判断该方向的下一格是否为墙壁（障碍物），在非障碍物的情况下，对炸弹人的坐标进行对应方向的修改从而实现炸弹人的移动控制。

2.2.2.3. 小鬼自动追踪炸弹人

同样的，为了减少繁复的随机数判定，小鬼出生位置设定为固定值，在每次复位后重生。为了控制小鬼的移动速度，单独为小鬼设置了运动时钟 Sprite_clk[4:0]，通过对模块时钟的分频获得。

为了决定小鬼最终的移动方向，设置临时方向寄存器 Direct_temp[1:0]，为了判定小鬼与炸弹人的相对位置关系，设置了位置差距变量 X_diff[4:0]与 Y_diff[3:0]。在小鬼移动周期内计算小鬼与炸弹人的二维位置差，通过差值的正负确定运动方向为八个可能的至多双向的方向组成（即：左上、左下、右上、右下、上、下、左、右）中的某一个。

在确定方向组成后，如果仅为单向运动，直接将临时方向寄存器的指赋给最终方向寄存器 Direct_sprite[1:0]，如果为双向运动，那么就按照预设的 XY 方向优先级在双向中选取一个优先级较高的方向转为单向运动。

2.2.2.4. 炸弹人与小鬼的死亡判定即游戏胜负判定

炸弹人的死亡判定为被小鬼吃掉即二者坐标完全相同，此时游戏失败。

小鬼的死亡判定为处于爆炸状态的炸弹中心周围 9 格范围内，此时游戏失败。

2.2.2.5. 炸弹的状态判断与切换

为炸弹设置状态寄存器 Bomb1_state[1:0]，2’d0，2’d1，2’d2 分别代表未释放、释放、爆炸，炸弹的各个状态切换需要对应的时钟进行控制，设置了 Bomb1_clk[7:0]作为对应时钟，通过对模块时钟进行分频获得。炸弹的状态判断与切换为一个时序状态机。

炸弹状态在复位后设置为“未释放”状态；在炸弹释放键值（Key_value[4]）变为 1 后，炸弹进入“释放”状态，与此同时炸弹时钟开始计数；在计数达到 8’d200 时（约 2s 时间），炸弹进入“爆炸”状态，同时炸弹时钟复位并重新开始计时，“爆炸”状态将持续至炸弹时钟达到 8’d50（约 0.5s），此时炸弹恢复至“未释放”状态，炸弹时钟复位。

根据游戏需要，我们应使炸弹在“未释放”状态不显示任何物体，在“释放”状态显

示炸弹，在“爆炸”状态显示一个十字火花（火花会因墙壁限制显示为┼、┴等不同的十字衍生形状）。

2.2.2.6. 墙壁的判定

根据墙壁块所在坐标的二进制数末位特点判断当前坐标是否为墙壁。

2.2.2.7. 物体写 RAM 输出

根据炸弹状态、游戏胜负状态进行判断，利用一个时序状态机输出物体写 RAM 操作相关数据。写 RAM 数据需要三个寄存器：ob_ram_address[2:0]，ob_ram_data[12:0]，ob_ram_writeenable 分别储存物体地址编号（不同的物体代表的编号不同）、物体写 RAM 数据（12 位控制是否显示物体，[11:9]位储存物体显示块编号，[8:4]储存物体 X 坐标，[3:0]储存物体 Y 坐标）以及物体写 RAM 使能信号。

状态 0 为空状态，相关数据输出均为 0，在进入写周期后，进入状态 1；

状态 1 根据游戏是否结束决定炸弹人写 RAM 数据，进入状态 2；

状态 2 根据游戏是否结束决定小鬼写 RAM 数据，进入状态 3；

状态 3 根据炸弹状态是否为“释放”决定炸弹写 RAM 数据，处于“释放”则进入状态 0，否则进入状态 4；

状态 4 根据炸弹状态是否为“爆炸”决定火花中心写 RAM 数据，处于“爆炸”状态则进入状态 5，否则进入状态 0；

状态 5 根据炸弹状态是否为“爆炸”以及左侧一格是否为墙壁决定火花左边缘写 RAM 数据，进入状态 6；

状态 6 根据炸弹状态是否为“爆炸”以及右侧一格是否为墙壁决定火花右边缘写 RAM 数据，进入状态 7；

状态 7 根据炸弹状态是否为“爆炸”以及下方一格是否为墙壁决定火花下边缘写 RAM 数据，进入状态 8；

状态 8 根据炸弹状态是否为“爆炸”以及上方一格是否为墙壁决定火花上边缘写 RAM 数据，进入状态 0；

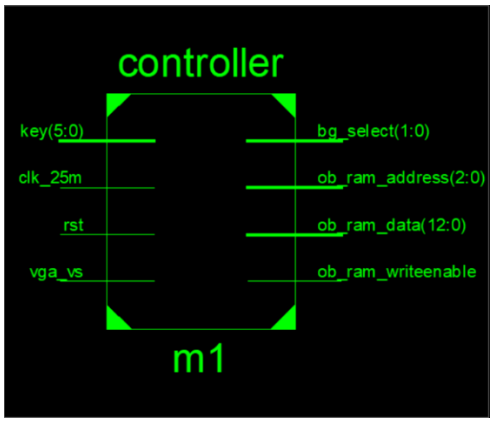

图 3 controller 模块图

2.2.3. m2-displayer 模块

输入：

clk_25m	25 兆赫时钟
rst	复位信号
bg_digin	背景选择输入
ob_digin	物体数据输入
tile_digin1	背景数据输入
tile_digin2	物体选择输入

输出：

vga_hs	VGA 水平同步信号
vga_vs	VGA 垂直同步信号
vga_rgb	VGA 颜色信号
bg_address	背景读 RAM 地址
ob_address	物体读 RAM 地址
tile_address1	背景 ROM 地址
tile_address2	物体 ROM 地址

功能：

2.2.3.1. 调用 timing 模块

调用 timing 模块获得 VGA 同步信号以及行列计数信号。

2.2.3.2. 背景抓取

根据行列计数信号以及背景选择信号决定对应 bgROM 中的背景显示块信息。

2.2.3.3. 物体抓取

根据行列计数信号以及背景选择信号决定对应 obROM 中的背景显示块信息。

2.2.3.4. 背景显示块抓取

根据生成的背景显示块信息从 bgTILE 中获取 RGB 信息。

2.2.3.5. 物体显示块抓取

根据生成的物体显示块信息从 obTILE 中获取 RGB 信息。

2.2.3.6. 图像显示

根据生成的背景和物体 RGB 信息，优先输出物体 RGB 信号，其次输出背景 RGB 信号。

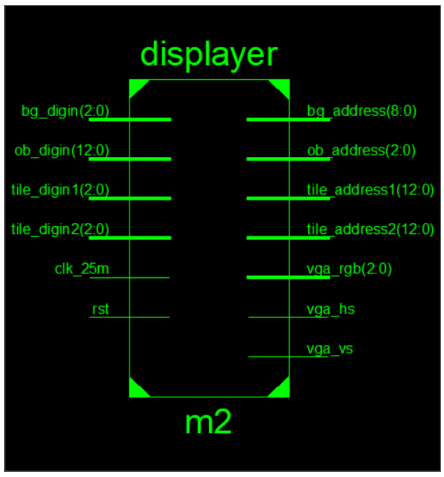

图 4 displayer 模块图

2.2.4. m21-timing 模块

输入：

clk_25m 25 兆赫时钟
rst 复位信号

输出：

hs VGA 水平同步信号
vs VGA 垂直同步信号
hc[9:0] 水平计数
vc[9:0] 垂直计数

功能：

2.2.4.1. 根据显示器参数（分辨率和刷新频率）利用时钟分出同步、计数信号

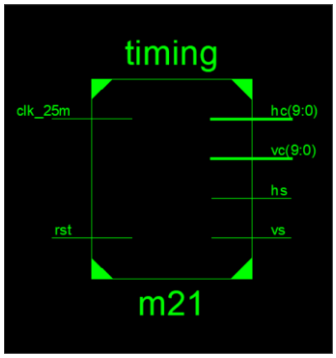

图 5 timing 模块图

2.2.5. m3-objectRAM 模块

输入：

clk_25m 25 兆赫时钟
write_address[2:0] 写 RAM 地址
data[12:0] 写 RAM 数据
write_en 写 RAM 使能
read_address[2:0] 读 RAM 地址

输出：

dig[12:0] RAM 输出

功能：

2.2.5.1. RAM 读写操作，输出要显示的物体的 ROM 数据

在写 RAM 使能时，将 data[12:0]写入对应地址的 RAM 中，在下一个周期进行读操作，获取 ROM 中的物体数据。

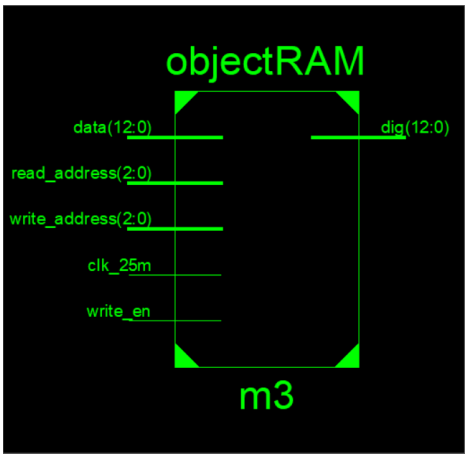

图 6 objectRAM 模块图

2.2.6. m4-bgROM1 模块

输入：

address[8:0] 背景 1ROM 地址

输出：

dig[2:0] 背景 1ROM 数据

功能：

2.2.6.1. 根据地址获取游戏状态背景数据

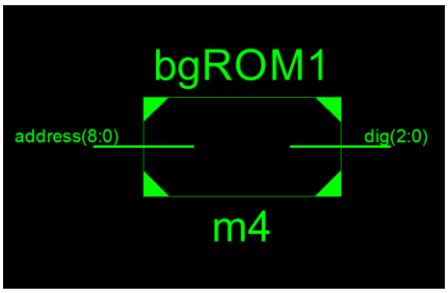

图 7 bgROM1 模块图

2.2.7. m5-bgROM2 模块

输入：

address[8:0] 背景 2ROM 地址

输出：

dig[2:0] 背景 2ROM 数据

功能：

2.2.7.1. 根据地址获取胜利状态背景数据

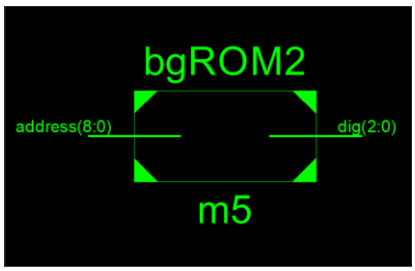

图 8 bgROM2 模块图

2.2.8. m6-bgROM3 模块

输入：

address[8:0] 背景 3ROM 地址

输出：

dig[2:0] 背景 3ROM 数据

功能：

2.2.8.1. 根据地址获取失败状态背景数据

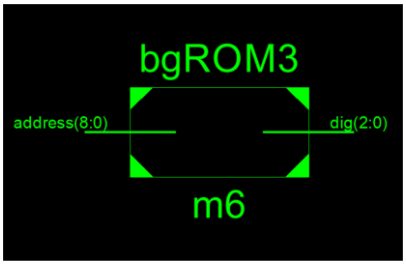

图 9 bgROM3 模块图

2.2.9. m7-bgTILE 模块

输入：

address[12:0]背景显示块 ROM 地址

输出：

dig[2:0] 背景显示块 ROM 数据

功能：

2.2.9.1. 根据地址获取背景显示块 RGB 数据

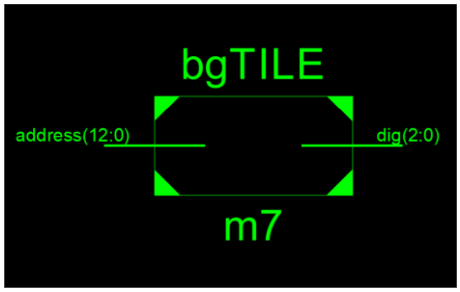

图 10 bgTILE 模块图

2.2.10. m8-obTILE 模块

输入：

address[12:0]物体显示块 ROM 地址

输出：

dig[2:0] 物体显示块 ROM 数据

功能：

2.2.10.1.根据地址获取物体显示块 RGB 数据

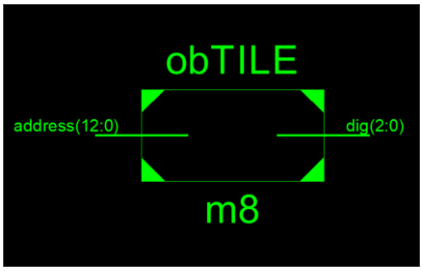

图 11 obTILE 模块图

2.2.10.2.m9-music 模块

输入：

clk　　100 兆赫时钟
reset　　复位信号

输出：

Speaker　　音乐
AUD_SD　　AUD 输出
Buzzer　　蜂鸣器

功能：

2.2.10.3.播放音乐并熄灭蜂鸣器

根据乐谱对应的乐音，转换为对应的频率，由时钟进行控制一个音符的播放时长实现阴郁的播放。

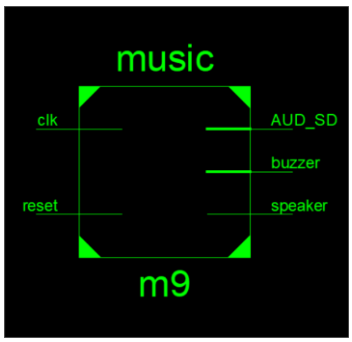

图 12 music 模块图

2.2.11. m10-mux 模块

输入：

bg_select　　背景选择信号
bg_digin_1　　游戏状态背景

bg_digin_2　　胜利状态背景
bg_digin_3　　失败状态背景

输出：
bg_digin_mux 背景选择输出

功能：

2.2.11.1.三选一选择器

根据背景选择信号，从三种游戏状态选取需要的进行输出。

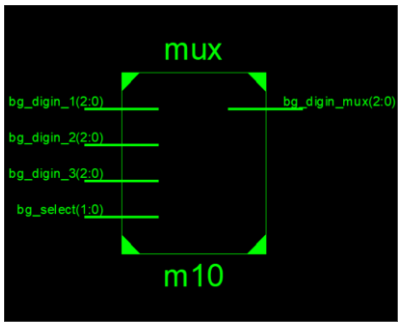

图 13 mux 模块图

2.2.12. m11-f_double 模块

输入：
CLK_IN1　　板载 100 兆赫时钟输入

输出：
CLK_OUT1　50 兆赫分频时钟输出
CLK_OUT2　25 兆赫分频时钟输出

功能：

2.2.12.1.时钟分频

根据分频规则，将 100 兆赫时钟分频为所需的频率时钟。

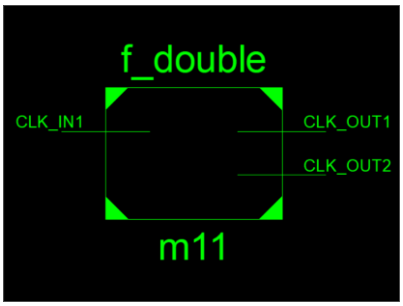

图 14 f_double 模块图

2.3. 硬件设计方案

为了控制炸弹人的移动和炸弹的释放，我们需要使用 5 个开关分别对应相应的功能，开关上下拨动一次即代表进行一次对应操作；为了重新开始游戏并在程序发生不可预见的错误时返回最初状态，需要设置一个复位开关。

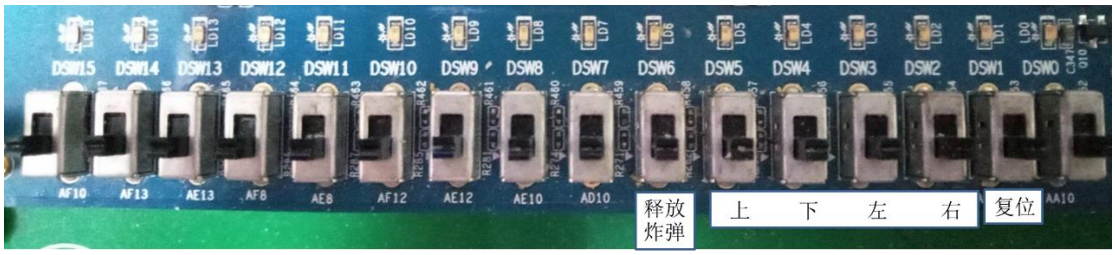

图 15 开关硬件对应图

游戏的显示方式采用 VGA 显示，VGA 为本次课程设计的主要输出端。

图 16 VGA 显示器

为了适当增加游戏的趣味性，添加了音乐模块，音乐的输出依靠于 3.5mm 耳机接口接入耳机后输出。

图 17 音乐输出硬件图

2.4. 模块间逻辑连接方案

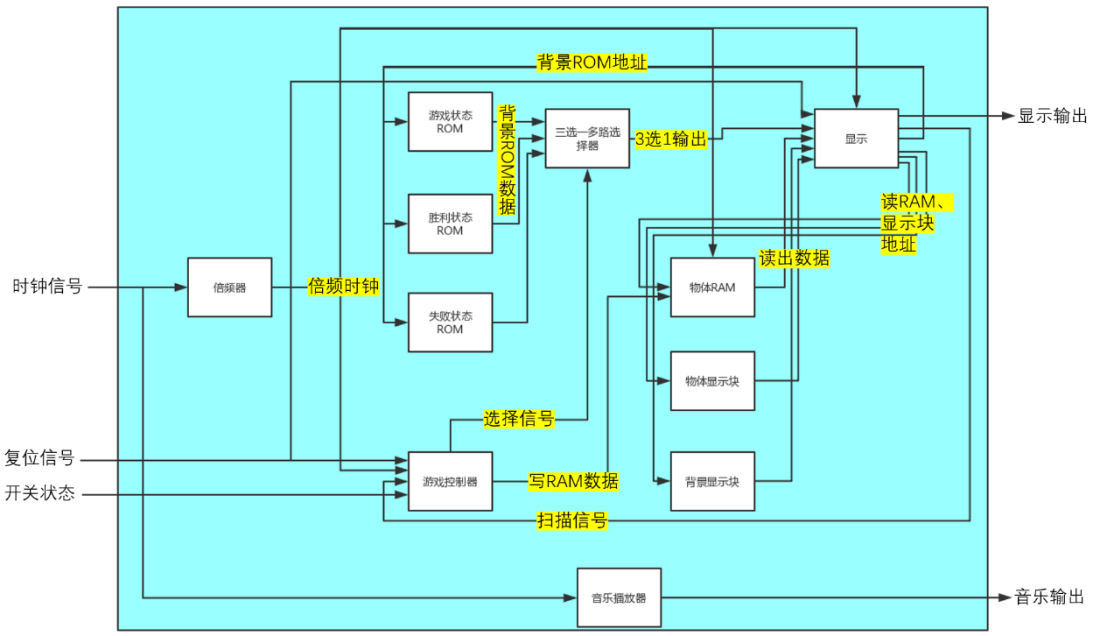

图 18 各模块逻辑功能连接

总体上来看，整个设计最难实现的模块就在于 RAM、游戏控制器以及显示模块三大部分，这三大部分相互之间的数据传输也是需要仔细考虑的，如何利用精简的输入输出达到目的是需要思考的。显示模块和 RAM 与 ROM 的数据传输等难点又都是实验课程所未提及的部分，这给课程设计带来了很大的困难，需要小组查阅较多相关资料，进行由简入繁的工程实践来逐步掌握相关操作。

3. 调试过程分析

3.1. 调试原则

在每个模块内部的部分功能实现后，都进行综合以检查 Warning 以及 Error，对于可以忽略的 Warning（如：赋值位数不匹配等）不做处理，对于较为严重的 Waring，通过调试分析其具体的产生原因，根据调试结果对代码做出改进，最终综合预期结果为不排除每一个 Waring 但可以保证模块的正常工作。

由于 VGA 输出的特殊性，部分模块存在着难以进行仿真、仿真结果不够直观、仿真时长不够长时难以观察到效果、仿真时间太长又有寻找目的结果繁琐（分钟级的仿真）等诸多问题，因此调试总体上采用直接下板或仿真后下板的调试思路，以下板为主、仿真为辅的原则进行调试过程。

在各模块调试过程中，涉及到其他模块输入信号时，均人工赋值为常量，尽量做到模

块独立调试，防止各个模块出现交叉错误造成难以调试的局面出现。虽然这样的调试方法也存在着需要数个调试工程进行辅助、代码修改量大等弊端，但是这可以很好的理清逻辑思路，独立解决问题，对于整个保证工程的健康度来说十分有效。

3.2. 各模块调试方法

对于 m11-f_double，m10-mux 等较为简单的模块，在综合后即可初步判断功能是否正常，但后续仍需相关仿真验证。

在 m2-displayer，m4-ROM1，m5-ROM2，m6-ROM3，m7-bgTILE，m8-obTILE 几个与 VGA 显示有关的模块中，采用了直接下板的方式进行调试，分阶段由简入难地进行调试。在刚入门 VGA 时，仅实现全屏幕纯色显示以了解 VGA 工作方式；逐步过渡到屏幕分区纯色块显示，在熟练掌握了分区色块显示后开始 ROM 以及 TILE 的调试，做到静态显示各个背景、物体。

在 m1-controller 模块中，由于存在多个对象（炸弹人、小鬼、炸弹、爆炸火花以及墙壁等），需要多个不同的时钟进行控制，同时也需要设计较多的时序状态机来实现功能。因此在这一模块中较多的使用了分阶段调试的方法，在每个对象的独立功能实现后（如炸弹人的移动等），进行直接下板验证，在独立功能正常后调试对象间的交叉功能（如小鬼追击炸弹人等），逐步实现各个对象间的互动。

在整个模块中各个时序状态机的功能使用仿真调试更加方便直观，如：炸弹人在释放炸弹后炸弹不爆炸、未显示炸弹直接显示火花、游戏胜利或失败后炸弹人仍出现在界面上等，此类问题都是写 RAM 状态机和炸弹状态机的不完备所造成的，通过仿真波形图可以直观地看到各个状态之间的切换，而下板验证则无法达到此目的。

对于 m3-objectRAM 模块，开始在没有理清 ROM，RAM， displayer，controller 四者间的相互数据传输时，RAM 模块的调试难以下手，在认真阅读了相关资料后，RAM 的读写操作都十分简单移动，代码量小的同时也清楚明晰，不需要进行过多的调试。

m9-music 模块的调试采用了直接下板验证的方式，从发出不同的音符开始逐步完成音乐的播放功能， 由于原理简单，虽然代码量大但是均为重复性代码，调试过程较为简单，模块实现也比较顺利。

3.3. 调试心得

调试是一个工程十分关键的步骤，对于整个工程的重要性不亚于编程过程，对于第一次在没有任何 PPT 及老师讲解的情况下完成一个较大的工程，调试是相对陌生的一个环节，代码量大、工程原理陌生给调试带来了很多困难，尤其对于本工程这样一个游戏工程来说，调试思路与一般工程不同，调试方法也要加以改变。

在状态机的调试过程中，较多的使用了 ISim 中的 Step、断点以及运行至断点功能，一步一步跟踪进程，跟踪状态的切换，观察状态发生非预期改变时刻的各个参数，寻找错误点，以此来完成状态机的调试。

在仿真调试中，勤用常量代替变量，每次改变一个输入变量进行对比调试，分析错误出现的原因，禁用无规律的随意调试，保证每次调试有目的、有意义，这样才能保证模块功能的成功实现。

除此之外，在工程的调试过程中，可以使用多台电脑进行调试，充分利用程序综合等耗费的时间，提高调试效率。

4. 核心模块仿真模拟分析

本工程中最适合也最需要进行仿真调试的模块就是两个重要的时序状态机：炸弹状态机、物体写 RAM 状态机，其余模块如上文第 2 部分直接下板调试，不适合进行仿真模拟调试。对于较为简单的模块如倍频时钟 m11-f_double 模块，由于篇幅所限以及在课程实验中已做过多次分析，此处不做仿真分析。

4.1. 炸弹状态机仿真调试

4.1.1. 炸弹状态机相关说明

在进行波形分析前，先对涉及炸弹状态机的相关变量做出描述：

clk：25 兆赫时钟信号

sysClkW[7:0]：模块时钟信号，每个时钟上升沿周期增 1，达到十进制 255 后归 0。波形上的值为 unsigned decimal（无符号十进制数）。

reset_n：置 0 为复位使能信号，即低电平有效。

iKEY[4:0]：炸弹人控制开关，由高位到低位依次代表：释放炸弹、上、下、左、右，同样为低电平有效。

keyW[4:0]：控制开关转换后真正键值，由高到低代表含义不变，高电平有效。

clkW[7:0]：炸弹计时器，仅在计时使能后且 sysClkW[7:0]=1 时自增 1 达到十进制 255 后归 0。波形上的值为 unsigned decimal（无符号十进制数）。

stateW[1:0]：炸弹状态寄存器。0-未释放，1-释放，2-爆炸。状态 0 转换至状态 1 的条件为：时钟上升沿且 sysClkW[7:0]=1 且 iKEY[4]=0；进入状态 1 后，炸弹计时器开始计时；计时达到 200 后，进入状态 2，计时器复位重新计时，计时达到 50 后，炸弹恢复为状态 0，计时器复位。波形上的值为 unsigned decimal（无符号十进制数）。

4.1.2. 炸弹状态机波形分析

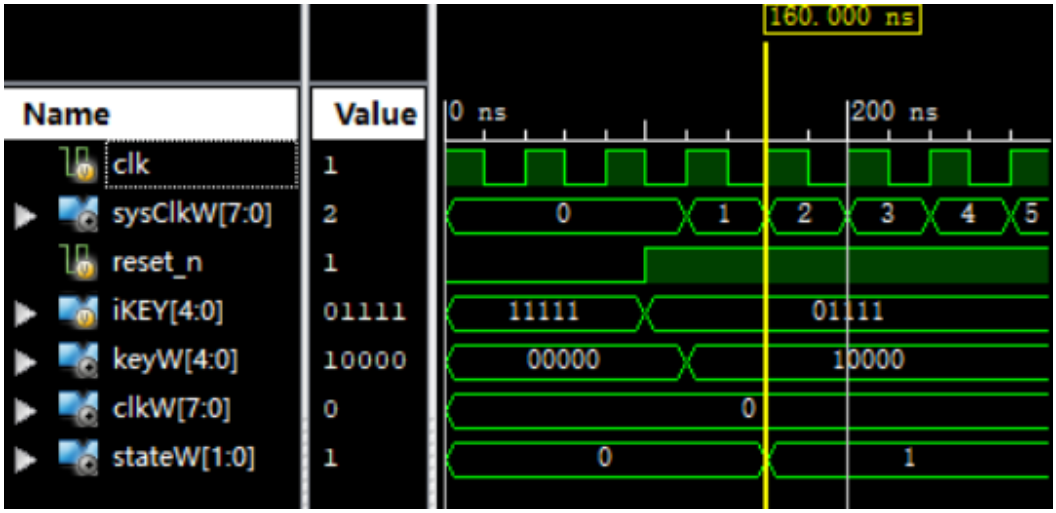

图 19 状态 0 至状态 1 波形图

分析：

0~100ns：将 reset_n 置 0，模拟一次重新开始游戏的操作。

100~120ns：由于重新开始游戏后，这段时间没有经过时钟上升沿，模块不工作，因此 iKEY[4]置为 0，但对应的 keyW[4]无变化。

120ns 之后：reset_n 置 0 复位为 1 后，模块开始正常工作，keyW[4]变为 1，没经过一个时钟周期，sysClkW 增 1，预期结果与波形相符。

160ns：keyW[4]=1，sysClkW[7:0]=1，时钟上升沿，stateW[1:0]=0, 炸弹状态机的现态为状态 0，符合炸弹“释放”状态要求，炸弹状态机进入状态 1，stateW[1:0]=1, clkW[7:0]计时使能，但下一个时钟周期不满足 clkW[7:0]=1，故计时器仍保持 0。预期结果与波形图相符。

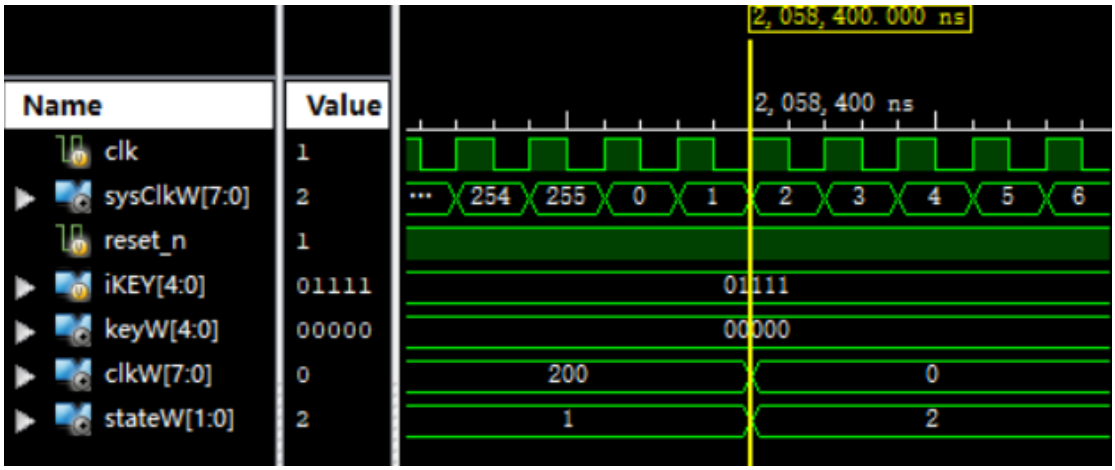

图 20 状态 1 至状态 2 波形图

分析：

2058320ns：sysClkW[7:0]=255，下一个时钟周期复位为 0，预期结果与波形相符。

160ns~2058400ns：clkW[7:0]进行计时，每当 sysClkW[7:0]=1 时自增 1。

2058400ns：clkW[7:0]=200，时钟上升沿，stateW[1:0]=1，炸弹状态机现态为状态 1，符合炸弹“爆炸”状态要求，stateW[1:0]=2，clkW[7:0]计时器复位，clkW[7:0]计时使能，但下一个时钟周期不满足 clkW[7:0]=1，故计时器仍保持 0。预期结果与波形图相符。

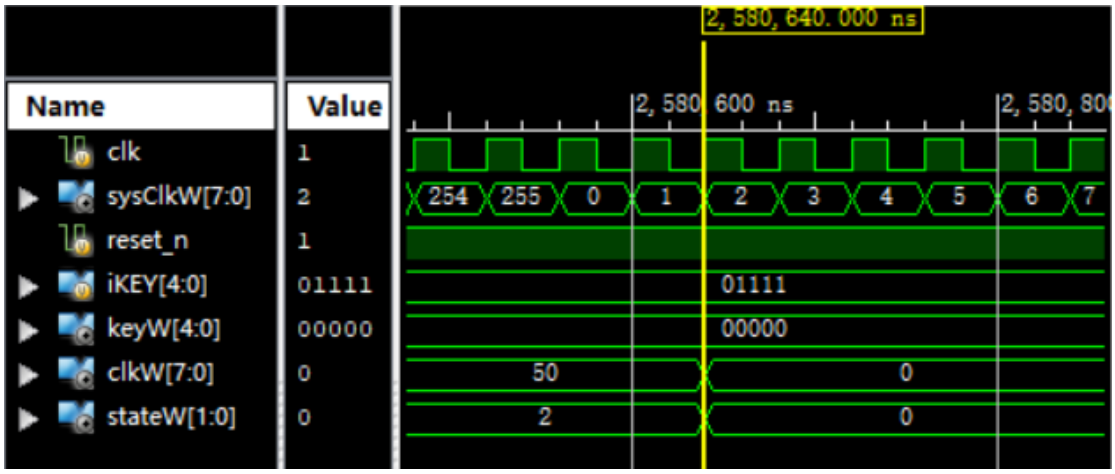

图 21 状态 2 至状态 0 波形图

分析：

2058400~2580640ns：clkW[7:0]进行计时，每当 sysClkW[7:0]=1 时自增 1。

2580640ns：clkW[7:0]=50，时钟上升沿，stateW[1:0]=2，炸弹状态机现态为状态 2，符合炸弹恢复“未释放”状态要求，stateW[1:0]=0，clkW[7:0]计时器复位，clkW[7:0]计时使能，但下一个时钟周期不满足 clkW[7:0]=1，故计时器仍保持 0。预期结果与波形图相符。

4.2. 物体写 RAM 状态机仿真调试

4.2.1. 物体写 RAM 状态机相关说明

在进行波形分析前，先对涉及物体写 RAM 状态机的相关变量做出描述：

clk：25 兆赫时钟信号

sysClk[7:0]：模块时钟信号，每个时钟上升沿周期增 1，达到十进制 255 后归 0。波形上的值为 unsigned decimal（无符号十进制数）。

reset_n：置 0 为复位使能信号，即低电平有效。

iKEY[4:0]：炸弹人控制开关，由高位到低位依次代表：释放炸弹、上、下、左、右，同样为低电平有效。

iVS：VGA 垂直同步信号。

bomb_state[1:0]：炸弹状态寄存器。0-未释放，1-释放，2-爆炸。状态 0 转换至状态 1 的条件为：时钟上升沿且 sysClk[7:0]=1 且 iKEY[4]=0；进入状态 1 后，炸弹计时器开始计时；计时达到 200 后，进入状态 2，计时器复位重新计时，计时达到 50 后，炸弹恢复为状态 0，计时器复位。波形上的值为 unsigned decimal（无符号十进制数）。

bg_select[1:0]：游戏背景状态寄存器。0-游戏状态，1-获胜状态，2-失败状态。波形上的值为 unsigned decimal（无符号十进制数）。

WRAM_state[3:0]：写 RAM 状态寄存器。0-空状态，1-写炸弹人，2-写小鬼，3-写炸弹，4-写火花中心，5-写火花左侧，6-写火花右侧，7-写火花上方，8-写火花下方。状态 0 为空状态，相关数据输出均为 0，在进入写周期后，进入状态 1；状态 1 根据游戏是否结束决定炸弹人写 RAM 数据，进入状态 2；状态 2 根据游戏是否结束决定小鬼写 RAM 数据，进入状态 3；状态 3 根据炸弹状态是否为“释放”决定炸弹写 RAM 数据，处于“释放”则进入状态 0，否则进入状态 4；状态 4 根据炸弹状态是否为“爆炸”决定火花中心写 RAM 数据，处于“爆炸”状态则进入状态 5，否则进入状态 0；状态 5 根据炸弹状态是否为“爆炸”以及左侧一格是否为墙壁决定火花左边缘写 RAM 数据，进入状态 6；状态 6 根据炸弹状态是否为“爆炸”以及右侧一格是否为墙壁决定火花右边缘写 RAM 数据，进入状态 7；状态 7 根据炸弹状态是否为“爆炸”以及下方一格是否为墙壁决定火花下边缘写 RAM 数据，进入状态 8；状态 8 根据炸弹状态是否为“爆炸”以及上方一格是否为墙壁决定火花上边缘写 RAM 数据，进入状态 0。波形上的值为 unsigned decimal（无符号十进制数）。

ob_ram_writeenable：写 RAM 使能信号。

ob_ram_address[2:0]：写 RAM 地址。波形上的值为 unsigned decimal（无符号十进制数）。

ob_ram_data[12:0]：写 RAM 数据（12 位控制是否显示物体，[11:9]位储存物体显示

块编号，[8:4]储存物体 X 坐标，[3:0]储存物体 Y 坐标）。波形上的值为 unsigned decimal（无符号十进制数）。

由于在波形仿真中，通过改变 iKEY[3:0]操作炸弹人逃避小鬼十分复杂且不直观，在调试写 RAM 状态机时，注释掉了小鬼的移动模块，小鬼在原地不动，以保证游戏不进入失败状态，方便波形调试。在本次调试中，炸弹人坐标为(1, 3)，小鬼坐标为(16, 13)。

4.2.2. 物体写 RAM 状态机波形分析

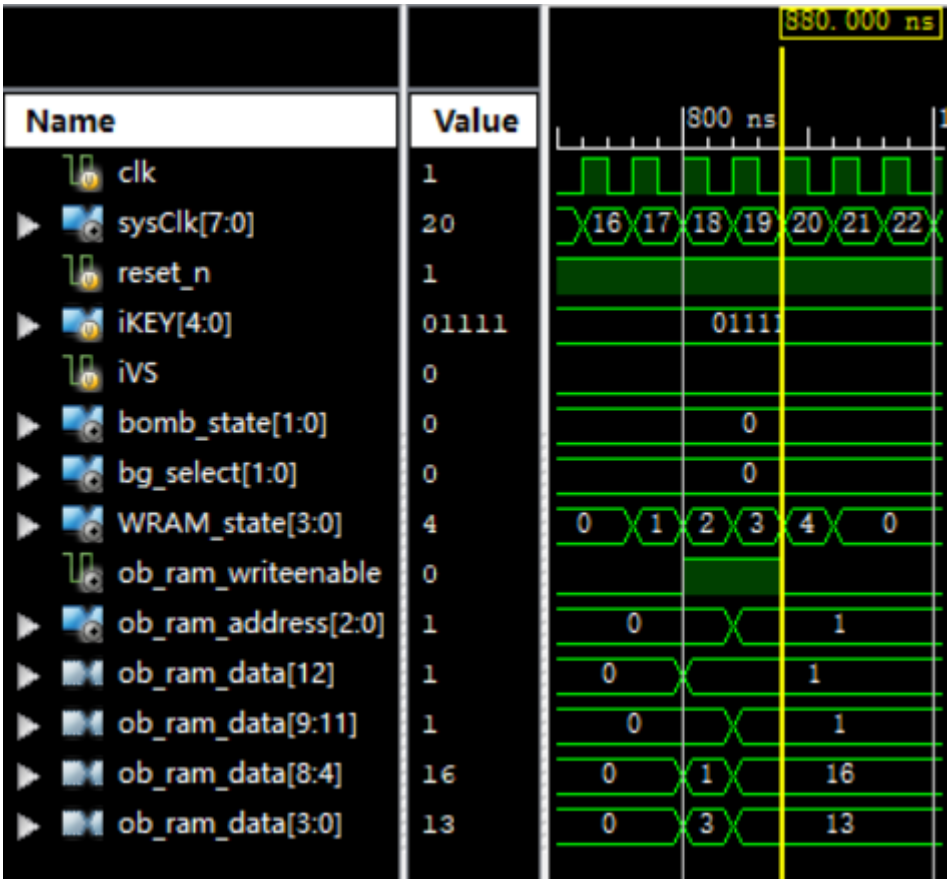

图 22 炸弹“未释放”时的写 RAM 状态机时序图

分析：

760ns 前：状态机处于空状态，输出的 RAM 相关值均为 0。

760ns~920ns：

760~800ns：状态机进入状态 1，游戏未结束，写炸弹人使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 2。

800~840ns：ob_ram_****输出前态状态 1 的相关数据，状态机进入状态 2，游戏未结束，写小鬼使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 3。

840~880ns：ob_ram_****输出前态状态 2 的相关数据，状态机进入状态 3，游戏未结束，炸弹“未释放”，写炸弹使能置 0，其余数据不做更改，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 4。

880~920ns：ob_ram_****输出前态状态 3 及未作改变的部分状态 2 的相关数据，状态机进入状态 4，游戏未结束，炸弹未“爆炸”，写火花中心使能置 0，其余数据不做更

改，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 0。

根据以上分析，波形图中 ob_ram_writeenable 显示两个周期的 1，ob_ram_data[12:0]在两个周期分别写炸弹人于坐标(1, 3)，小鬼于坐标(16, 13)，波形与预期结果相符。

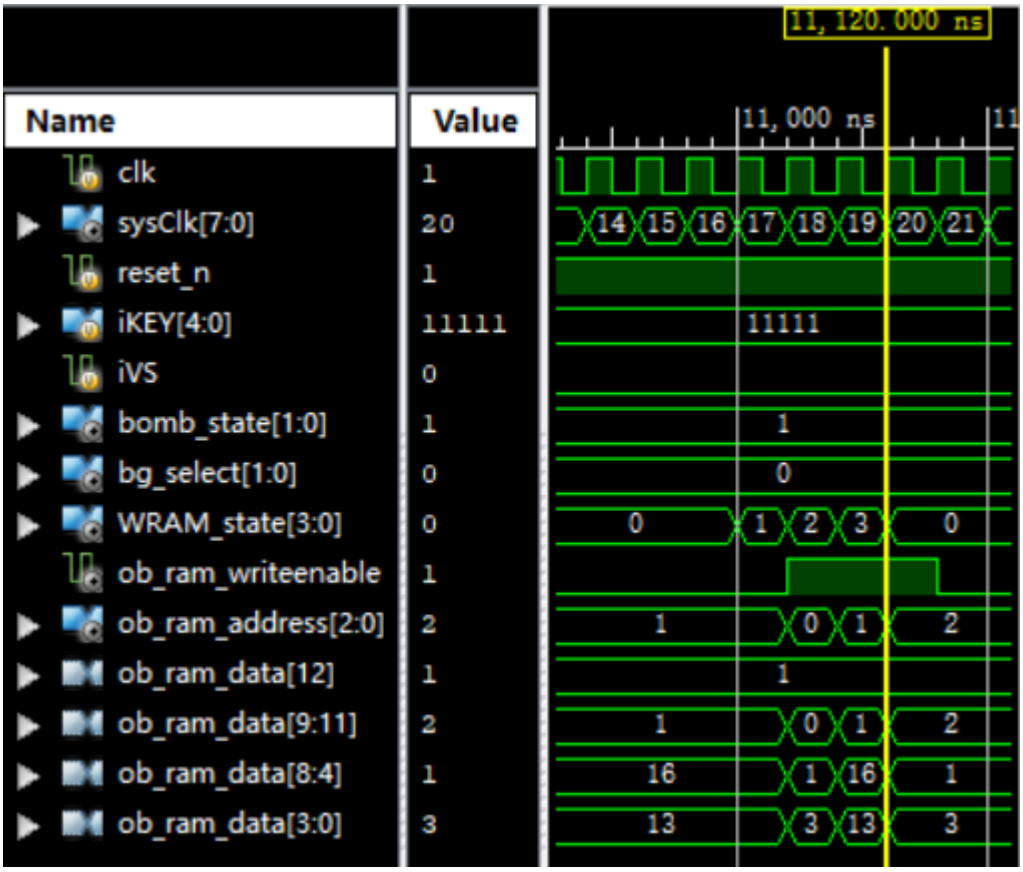

图 23 炸弹“释放”时的写 RAM 状态机时序图

分析：

11000ns 前：状态机处于空状态，输出的 RAM 相关值均为 0。

11000ns~11120ns：

11000~11040ns：状态机进入状态 1，游戏未结束，写炸弹人使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 2。

11040~11080ns：ob_ram_****输出前态状态 1 的相关数据，状态机进入状态 2，游戏未结束，写小鬼使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 3。

11080~11120ns：ob_ram_****输出前态状态 2 的相关数据，状态机进入状态 3，游戏未结束，炸弹“释放”，写炸弹使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 0。

根据以上分析，波形图中 ob_ram_writeenable 显示三个周期的 1，ob_ram_data[12:0]在三个周期分别写炸弹人于坐标(1, 3)，小鬼于坐标(16, 13)，炸弹于坐标(1, 3)，波形与预期结果相符。

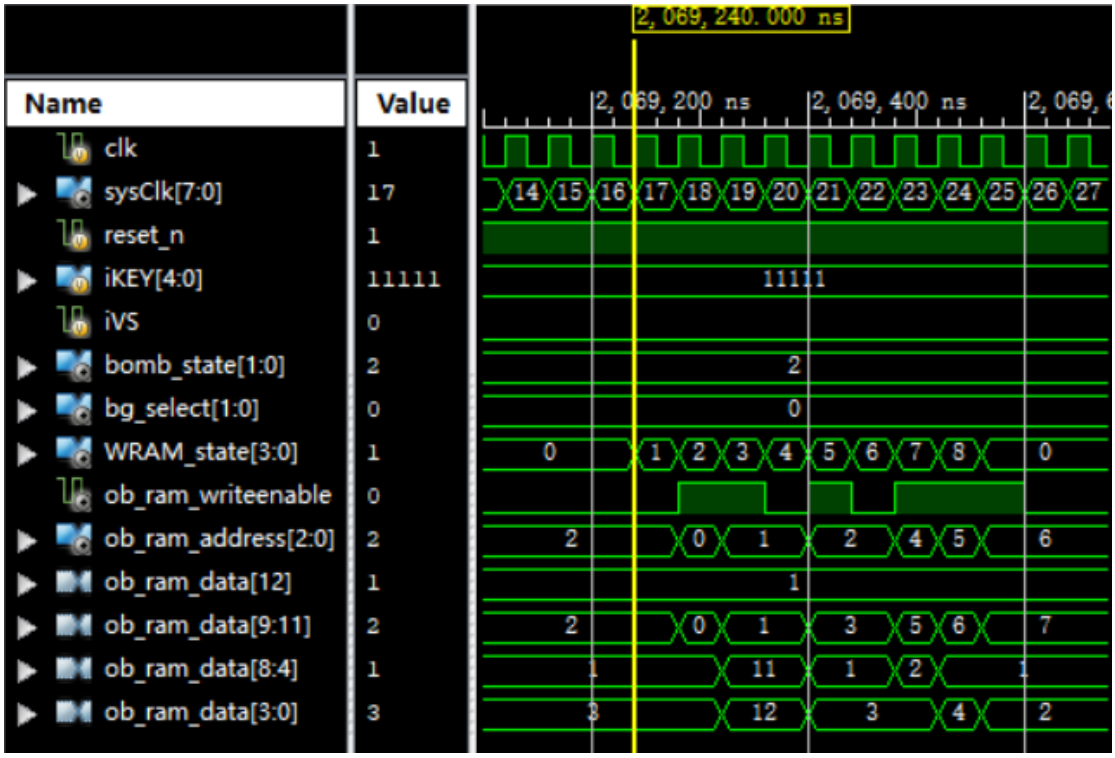

图 24 炸弹“爆炸”时的写 RAM 状态机时序图

分析：

2069240ns 前：状态机处于空状态，输出的 RAM 相关值均为 0。

2069240ns~2069560ns：

2069240~2069280ns：状态机进入状态 1，游戏未结束，写炸弹人使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 2。

2069280~2069320ns：ob_ram_****输出前态状态 1 的相关数据，状态机进入状态 2，游戏未结束，写小鬼使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 3。

2069320~2069360ns：ob_ram_****输出前态状态 2 的相关数据，状态机进入状态 3，游戏未结束，炸弹“爆炸”，写炸弹使能置 0，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 4。

2069360~2069400ns：ob_ram_****输出前态状态 3 的相关数据，状态机进入状态 4，游戏未结束，炸弹“爆炸”，写火花中心使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 5。

2069400~2069440ns：ob_ram_****输出前态状态 4 的相关数据，状态机进入状态 5，游戏未结束，炸弹“爆炸”，火花左侧检测到墙壁，写火花左侧使能置 0，其他数据不做更改，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 6。

2069440~2069480ns：ob_ram_****输出前态状态 5 及部分未更改的状态 4 的相关数据，状态机进入状态 6，游戏未结束，炸弹“爆炸”，火花右侧检测无墙壁，写火花右侧使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 7。

2069480~2069520ns：ob_ram_****输出前态状态 6 的相关数据，状态机进入状态 7，游戏未结束，炸弹“爆炸”，火花下方检测无墙壁，写火花下方使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 8。

2069520~2069560ns：ob_ram_****输出前态状态 7 的相关数据，状态机进入状态 8，游戏未结束，炸弹“爆炸”，火花上方检测无墙壁，写火花上方使能置 1，于下个时钟

周期输出相关数据（坐标及编号），次态置为状态 8。

根据以上分析，波形图中 ob_ram_writeenable 显示六个周期的 1，ob_ram_data[12:0]在六个周期分别写炸弹人于坐标(1, 3)，小鬼于坐标(11, 12)（小鬼坐标另外赋值过一次），火花中心于坐标(1, 3)，火花右侧于坐标(2, 3)，火花下方于坐标(1, 4)，火花上方于坐标(1, 2)，波形与预期结果相符。

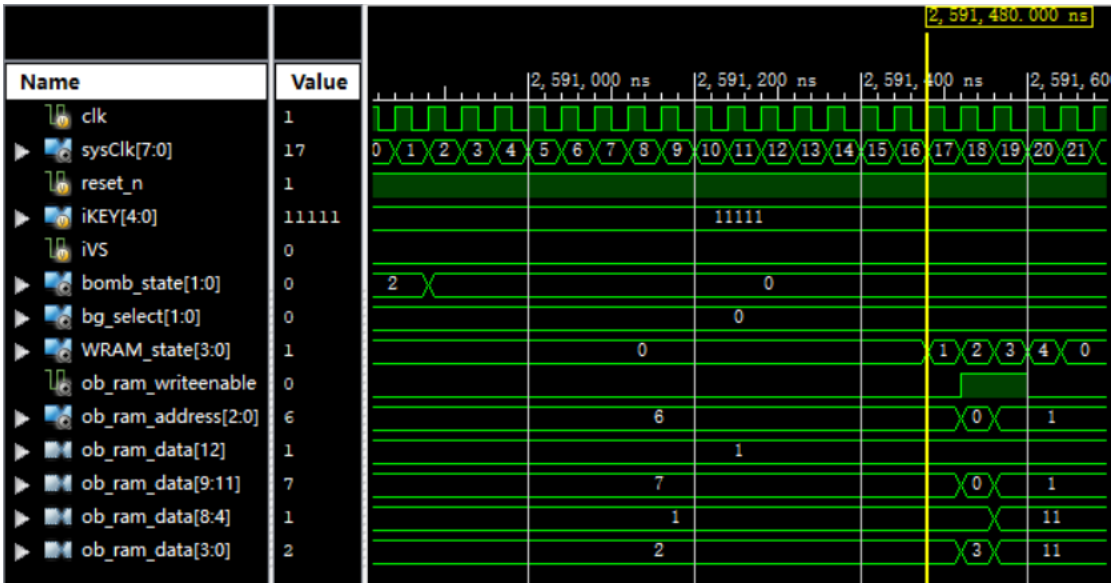

图 25 炸弹恢复“未释放”时的写 RAM 状态机时序图

分析：

2590880ns：炸弹恢复为“未释放”状态。

2591480ns 前：状态机处于空状态，输出的 RAM 相关值均为 0。

2591480ns~2591640ns：

2591480~2591520ns：状态机进入状态 1，游戏未结束，写炸弹人使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 2。

2591520~2591560ns：ob_ram_****输出前态状态 1 的相关数据，状态机进入状态 2，游戏未结束，写小鬼使能置 1，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 3。

2591560~2591600ns：ob_ram_****输出前态状态 2 的相关数据，状态机进入状态 3，游戏未结束，炸弹“未释放”，写炸弹使能置 0，其余数据不做更改，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 4。

2591600~2591640ns：ob_ram_****输出前态状态 3 及未作改变的部分状态 2 的相关数据，状态机进入状态 4，游戏未结束，炸弹未“爆炸”，写火花中心使能置 0，其余数据不做更改，于下个时钟周期输出相关数据（坐标及编号），次态置为状态 0。

根据以上分析，波形图中 ob_ram_writeenable 显示两个周期的 1，ob_ram_data[12:0]在两个周期分别写炸弹人于坐标(1, 3)，小鬼于坐标(11, 11)（小鬼坐标另外赋值过一次），波形与预期结果相符。

5. 系统测试验证与结果分析

5.1. 功能测试

经过一段时间的对代码的测试修改，最终完成结果如下图：

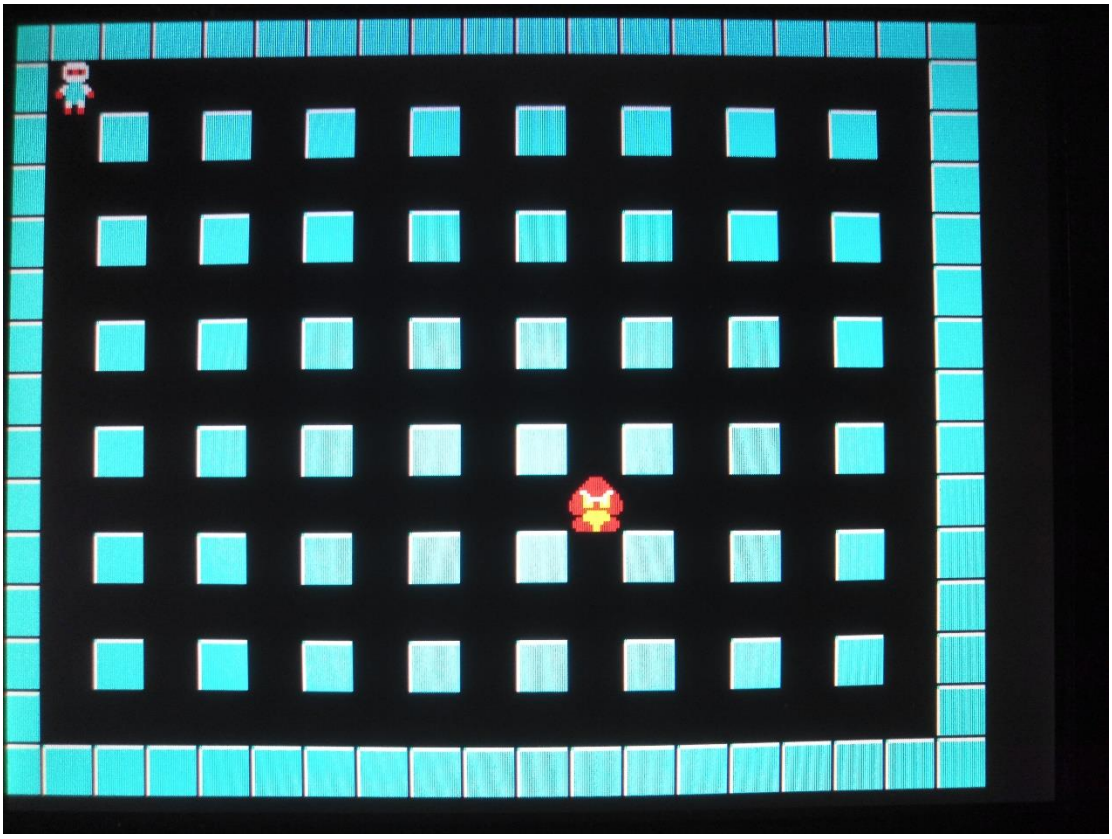

图 26 游戏界面

目前为止已经完成并成功通过测试的功能有：

1) 正常游戏界面和游戏成功/失败界面可以成功显示。
2) 正常游戏中利用开关控制炸弹人上下左右移动功能正常。
3) 正常游戏中炸弹人可以在玩家操作情况下在指定位置放置炸弹。
4) 炸弹放置以及爆炸效果显示正常。
5) 炸弹人被小鬼追上时，可以判断玩家游戏失败，并顺利显示出“GAME OVER”界面。
6) 小鬼在碰上炸弹爆炸时，可以判断玩家游戏胜利，并顺利显示出“WIN”界面。
7) 根据开关定义，成功实现复位功能，任何情况下，只要拨动复位开关，游戏都重新开始。
8) 删掉了一直鸣叫的蜂鸣器 Buzzer，加入播放音乐的模块，让其可以播放音乐，为玩家提供良好的游戏环境。

5.2. 技术参数测试

5.2.1. 全局复位的 Reset 测试

Reset 适用于游戏结束时候的重新开始，也适用于游戏过程中任何时候的复位，在游戏进行中和游戏结束后都可以拨动 Reset 界面实现游戏的全局复位，复位后，炸弹人和小鬼分别出现在游戏地图的左上角和右下部。

5.2.2. 游戏地图的测试

游戏中地图的边界、墙壁和障碍物是规定了炸弹人合法的移动路径，我们着重测试了炸弹人和小鬼在地图多个位置的移动，确认了不会出现穿墙、移动出界等 bug。

5.2.3. 游戏输赢的判定

此游戏的输赢十分简单，玩家控制的炸弹人被小鬼追到即为输，用炸弹炸死小鬼即为赢。根据多次的测试，一旦出现这两种情况都会触发输赢判定，跳转到“WIN”或者“GAME OVER”的界面。

5.2.4. 炸弹的放置与爆炸

测试结果正常，玩家可以选择在任意可通过的地方放置炸弹，炸弹能够在约 2s 后爆炸，只要炸弹爆炸时(火花出现约 0.5s 内均算作爆炸时)小鬼与炸弹之间间隔在 1 个方格以内，小鬼就会被炸死。

5.3. 系统演示与操作说明

5.3.1. 分析验证结果

在实验过程中，已经进行了烧录板子的实验。下载验证后，游戏可以正常运行。各种情况判断正常，相关界面显示也正常，复位信号也可以正常输入，VGA 显示正常，音乐播放正常，基本没有发现什么 bug。

5.3.2. 存在的问题及原因

将完整的程序烧录到实验板上后，可以成功实现预期的功能，不可否认的是，整个游戏也存在一定的问题：

1) 拨动复位开关后，有时会需要较长的时间，游戏界面才会显示在屏幕上。可能的原因是从 IP 读取存储需要的时间较长，会出现所谓的“卡顿”现象。
2) 上下左右的开关拨动并不是非常人性化，因为开关拨动，每上下拨动一次，游戏炸弹人

才会移动一格，并且因为开关是并排一行的位置，所以很容易分不清上下左右，导致游戏体验不是很好。

3) 炸弹每次只能放置一个，在上一个炸弹还没爆炸时，如果再次拨动放置炸弹的开关，就会在现在的位置重新放置一个炸弹，原来的炸弹会消失。出现这个情况的原因，是一开始设置游戏时，就没有考虑炸弹是否能一次放置多个，最后做出的就是每次只能放置一个炸弹。
4) 小鬼走到炸弹上时，炸弹若处于未爆炸状态，则小鬼不会死亡，不会判断玩家胜利，因为我们设定的小鬼死亡判定是被炸到，而炸弹爆炸是受时间控制的，不受小鬼是否踩到炸弹影响。后续完善可以考虑将炸弹爆炸再加一个判断条件，在小鬼踩上炸弹时也会触发炸弹爆炸，这样增加游戏的趣味性，并将游戏难度降低。
5) 每次一开始就是正常游戏进行状态，因为我们没有设置游戏开始界面，为提升游戏体验，可以考虑加入游戏开始界面，让用户进行选择是否开始游戏。
6) 游戏一旦进行就无法暂停，可以考虑加入暂停模式。同时，为提高趣味性，可以加入计分模式，可以循环玩下去，就不存在玩家胜利的界面，不过这样因为游戏较简单，所以出现失败情况可能会耗时较长，因此没有加入计分模块。可经过后续考虑进行一定优化。

5.3.3. 系统操作说明

在载入游戏后，会直接进入游戏界面，如下图所示：

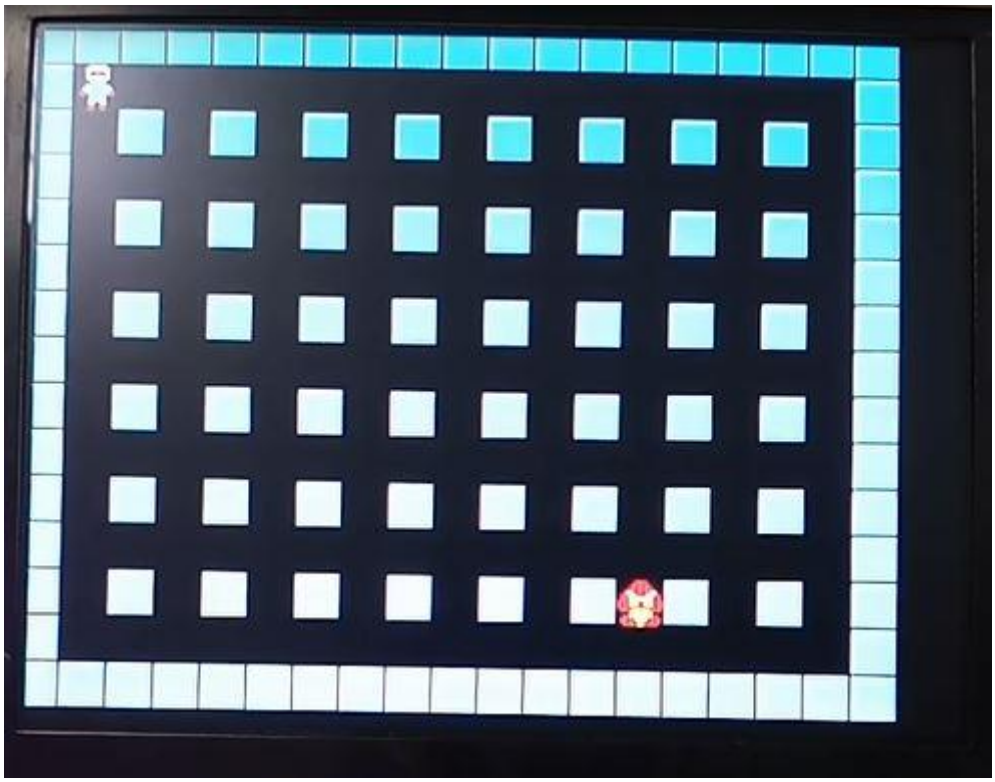

图 27 游戏状态界面

小鬼会根据最短路径向炸弹人移动，并随着炸弹人位置的改变而改变移动方向。

开关可以控制炸弹人的上下左右移动，开关具体分配如下：

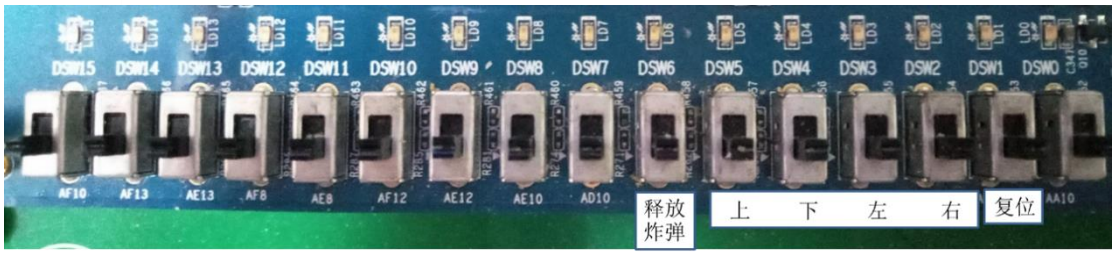

图 28 操作开关图

下图为放置炸弹的界面：

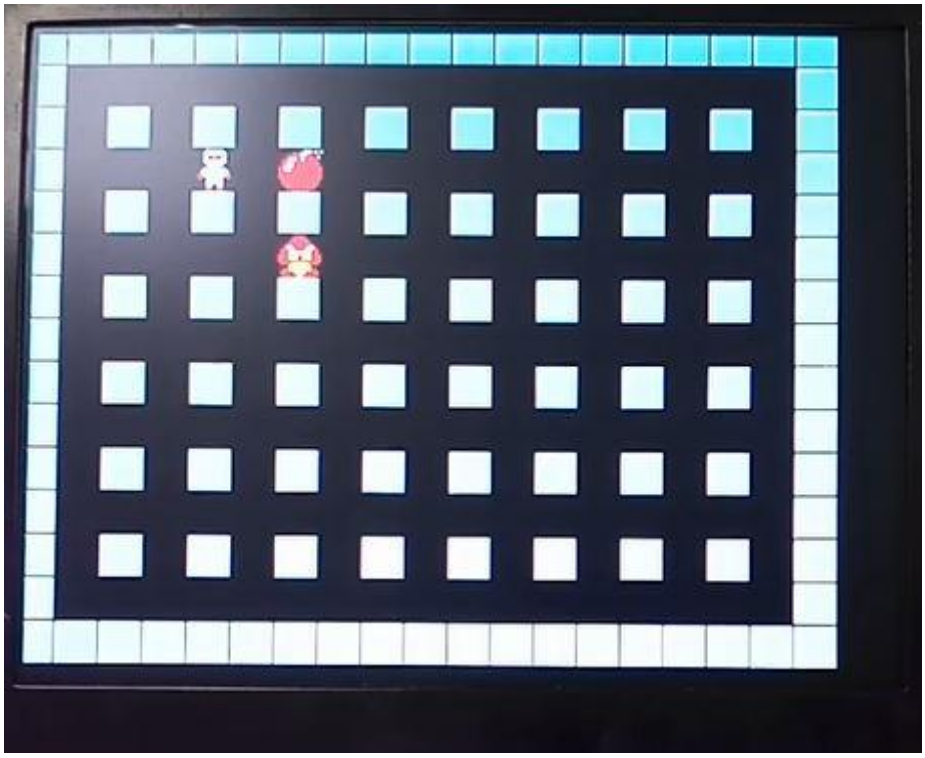

图 29 炸弹人释放炸弹

下图为炸弹爆炸界面:（炸弹爆炸特效为以炸弹为中心，向上下左右四个方向放出火花，因为此图炸弹左右都为墙，所以只显示出了，上下两个方向的火花）

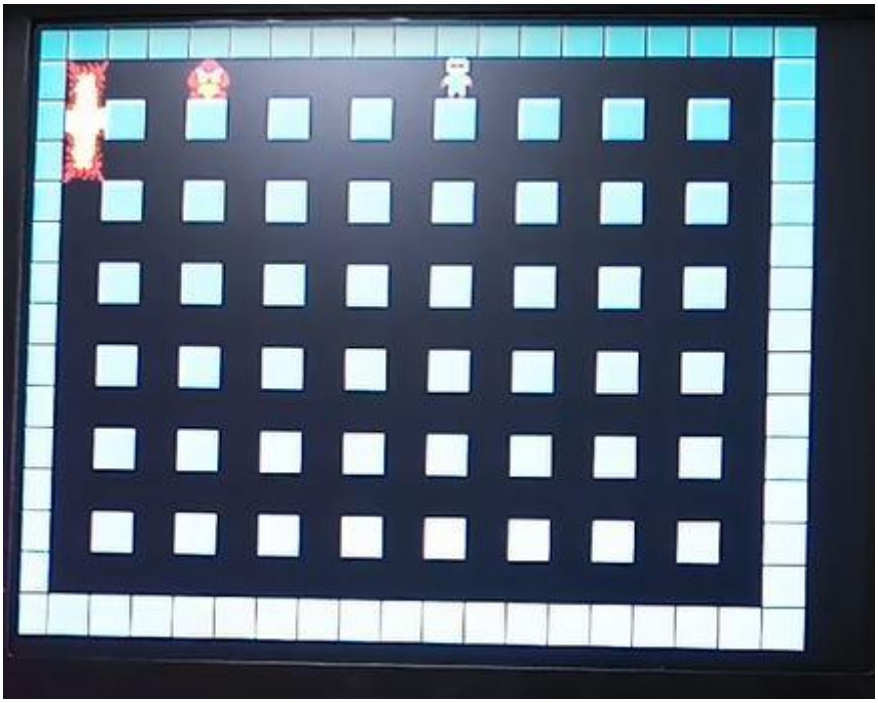

图 30 炸弹爆炸

下图为胜利判断界面：（下一步，小鬼被炸死，玩家胜利，游戏结束）

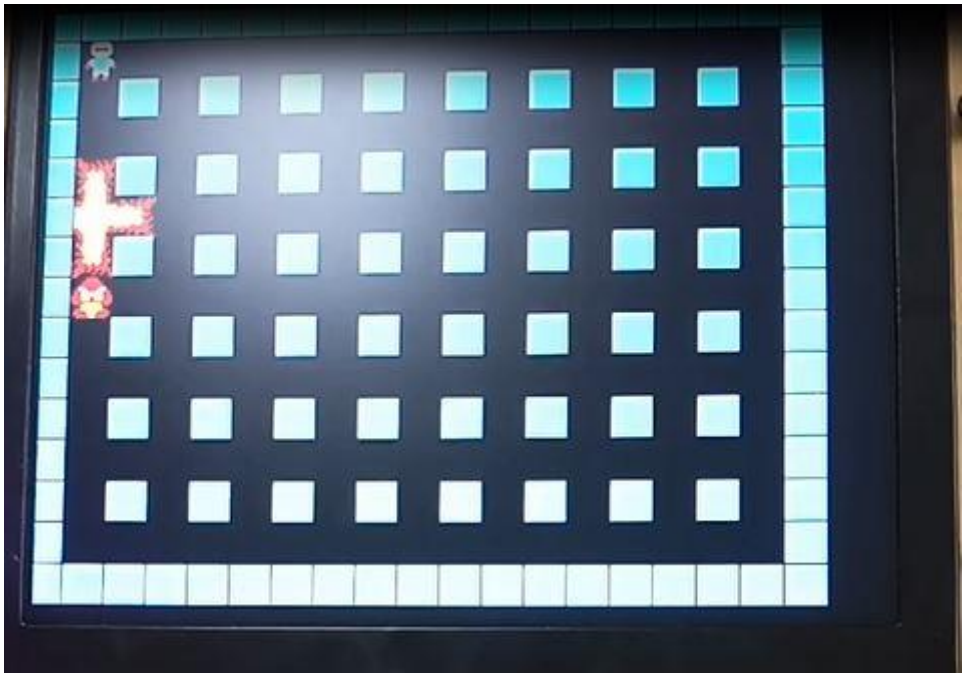

图 29 小鬼死亡前最近一次刷新界面

下图为胜利显示界面：

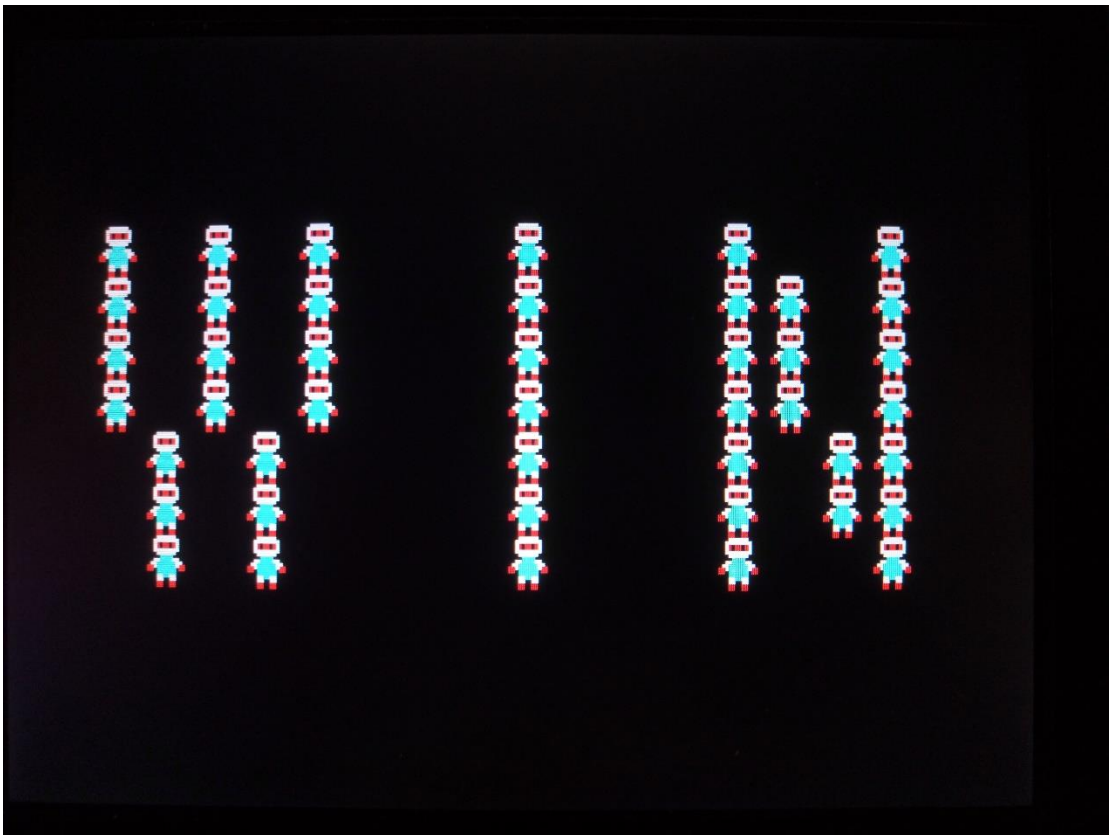

图 30 炸弹人游戏胜利界面

下图为死亡判断界面：（下一步，小鬼吃到炸弹人，玩家失败，游戏结束）

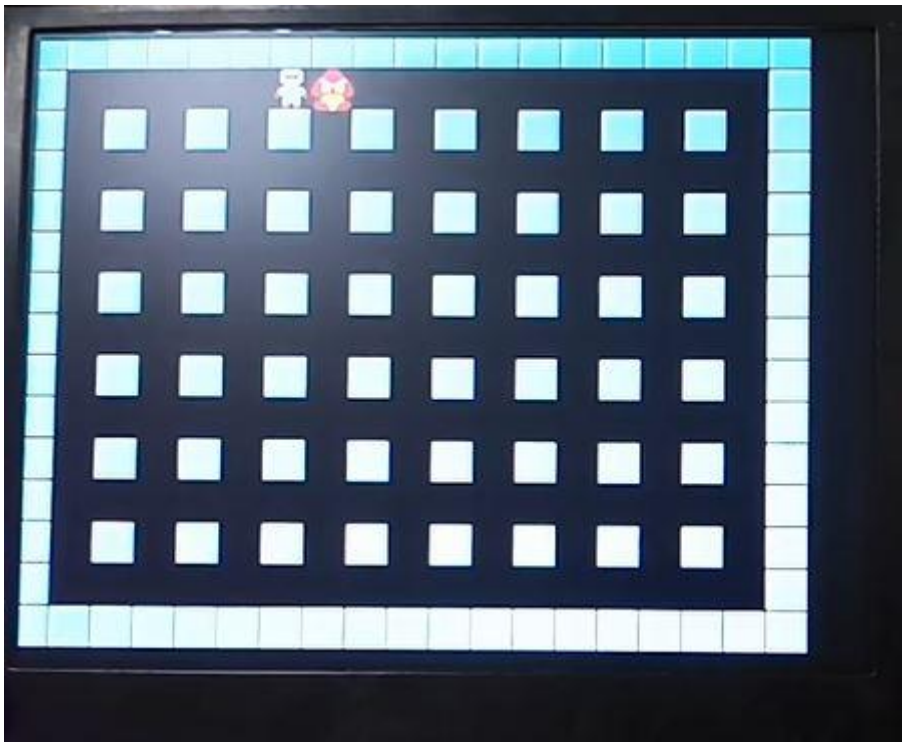

图 31 炸弹人死亡前最近一次刷新界面

下图为游戏失败界面：

图 32 炸弹人游戏失败界面

6. 结论与展望

通过这次课程设计，小组学习到了很多数字逻辑设计课程的拓展内容，如：VGA、音乐等，帮助我们理解并运用了相关知识，实现了自己预期的设计，给予了我们一定的成就感。

但是可以看到的是虽然我们完成了一定的设计实验，但是我们所完成的小游戏只能是一种简单的成果，不具有强大的可操作性，相对而言也比较简单，功能也不够完备。希望可以通过自己未来的学习，实现更加完美的设计。

加上这次对于 VGA 的使用，结合在实验课上所学习的其他的功能，都让我们感觉自己非常有收获，但是两节实验课的课时安排不是非常能满足我们的实验要求。在实验室，大家全身心投入实验设计，时间过得很快，每次离开实验室，都依依不舍。希望能增加实验室的课时，让大家对于实验能有更深的掌握。

7. 源代码

由于 ROM 模块代码重复度高，导致行数过多，源代码不在报告放出，全部源代码文件请见文件夹 src。

8. 参考文献

[1] 陆重阳, 卢东华. FPGA 技术及其发展趋势[J]. 微电子技术, 2003, 31(1):5-7.
[2] 刘小平, 何云斌, 董怀国. 基于 Verilog HDL 的有限状态机设计与描述[J]. 计算机工程与设计, 2008, 29(4):958-960.
[3] 基于 FPGA 的 verilog HDL 语言设计优化[J]. 王春旭,周晓平,王黎黎. 电子元器件应用. 2008(11)
[4] Verilog HDL 语言在 FPGA/CPLD 开发中的应用[J]. 彭保,吴坚,于春梅,马建国. 今日电子. 2004(05)
[5] 基于 Verilog HDL 语言的 FPGA 设计[J]. 彭保,范婷婷,马建国. 微计算机信息. 2004(10)

9. 组内成员分工及贡献比例

9.1. 成员分工

工程代码及调试：郝家辉
设计报告：郝家辉、罗昊、马存
视频录制及后期制作：郝家辉、罗昊、马存

9.2. 贡献比例

组长：郝家辉 3160101827 40%
组员：罗　昊 3160101822 30%
　　马　存 3160101779 30%