**Research Project**

**基于FPGA的小恐龙跑酷游戏设计报告**



**3170101507 包文鼎**

**3170101508 邬树辉**

**3170101509 钟嘉农**

**Date:2018-1-14**

# 一、背景介绍

如果你不知道小恐龙游戏的话，请打开电脑中的Chrome浏览器，在地址栏中输入chrome://dino/，然后按空格体验游戏。而我们的这个项目是一个运行在FPGA上的同款小恐龙游戏。为了维护世界和平和防止小恐龙的灭绝，玩家需要控制小恐龙的跳跃来躲避路径上的仙人掌，玩家坚持的时间越久，分数越高。随着游戏的进行，仙人掌的运动速度会越来越快直到一个峰值。游戏场景会发生白天黑夜的变化让游戏变得更cooooooooool。

# 二、设计说明

## 设计开发环境

实验平台：SWORD

开发环境：Xilinx ISE

硬件描述语言：Verilog

## 输入输出交互选择

### 输入

按钮、拨动开关

### 输出

调用四位7段数码管、LED、VGA

### 操作说明

这个项目有两个外界输入，一个是SWORD最左侧的开关（AF10），用于控制游戏开始，另一个是矩阵按钮中的最左下角那个（V18，W16），用于控制跳跃。刚programme完毕之后显示的是没有game\_over但是游戏暂停的状态，拨动开始开关（要置回0）可以开始游戏。按下跳跃按钮小恐龙会跳跃，以躲避障碍物。数码管的分数会随着时间递增。当小恐龙撞上障碍物后游戏会暂停，LED全亮表明game\_over，计分板暂停计分。此时如果想重新开始需要再拨动开关，此时分数会清零，游戏状态会重置。

# 三、核心模块说明

## Top

### 输入

input clk：系统时钟信号

input rstn：重置输入（本来想用但是没用到）

input [15:0]SW ：15个开关输入

input wire BTN：一个按钮输入

### 输出

output wire LED[7:0]：8个LED输出

output hs：VGA水平同步输出

output vs：VGA垂直同步输出

output [3:0] r：VGA红色输出

output [3:0] g：VGA绿色输出

output [3:0] b：VGA蓝色输出

output wire BTNX4：按钮使能

output wire [3:0]AN：Arduino子板数码管使能

output wire [7:0]Segment：Arduino子板数码管段码输出

### **图**

（截RTL那个好了）

### 功能

调用所有模块，将Game模块生成的状态输出到数码管和vga上。

## Game

### 输入

input wire game\_clk：游戏时钟

input wire jump：跳跃信号

input wire start：开始信号

### 输出

output reg night：白天夜晚输出

output reg [11:0] dino\_y：恐龙中心y位置输出

output reg signed [11:0] obstacle\_x：障碍物x位置输出

output reg game\_over：游戏失败信号

output reg [1:0] dino\_state：恐龙状态

### **图**

（截RTL那个好了）

### 功能

以时钟输入作为游戏时钟，根据parameters.v中的游戏参数控制游戏元素的状态。根据输入的start信号控制游戏的开始和重置，根据输入的jump信号控制小恐龙的跳跃。维护小恐龙的竖直速度以完成完整的跳跃过程。判断小恐龙和仙人掌的碰撞以输出game\_over。对游戏始终计数来控制白天黑夜以及小恐龙运动状态的变化。

## Vga

**输入**

input wire[11:0] d\_in; // bbbb\_gggg\_rrrr, pixel

input wire vga\_clk; // 25MHz

input clrn;

2.输出

output reg [8:0] row\_addr; // pixel ram row address, 480 (512) lines

output reg [9:0] col\_addr; // pixel ram col address, 640 (1024) pixels

output reg [3:0] r,g,b; // red, green, blue colors

output reg rdn; // read pixel RAM (active\_low)

output reg hs,vs; // horizontal and vertical synchronization

3.图（截RTL那个好了）

4.这个模块来自课程网站上的vgademo。它读入12位的颜色数据，控制vga板上对应位置的rgb颜色。

## Renderer

### 输入

input wire [11:0] dino\_y：来自Game的恐龙y信号

input wire [1:0] dino\_state：来自Game的恐龙状态信号

input wire [11:0] obstacle\_x：来自Game的障碍物x位置信号

input wire night：来自Game的白天夜晚信号

input pixel\_clk：像素绘制时钟

input [11:0] vga\_x：当前像素位置X

input [11:0] vga\_y：当前像素位置Y

### 输出

output reg pixel：当前像素是否点亮，点亮为1

图（截RTL那个好了）

功能：负责根据地面、小恐龙、障碍物的位置，给出每一个像素点的颜色。小恐龙、仙人掌的图像由几个常值矢量（ROM）保存，通过计算当前像素与它们位置的相对位置，来判断是不是在图像范围内，以及需不需要点亮像素。

## DispNum

### 输入

input wire clk: 来自Top的游戏时钟

input wire RST: 1'b1

input wire num [15:0]: 来自Top的分数信号

input wire points[3:0]: 小数点信号，为4'b0

input wire LES[3:0]: 使能信号，为4'b0

### 输出

output wire AN[3:0]: 输出使能信号

output wire Segment[7:0]: 输出显示信号到四位七段数码管

## Score

### 输入

input wire clk: 来自Top的游戏时钟

input wire L\_D: 来自Top的控制计数暂停信号

input wire clear: 来自Top的清零信号

### 输出

output wire[15:0]: 16位BCD码表示4位十进制数，输出到Top模块

## Parameters

提供控制游戏的所有参数（常量），便于优化操作体验。还保存了小恐龙、仙人掌的二值位图，它们是通过一个自制的图像二值采样脚本得到的。你可以通过退后一点、摘掉眼镜、观察编辑器（VS Code）缩略图等方式看到图像。

# 四、调试过程分析

## 1.RTL综合电路示意图

# C:\Users\lifelongwar\Documents\Tencent Files\769808731\FileRecv\top.PNG

# 主程序外部RTL电路示意图）

# C:\Users\lifelongwar\Documents\Tencent Files\769808731\FileRecv\all.PNG

# (主程序内部RTL电路示意图)

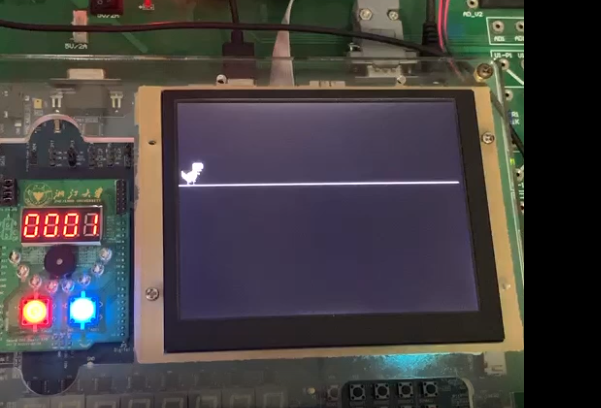
在完成工程的调试后，我们通过RTL电路图判断主程序的输入输出正确，之后我们下载.bit文件到实验板上进一步验证程序的正确性。

## 2.调试

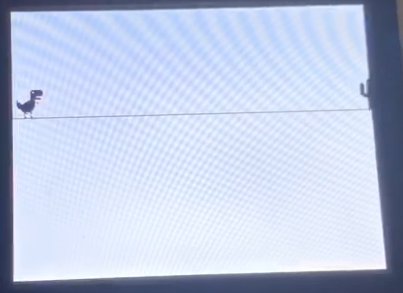


在第一次调试中，我们发现在按下跳跃按钮的时候，小恐龙并没有按照预期的效果向上跳，而是向下做自由落体运动至屏幕外，在经过对程序代码和实际效果的分析，我们发现设置小恐龙跳跃状态下初始位移方向的变量与预期相反，导致小恐龙位移方向相反。在修正bug后并且重新加载下载文件后，我们程序运行结果基本符合预期。

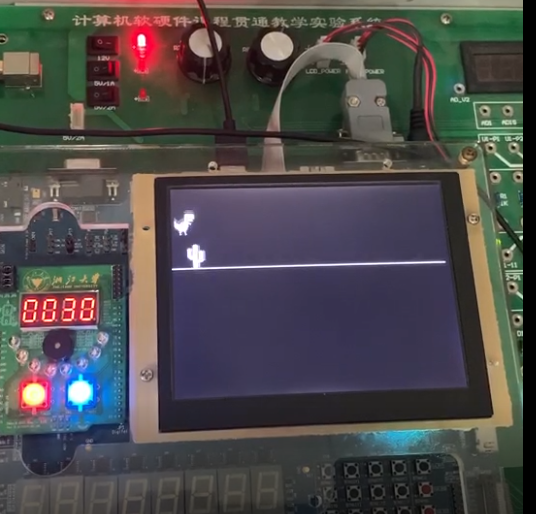
**(1) 在游戏结束状态下拨动开关，小恐龙开始移动并且计数器开始计数**



**（2）每隔一定时间后，出现黑夜白天循环变化**

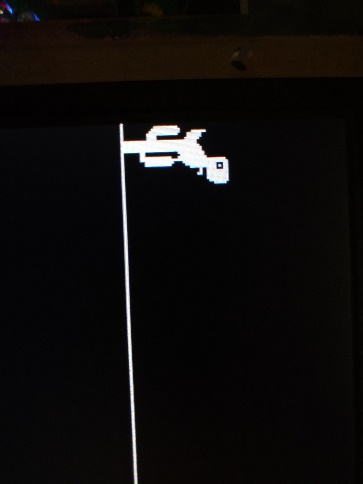


**（3）按动跳跃按钮后，小恐龙实现跳跃功能**



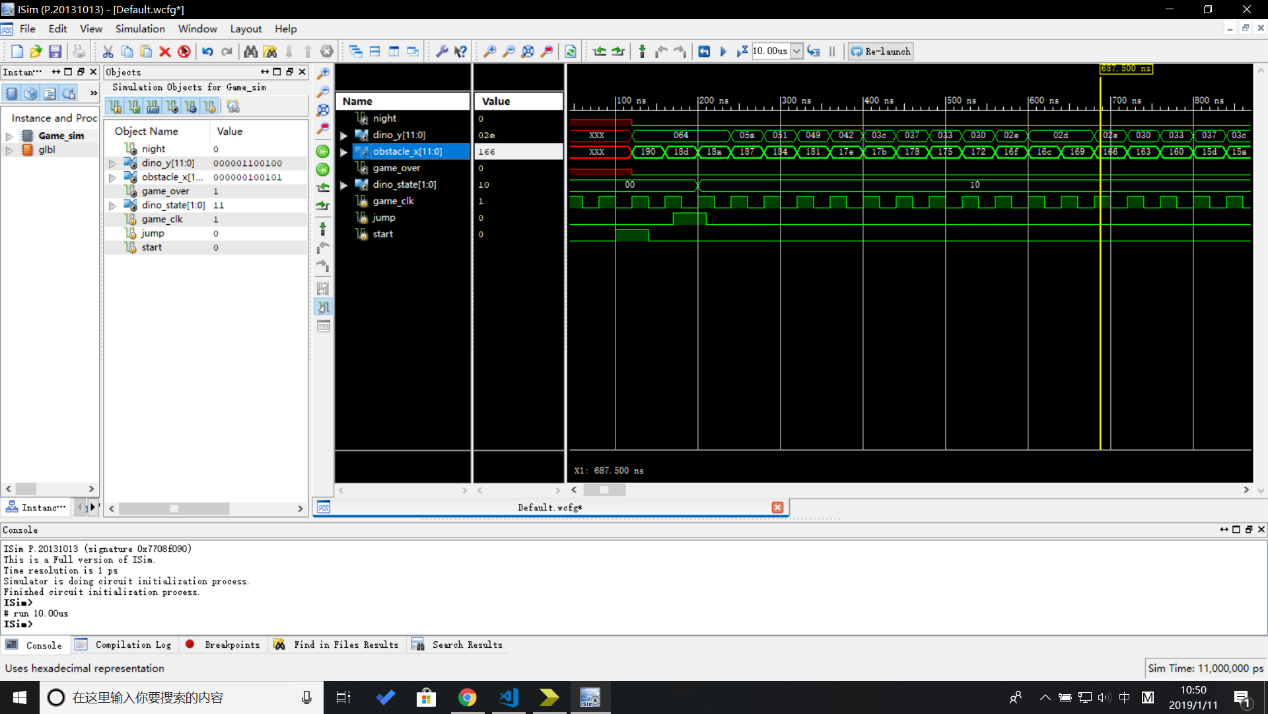
**(4) 在小恐龙碰到仙人掌后，计数器停止计数，小恐龙位图变化，游戏结束**



（游戏结束时小恐龙位图）

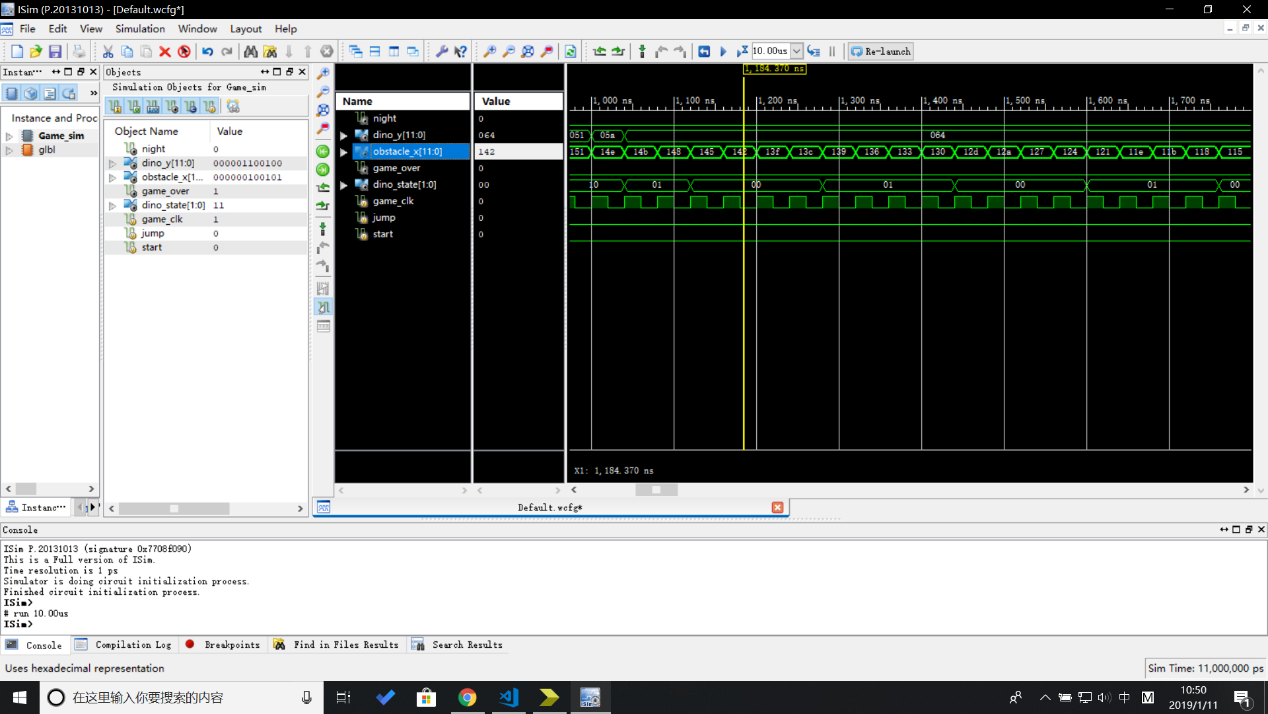
# 五、核心模块模拟仿真时序图

## Game模块：开始及跳跃



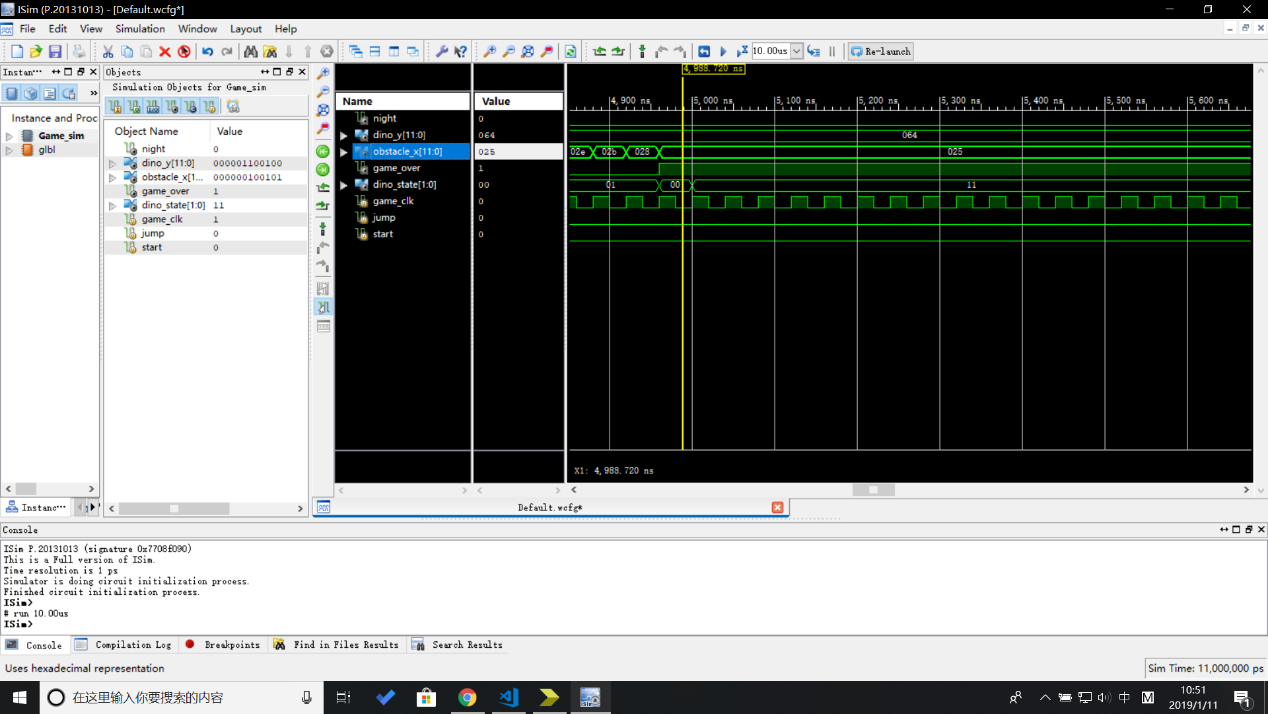
可以看到game模块的输出obstacle\_x再开始后每一个游戏时钟周期减1，开始跳跃后输出dino\_y以抛物线的方式变化。 Game模块开始及跳跃

## Game模块：跳跃结束



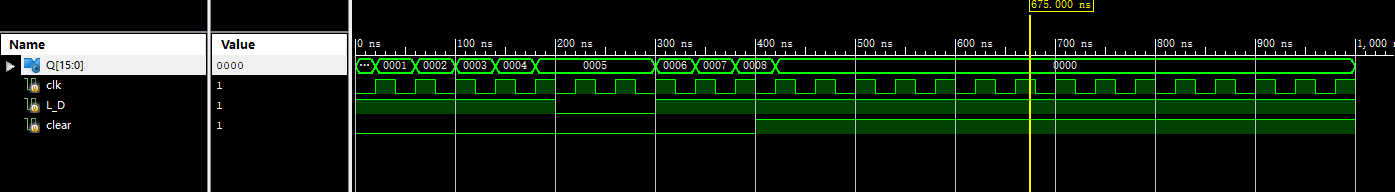
当小恐龙落回地面后，dino\_y停止变化，结束跳跃状态，可以注意到小恐龙在水平移动时dino\_state不断切换。 Game模块跳跃结束

## Game模块：游戏失败



发生碰撞后游戏停止，输出game\_over为1，表明游戏失败。

## Score模块



当 L\_D=1且clear=0 时代表游戏进行，此时计数器正常计数；L\_D=0时代表游戏结束，此时计数器停止计时；clear=1时，表明游戏开始，此时计数器清零。

# 五、组内成员分工说明及贡献比例

## 成员分工

资料查阅：钟嘉农

程序代码：包文鼎、钟嘉农、邬树辉

报告背景介绍、设计说明：邬树辉

程序调试及分析：包文鼎

模块仿真模拟及分析：包文鼎、邬树辉

执行过程及视频拍摄及后期制作：邬树辉、钟嘉农

## 贡献比例