实验报告

实验题目: lab11 – 大作业 日期: 2018 年 12 月 14 日 **我的压缩包:**

我是按照助教在群里的提示,用下列设置对代码进行压缩的:

| Create a compressed (.zip) file that contains all the sources, settings and other files associated with this project. | |
|---|---|
| Archive name: | Game |
| Archive <u>l</u> ocation: | C:/Users/fandahao/Desktop |
| Archive file will be | created at: C:/Users/fandahao/Desktop/Game.xpr.zip |
| Temporary location: | sers/fandahao/Desktop/Game/.Xil/Vivado-6008-DESKTOP-4ECNA72 💿 🕝 |
| ✓ Include configura | tion settings |
| ☑ I <u>n</u> clude run resul | ts |
| ✓ Include run resul | ts |

设计目标:

我希望实现一个去年此时火遍全国的"跳一跳"小游戏,并在此基础上进行创新:除去经典的通过玩家按键时间确定跳跃距离外,我还希望利用开发板上的 ADXL362 加速度传感器,增加利用开发板的倾斜来改变跳起方向,从而改变跳跃距离的玩法,最终的实现效果如压缩包中视频"重力感应演示"所示。

实现方式:

为了实现我的设计目标,我需要解决以下困难:

- 1. 在 VGA 显示器上实现"跳跃"的效果。
- 2. 精确记录玩家的按键时间,并将其转化为跳跃距离。
- 3. 实现 FPGA 与加速度传感器间的通信,获取加速度传感器的数据。

我解决上述问题的方法是: 首先,利用网络资源,获取 VGA 的驱动程

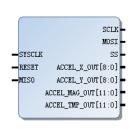
序(我使用的是从老师处借来的 800x600 显示器)。我将"跳跃"的轨迹 建模为一个抛物线,由于跳跃的起始点和跳跃距离已知,只要运用**抛物线 的两点式**:

$$y = A(x - x_1)(x - x_2)$$

即可得到物点的坐标,根据实验的实际情况,我选择 A = -1/128,这在硬件中是比较容易实现的,因为除以 128 的操作相当于将数字右移 7 位;

记录玩家的按键时间比较容易,我使用了一个**计数器**进行计时,在玩家按键时开始计时,在松开键时结束计时;

加速度传感器的使用比较困难一些,经过搜索,我找到了 Nexys4-DDR <u>开机程序的源代码</u>,其中涉及到加速度传感器的部分,可用用来获取板子的方向,但这段代码使用的是 VHDL 语言,不能直接在项目中使用,于是



在我的项目中**实例化该 IP**(如左图所示),获得了加速度传感器的数据。 我需要的是 X 方向(竖直方向)的输

我根据 Xilinx 官方教程利用这段代码自主生成了 IP,

出数据,我将其按角度不同分为16个档,分别对应0到

tan-1 4之间的 16 个不同的跳跃角度和距离,这对于本游戏已经足够。

除此之外,为了增强游戏体验,每两个墩之间的距离都是**随机生成**的, 该伪随机数是通过将当前数值不断异或产生的。

游戏逻辑:

程序的结构如下图所示:

```
· ctrl - Control (Control. v)

    ai - AccInput (AccInput. v)

    @ gi − GameInput (GameInput. v)

 -- sc - ShowScore (ShowScore.v)
..... r1 - Random (Random. v)
± 4 acc - AccelerometerCtl2_2 (AccelerometerCtl2_
```

Clock 模块:产生不同频率的时钟以供其他模块使用。

Control 模块:负责游戏的控制,包括游戏中、主角死亡、重新开始等。

AccInput模块:处理由AcclerometerCtl模块传入的开发板倾斜程度数据。

Game Input 模块:负责记录经典模式下玩家的按键时长。

ShowScore 模块:记录玩家的当前得分和最高分并在七段数码管上显示。

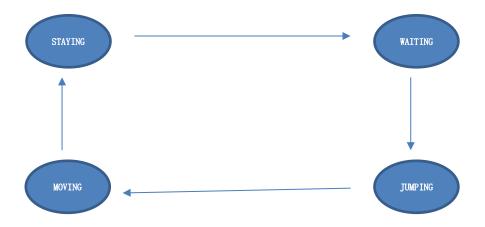
Man 模块:游戏主角的控制模块,负责处理主角的跳跃行为及其状态的切换。

Platform 模块: 控制跳跃的平台的产生和移动。

AcclerometerCt1 模块: 利用 VHDL 代码生成的加速度传感器驱动模块。

VGA_COLOR 模块:负责将图形在 VGA 显示器上显示。

其中, Man 模块是游戏的核心控制模块, 其状态图如下所示:



其中,STAYING 状态下,机器在等待玩家的输入,WAITING 状态只在经典模式下有效,是在玩家按下控制键而尚未松开时的状态,JUMPING 状态为跳跃状态,跳跃结束后将根据主角位置判断其是否死亡,若未死亡则进入MOVING 状态,游戏主角与墩一起向后移动。

评估:

我分别利用了按键时间长短(**视频"按键感应游戏效果"**)和开发板倾斜角度(**视频"重力感应游戏效果"**)进行了游戏。游戏结果实现了我的设计目标,且利用重力加速度传感器的新玩法游戏体验更佳。

总结:

- 本次实验的不足之处在于由于时间紧迫,只实现了游戏的功能,没有设计出一个好看的图形界面,导致游戏的卖相很丑,以后要加以改进。
- 2. 本次实验,我在设计 Man 模块时遇到了瓶颈,由于起初状态机的设计不合理,导致出现了多次仿真正确而下载错误的情况,最终我重写了整个模块,采用了规范的两段式写法,最终成功实现功能。这说明在硬件编程时一定要特别注意代码的规范性。