**计算机逻辑设计基础课程设计**

**小游戏Flappy Bird设计报告**

**Group：费姚瑞（3220103113），仇国智（）**

**Date：2023.12.22**

1. **背景介绍**

Flappy bird是一款2D横版闯关游戏，画面中一只小鸟始终向右飞行，玩家可以通过一直点击来让小鸟不断向上飞行，一旦停止点击小鸟就会开始下落。这个游戏要求玩家利用这一机制使小鸟保持适当的高度来通过不同高度的障碍物。

在本次课程设计中，我们小组将利用Verilog HDL语言和相关实验环境实现这一小游戏的运行和游玩。

1. **设计说明**
2. **设计开发环境**

实验平台：Sword Kintex7

开发环境：Xilinx ISE、Xilinx Vivado

硬件描述语言：Verilog HDL

1. **输入输出交互选择**

输入：Sword平台上clk时钟信号、BTN按钮及SW[15]开关

输出：Sword平台上VGA显示器以及4位7段数码管

1. **控制模块设计——Control Module**

实现Flappy Bird这一小游戏的核心模块就是控制模块，即读入玩家的操作、将其反馈到游戏中并结算游戏状态。

我们对这一小游戏的交互想法很简单，玩家的输入只需要两个：一是控制小鸟向上飞的BTN按钮，另一是控制游戏开始以及重新开始的SW[15]开关。因此读入玩家的操作只需要两个一位的控制信号（up、rst）即可储存。

而对于游戏某一时刻的状态确定，我们需要以下信息：

1. clk：Sword平台提供的clk时钟信号
2. fail: 游戏是否处于游玩状态，在拨动开始开关后游戏便开始游玩状态，倘若小鸟撞到了障碍物上，则游戏失败，退出游玩状态。
3. rst：控制游戏启动及重启，倘若rst为1则将游戏状态清0。
4. up：记录玩家是否按下了控制小鸟飞起的按钮，若为1则小鸟飞起
5. score：记录本局游戏至此的分数，小鸟每成功穿过一个障碍物则加一分，初始为0分
6. bird\_y：记录当前状态下小鸟的高度位置（以小鸟左下角为基准点）
7. pipe：记录当前障碍物（管子）的横坐标及高度（以管子左上角为基准点），屏幕中同时只出现三个障碍物。

而对于控制模块的实现思路如下：

1. 初始状态下或rst重置后在屏幕右侧生成三个pipe，高度由固定范围随机数生成，横坐标固定；将小鸟生成在屏幕左侧中央。
2. 只要游戏没有结束，小鸟就向右飞行，而在游戏实现中则是小鸟横坐标不变，而pipe不断向右移动；一旦pipe横坐标小于0则将其初始化到最右边并重置随机高度。
3. 一旦玩家按下BTN按钮，则使up为1，代表小鸟接下来将向上飞行。小鸟向上飞时bird\_y增加，若下落时则bird\_y减少
4. 由于游戏的流畅性及真实性考虑，小鸟应当在按下BTN按钮后向上飞行数秒再开始下落，因此需要变量bird\_falling标记小鸟距离开始下落的周期数，当bird\_falling为0时小鸟下落，而当bird\_falling大于0时，此周期小鸟向上飞并使bird\_falling减一。

因此bird\_falling初始为0，当按下BTN按钮时置bird\_falling为3，代表小鸟将飞行三个周期。

1. 一旦小鸟撞到了障碍物，此时置fail为1，代表游戏失败结束，此时pipe将不会移动，并且小鸟将无视bird\_falling的值和玩家操作直接下落直至掉出屏幕
2. 为提高游戏真实性，我们需要检测这一周期小鸟是在下落还是在上升，若下落则显示小鸟俯冲的贴图，若上升则显示小鸟扑腾翅膀向上飞的贴图，因此将bird\_y的最高位作为符号位记录小鸟当前状态，符号位为1时小鸟向上飞
3. 倘若游戏还在进行并且小鸟顺利通过了一个障碍物，则使score加一，表示分数加一。

为了保证游戏的合理性，此模块中还需要添加pbdebounce模块用于按钮防抖、clk\_div模块分频生成1ms频率供防抖，以及clk\_100ms模块用于调整游戏速度，若要修改游戏运行速度则修改其中参数即可。

1. **显示模块设计——Display Module**

实现这一小游戏的另一重要模块便是显示模块，这一模块需要接收control模块运算得到的游戏状态并将其通过VGA可视化输出