COMPSYS305 Final Report

Carrie Ng, David Huang, Jesse Zeng
University of Auckland
Auckland, New Zealand
cng822@auckanduni.ac.nz,
zhua687@aucklanduni.ac.nz,
jzen349@aucklanduni.ac.nz

***Abstract*—The project was to implement a game similar to Flappy Bird with a training and game mode. The game mode requires 3 levels of difficulty. The player would need to interact with the game by controlling the bird and ensure it does not lose all its health points. The interaction can be done with switches and buttons on the DE0 board as well as its mouse inputs. A high-level finite state machine is required to define the states and control the overall operations of the system. The game is organized in blocks which with each other's I/O. The game required < 1% of its total memory bits. For future improvements, with the remaining number of memory bits available, the game's design or bird could be improved by adding rendered images.**

Keywords— DE0 board, Flappy Bird, VHDL, game, mode

I. INTRODUCTION

For this mini-project, our group is tasked to create a game similar to Flappy Bird in VHDL using ModelSim and Quartus II on Windows 10. The game is implemented on a DE0 board. There are two modes - training mode and game mode. The game mode, in particular, consists of 3 levels of difficulty, which has been implemented to make it challenging for the player as the game progresses. The player must score 55 in order to complete the game.

II. GAME DESIGN

A. Game Overview

The main objective of the game is to successfully pass all levels and pipes by scoring 55 in the game mode without losing all the player's health points. There is also a training mode for the player to practice in.

Training Mode - an unlimited number of pipes is generated and the speed of the game will not increase as the player's score increases. Any features that occur in the game mode (e.g invisibility, larger size bird) will not appear in this mode. This mode ends when the player returns back to the main screen.

Game Mode - 3 levels of difficulty are implemented in this mode and they all have different features that distinguish the difficulty of the level. As the player's score increases, the horizontal speed of the pipes increases also. The player's bird must achieve certain scores to be able to succeed to the next level.

For each pipe the player successfully passes, their current score will be increased by one. The player will have shields in Level 1, a bigger sized bird at Level 2, and the game will become difficult at Level 3 when the bird becomes invisible after every 5th score whilst flying and will only reappear once the player successfully gains a score. The player would also gain 1 HP every 5th score they achieve.

The player loses points when the bird touches the pipes as it flies through. The game will end when the player no longer has any health points remaining or if the bird touches the top or bottom of the screen before they successfully achieve 55 points.

B. User Interfaces

The game will interact with the player with mainly the following interfaces:

- DIP switch (SW0)
- Push Button 2 (PB2)
- PS/2 Mouse
- VGA display
- Seven-Seg display

The game will be connected to a VGA monitor at 640x480 resolution. When connected, a menu screen will be displayed where the player can select either the training mode or game mode.

The player can select this by using the mouse's right-click or left-click to indicate the respective game mode

they wish to play. The mouse will become the player's main controller in the game and left-clicking on the mouse will change the bird's vertical direction while right-clicking when the game is over will restart the game. Left-click and right-click is also used to select which game mode the player wants to play.

At any time of the game, if the player wishes to stop playing, PB2 on the DE0 board can be pressed and it will reset the game and return the player back to the selection screen. The player can pause the game by moving the SW0 to '1'. Otherwise, the game will continue until the bird loses all its health points. At the end of the game, the game over screen will be displayed and the player can left-click on the mouse to return back to the main menu.

If the player wishes to know their score on the DE0 board, the four seven-seg display on the board will be able to tell them their health and score of the current game. Otherwise, the VGA screen will also display their score and health.

III. IMPLEMENTATION

The game was implemented and organized in separate blocks and components. For the game to run smoothly, these blocks interact with one another via their inputs and outputs. We also implemented a restart functionality in the game where the player can right click after the end of the game. This will bring them back to the start of the game and they can start playing without having to go to the menu.

A. Implementation Changes

We had made some changes in regards to the inputs used in the game. We realized that the pause button would become inconvenient to the player if it was set as a push button. The player would either have to constantly hold onto the button until they wish to resume or we would need to make a component which stores the pause button. For the player's convenience we decided to use a DIP switch (SW0) instead which can pause the game without the player needing to hold onto the button. The player can easily flip the switch back.

Another change we made was making the selection inputs for the game modes in the menu screen just left click and right click inputs. We decided to use left click and right click rather than the DIP switch as it will be convenient for the player as their hand would be already on the controller (mouse). For this reason, we can make an additional feature of the game where the player can restart the game by right clicking. by right-clicking, the mode changes from 'game over' to 'main menu' to 'gamemode' very quickly causing an illusion for the player to think the game was restarted but was in fact due to rapid changes in mode.

.

B. Components

- Mouse - The player's input. Required for the control of the bird.
- Linear Feedback Shift Register (LSFR) - A Pseudorandom number generator which selects the positions of the upcoming pipes
- Char_rom - Stores the characters into the game's memory
- PLL - Clock divider which divides the clock signal from 50MHz to 25MHz as that is the required clock signal for the DE0 board.
- In-game Stats - Determines game statistics including score and health
- Bird - Controls the size and movement of the bird
- Pipe Gen - Generates the pipes and controls its movement
- FSM - Indicates which state the game should transition to
- Display Selector - Determines the display positions, priority and colours of pixels

C. Block Diagram

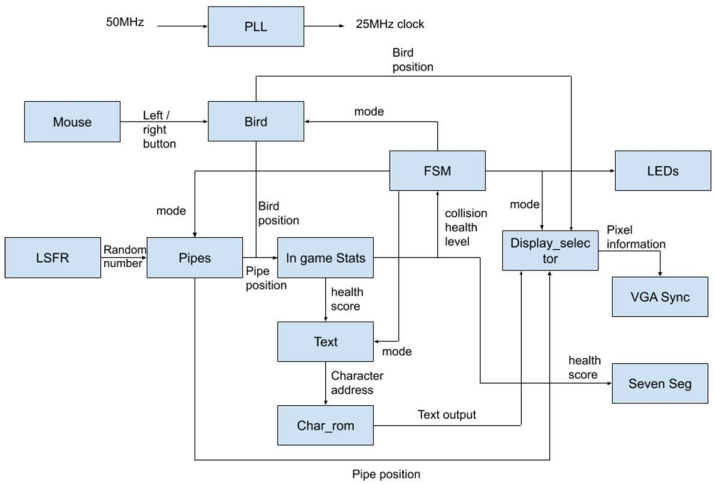

Fig 1. Block Diagram

D. FSM

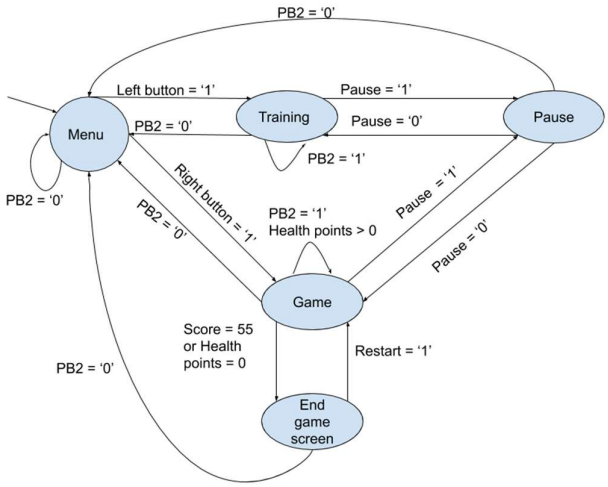

Fig 2. High Level Finite State Machine

The game has implemented a Mealy FSM where the output is dependent on the current input and state. The FSM has 5 main states. These states are "menu", "training", "game", "pause", "end game screen" and they are used to transition the game's behaviour. FSM will be responsible for how the game transitions from one state to another such as the start of the game, the next state will be determined depending on what user inputs it receives from the DE0 board. Each state will have different user inputs conditions and is dependent on its previous state in the system. At any point in the game if PB2 is 0, the state will return to 'menu'.

IV. Design Decisions and Trade Offs

As previously stated in the interim report, the initial idea was to have a medkit appear on the screen where the bird can pick up whilst flying through. However, due to time constraints and complications, this was replaced with implementing an HP increase after every 5th score.

There were some changes with the implementation of the invisibility aspect in the third level of the game. Rather than making the pipes invisible in the game periodically, we decided to make the bird invisible through making the background colour the same as the bird instead. Making the bird invisible meant the game implementation would become easier. For the pipes to become invisible every x amount of seconds a clock divider needs to be implemented in the game. With the bird becoming invisible, there is no need for a clock divider and helps with the time constraints.

Similar to changing the invisibility from pipes to the bird, the other change was to make the bird go invisible after every 5th score instead of every 5th second. Having our pipes become invisible will also require another clock and more components which may become time consuming and complicated. Therefore, if the invisibility is dependent on the score, fewer components will be used.

V. Performance

A. Resource Usage

The game uses 1443 of the total logic elements available which is about 9% of its total available logic elements. It also uses 257 of the total registers and 50/347 (14%) of the pins on the DE0 board are used for this game. The program also uses 4096 total memory bits which is less than 1% of its total available memory.

Flow Status	Successful - Sat Jun 06 21:20:19 2020
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	ExampleCodes
Top-level Entity Name	interim
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	1,443 / 15,408 (9 %)
Total combinational functions	1,409 / 15,408 (9 %)
Dedicated logic registers	257 / 15,408 (2 %)
Total registers	257
Total pins	50 / 347 (14 %)
Total virtual pins	0
Total memory bits	4,096 / 516,096 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	1 / 4 (25 %)

Fig 3. Compilation report

B. Max Operating Frequency

The TimeQuest timing analyser report for the game at Slow 1200mV 85C model shown below displays few of the components having a higher frequency than 500MHz. Therefore, their Fmax frequency has been limited to 500MHz. The maximum operating frequency for this implementation is 64.39 MHz.

	Fmax	Restricted Fmax	Clock Name
1	64.39 MHz	64.39 MHz	clockdivider|altpll_compo...uto_generated|pll1|clk[0]
2	146.54 MHz	146.54 MHz	VGA_SYNC:display|vert_sync_out
3	320.31 MHz	320.31 MHz	MOUSE:mouse|MOUSE_CLK_FILTER
4	344.59 MHz	344.59 MHz	InGame_stats:game_stats|health_up
5	390.78 MHz	390.78 MHz	InGame_stats:game_stats|collision
6	809.72 MHz	500.0 MHz	pipe_master:pipe_control|reset_3
7	821.02 MHz	500.0 MHz	pipe_master:pipe_control|reset_2
8	851.06 MHz	500.0 MHz	pipe_master:pipe_control|reset_1

Fig 4. TimeQuest Timing analyser report on Quartus

VI. Conclusion

This project was to create a Flappy Bird clone game in VHDL and implemented on a DE0 board. The game requires two modes: a training mode and a game mode. Our game-mode needed to be implemented with three levels of difficulties with unique speeds and special features that enhance the levels. The game was

designed for the user to practice and be able for the user to learn how to complete the game.

In terms of future improvements, there are many aspects or features we could improve or implement in the game. One improvement we wish to implement in the game is the quality of the bird. Currently, our bird is just a yellow pixel. By implementing an improved version of the bird or adding images and render them, this feature would enhance our game design and quality. If possible, in the future we would also like to implement additional features or complex levels (e.g involving math equations) to gain scores which makes the game more interesting for the player.