Derpy Bird Team 21 Report

Tony Huang

Department of CSE

University of Auckland

qhua835@aucklanduni.ac.nz

Auckland, NZ

Harsh Thorat

Department of CSE

University of Auckland

htho884@aucklanduni.ac.nz

Auckland, NZ

Lojanan SivanantharubanDepartment of CSE

University of Auckland

lsiv157@aucklanduni.ac.nz

Auckland, NZ

**Abstract - This project intends to provide a gaming console experience similar to a Gameboy. Designing a side scroller flappy bird game using a FPGA board (Cyclone V 5CEBA4F23C7) with the knowledge learnt from the course COMPSYS305. The goal of the game is to achieve the highest score possible by flapping the bird through the gap opening. In our game we provide unique features such as changing background by pushing a button on the FPGA board. Although the basic requirements of the game are reached, future improvements such as higher max frequency, smoother game play, display text with MIF, more power ups can be implemented.**

***Key words - FPGA, VGA, VHDL, FSM, Switches, Push Buttons, Flappy bird.***

**Introduction**

The goal of this project is to design a custom flappy bird themed game using digital logic and VHDL to program a FPGA board.

The rules of the flappy bird game are to keep the bird floating while avoiding obstacles such as pipes and the ground. Since the project is not finished there are more features to come such as extra score, lives.

**Game Features**

The flappy bird game is an endless game with three lives. Like Jetpack Joyride. Player will need to “flap” the birds through the opening of the gaps without the bird touching the pipe to score.

The game consists of two stages, there is training and normal mode. In training mode, the speed of the game will not increase and able to pick up power ups. When the player touches the pipe, they will lose one life and when the player touches the ground, they will instantly die. In normal mode, similar to training mode. The game speed will increase every 10 points that have been accumulated. Capped out at 20 points.

The game consists of two types of power ups, extra life, and extra points. Extra life will be a heart looking object. When the player touches the heart, life will increase by one. Max life is capped at three. The second power up will be a coin looking object, when the player touches the coin, the score will increment by one until the coin disappears.

**Setup and game tutorial**

**Equipment needed:** Cyclone V 5CEBA4F23C7, PS2 mouse, Cables, Screen.

Plug everything together, connect the HDMI cable to the back of any monitor. Press the red power on button.

Player press KEY2 to start the game.

**Key Binds:**

KEY[0]: Change background

KEY[1]: Pause game

KEY[2]: Start game

KEY[3]: Reset game

SW9: up is normal, down is training mode.

**Rules**

If the bird touches the ground, the bird will die. When the player touches the pipe bird will lose one single life.

**Design and Implementation**

|  |
| --- |
| Fig 3 – Block diagram |

Our design consists of a display control unit, a FSM, and several components, such as the bird, pipes, ground…. The state logic is handle by the FSM and individual logic is handled between the components.

|  |
| --- |
| Fig 1 – FSM game state |
| State diagram for sprite animation  Fig 2 – FSM bird animation |

**Minji Kim**

**FSM game state (Fig 1):** Our Moore FSM consists of four different states which are “HOME”, “START”, “Pause”, “GAME\_END”. HOME refers to the starting menu of when the game is first initialized or after each reset. Depending on the input each state receives from the user input, it will change the states and the output to the correct signal to change the game settings.

**FSM bird animation (Fig 2):**  Our Mealy FSM for bird’s wing flap animation consists of three states, “Sprite1”, “Sprite2”, “Sprite3” these refers to the wing location of the bird. They change states every 0.5 seconds.

|  |
| --- |
| Fig 4 - FSM |

**Yeji Hwang**

**Input & FSM:** This circle is responsible for the input from the physical buttons and switches on the FPGA board and the mouse. It gives the inputs to the FSM, thus changing the game states. E.g. when “KEY[2]” is pressed the game pauses.

|  |
| --- |
| Fig 5 - Bouncy |

**Bouncy Ball:** This block represents our derpy bird. It contains the logic of the bird, such as the vertical flapping motion, collision detection, score counting. It takes in the state information from FSM, it also passes data from and to pipes to do the logic calculations.

|  |
| --- |
| Fig 6 - Pipes |

**Pipes/Power ups:** This block represents our pipes and power ups. Pipes move from the right side of the screen to the left side of the screen, as the game goes on the speed of the pipes will increase. Pipes gap position is also randomized by LFSR. Power ups contain bonus points and extra lives. Similar to bouncy ball it also takes in states from FSM.

|  |
| --- |
| Fig 7 - background |

**Background:** This block represents our background, which we have two types of backgroun. One of them is daytime and the other one is nighttime. Pressing KEY[0]/PB1 will change them.

|  |
| --- |
| **Fig 8 - text** |

**Text Display:** This block represents our text display, which we use to display all of the text in the game. It takes in the input from FSM, depends on which state it is on; corresponding text will be displayed.

|  |
| --- |
|  |
| Fig 9 – Display Controller |

**Disp\_controller:** This block represents our display controller; the display controller decides which component to display first. It takes in 12-bit color input from the game components and passes the 12-bit RGB value to VGA Sync.

**Decision & Trade-off**

At first we did not use

Optimization

Initially in our design, we did not consider implementing our games using FSM. It was pure spaghetti code. Everything was very messy, and it was a pain to manage all the different signals. There were lines going everywhere.

|  |
| --- |
| Fig 5 – Bad optimization |
| Fig 6 – Better optimization |

With reference to Fig 5. & Fig 6, both codes serve the same logic. However, Fig 5 uses a lot more registers compared to Fig 6. Initially Fig 5 is pure spaghetti code and Fig 6 we have implemented FSM. This drastically reduces the number of resources used.

|  |
| --- |
| Fig 7 – Timing Analysis |

**Compilation**

|  |
| --- |
| Fig 4 – Compilation Report (Yet to come) |

We used A PLL block to divide the board clock from 50 MHz to 25 MHz to make video timing more compatible. As the VGA display is 640x480, we need to run the display at least 60 Hz. Since there is 800 pixels per line and 525 lines at 60 Hz. This would roughly equate to 25 Mhz.

**Conclusion**

In conclusion, this project aims to familiarize students with the FPGA board making a similar game to flappy bird using digital logic and digital concepts such as FSM.