Hero of Konoha

COMPSYS 305 Mini-Project

Matt Eden, Sherjeel Shehzad
ECE Department
Auckland, New Zealand
[mede607, sshe325] @aucklanduni.ac.nz

***Abstract*—A Flappy Bird clone implemented on a DE0 board, written in VHDL. Supports a menu, two game modes and a game over screen. Has 3 levels of difficulty.**

***Index Terms*—FPGA, VHDL, Flappy Bird, DE0, PS/2**

I. Introduction

As a mini-project to further develop skills associated with digital systems design, this team was tasked with creating a Flappy Bird clone in VHDL and implementing it on an FPGA. In order to effectively complete this task, it was necessary to consider the behaviour of a wide range of a systems and manage the interactions between them in order to create a smooth user experience. The premise of our game involves a ninja throwing a kunai through a forest to defeat enemies. The kunai must pass through gaps in the trees, so that "Konoha" can be saved.

II. Game Strategy

The objective of the game is to achieve the highest score possible, which in essence means that the player needs to try and keep their kunai flying for as long as possible. A player's score is increased with each gap in the tress that the kunai flies through, so precise timings are required to achieve a high score. The player will first be presented with a choice between two modes, "Training" and "Story". "Training" will allow the player to practise and get familiar with the game mechanics. "Chakra" is not used in "Training", so the this mode will end once the kunai hits a tree. "Story" introduces the concept of "Chakra" and allows the player to obtain a score. Once a mode is selected, the player then needs to click the left mouse button in sucession to keep the kunai elevated and time their clicks such that it passes through gaps in the trees. For every 3 trees passed, "Chakra" is recovered. Each time the kunai hits a tree, some "Chakra" is lost. "Story" also contains 3 levels of difficulty, with level increasing the game speed to provide more challenge to the player.

For every three trees passed, a player's "Chakra" will be increased. Each collision with a tree drains the player's chakra by a small amount, so it is important not to hit the trees, as once a player's charkra runs out, the game is over. There are 3 levels of difficulty, with each one unlocking after the player passes a certain number of pipes. Each level increases the speed of the kunai and the trees, so the player needs to be on high alert if they wish to keep their score up.

A. How To Play

On the menu screen, the player should use the right-most DIP switch to select their preferred game mode. When the DIP swtich is up, this will select "Story". Otherwise, the selection is "Training". They should then use PB1 to confirm this selection. Once this has been done, the game begins and the player will need to use the left mouse button to control the elevation of the kunai. At any point, PB2 can be used as a reset to return to the menu screen. If or when the player loses and they enter the game over screen, PB0 can then be used to exit that screen. Additionally, at any time during either of the two modes the game can be paused by setting SW1 to high.

B. Interface Description

The player will interact with the game via a combination of switches, buttons and mouse input.

The game itself will be displayed on a monitor with using a 640x480 resolution, connected with a VGA display cable using 4-bit colour. The player will select their game mode using a DIP switch, then confirm their selection with one of the three push buttons available. Playing the game will involve clicking the left mouse button on the PS/2 mouse to elevate the kunai on screen. At any time, the game can be reset back to the menu via presing one of the push buttons.

III. Implementation

The game is implemented using VHDL on an Altera FPGA Development Kit, otherwise known as a DE0 board. It was compiled using a 64-bit version of Quartus 2 on Windows 10.

A. Overview

The game is organised into separate and distinct blocks, which interact with each other via their I/O to create the final game. Each state of the finite state machine is given its own block, as well as there also being blocks for ROM.

- display_sel
  This is a 4-to-1 34-bit MUX, which selects the render signals to send to the VGA based on the state of the FSM. All four VGA_sync components for all four game states are run at once, and the output from the VGA_sync associated with the current state is sent to the output.
- ball_menu
  This represents the "Menu" state. It uses the character rom to display the game title and information on how to select the two game modes.
- ball
  This represents the "Story" state, otherwise referred to as the "normal" state. It implements the regular game mode, with "chakra" consumption and level progression.
- ball_training
  This represents the "Training" state. This does not have any "chakra" consumption or level progression, but the game ends when the kunai collides with any tree.
- ball_over
  This represents the "Game Over" state. It displays the score achieved while playing either of the two game modes.
- LSFR
  This generates pseudo-random numbers based on an initial seed which are used to vary the gaps in the pipes.
- PLL
  This divides the clock from 50MHz to 25MHz, as this was a requirement to work with the DE0 board.
- char_rom
  Reads the MIF and stores the character set for the game in memory.
- background_rom
  Reads the MIF and stores the background for the game in memory.

B. High-Level System Block Diagram

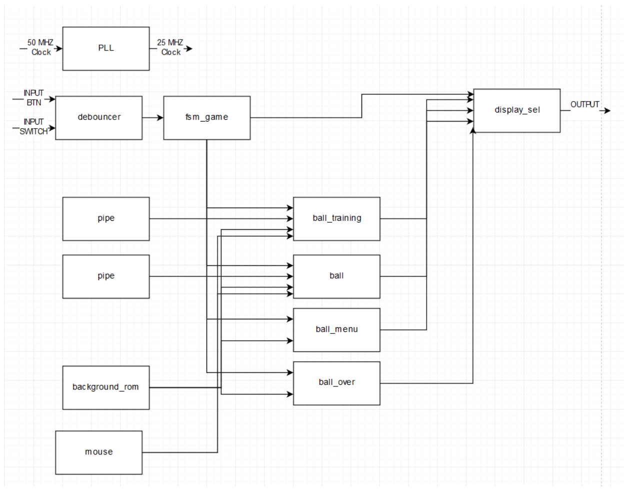


Fig. 1. The High Level System Block diagram

C. FSM Description

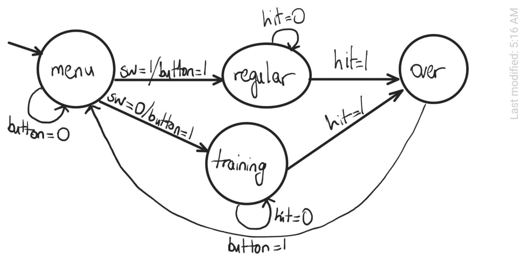


Fig. 2. An overview of the Finite State Machine

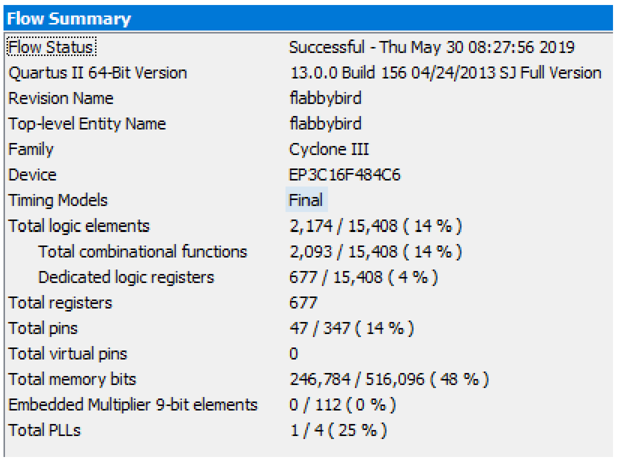
There are five states we are concerned with: “Menu”,“Story”, “Training”,“Over” and “Pause”. The system begins in “Menu”, where the next state to move to depends on the value of the switch ’sw’ and the value of the button. Either “Story” or “Training” will be the next state. The system will persist in this state until ’hit’ becomes 1. At which point, the system moves to the “Over” state. At any point, if ’reset’ is 1, then the system resets to “Menu”.

IV. Results

At completion of the project, there are various aspects that are of interest to reflect on, such as the performance and the design decisions made.

A. *Resource Consumption*

Our game makes use of 14% of the available logic elements, 677 total registers, uses 47/347 of available pins and takes up 48% of the total memory bits. A single PLL is used, despite the four available on the DE0 board.



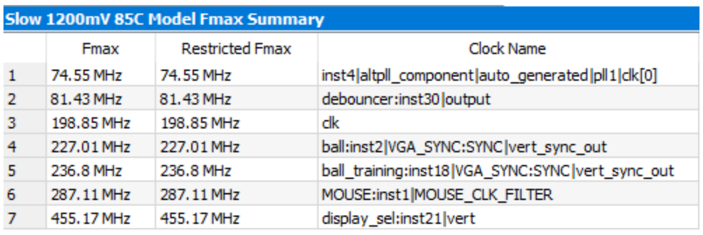
Flow Summary

Flow Status	Successful - Thu May 30 08:27:56 2019
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Full Version
Revision Name	flabbybird
Top-level Entity Name	flabbybird
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	2,174 / 15,408 (14 %)
Total combinational functions	2,093 / 15,408 (14 %)
Dedicated logic registers	677 / 15,408 (4 %)
Total registers	677
Total pins	47 / 347 (14 %)
Total virtual pins	0
Total memory bits	246,784 / 516,096 (48 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	1 / 4 (25 %)

Fig. 3. The compilation report

B. *Timing Analyses*

The maximum operating frequency of this implementation is 74.55 MHz. As can be seen on Fig 3, the bottleneck for the implementation is the debouncer at 81.43 MHz. However, this is acceptable as the maximum frequency used in the design is 50MHz, which is much lower than our maximum operating frequency.



Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name
1	74.55 MHz	74.55 MHz	inst4|altpll_component|auto_generated|pll1|clk[0]
2	81.43 MHz	81.43 MHz	debouncer:inst30|output
3	198.85 MHz	198.85 MHz	clk
4	227.01 MHz	227.01 MHz	ball:inst2|VGA_SYNC:SYNC|vert_sync_out
5	236.8 MHz	236.8 MHz	ball_training:inst18|VGA_SYNC:SYNC|vert_sync_out
6	287.11 MHz	287.11 MHz	MOUSE:inst1|MOUSE_CLK_FILTER
7	455.17 MHz	455.17 MHz	display_sel:inst21|vert

Fig. 4. The TimeQuest timing analyser timing report

C. *Design Decisions*

Due to memory constraints, we were only able to have one background and that background could only be 160x120. While we investigated having multiple backgrounds for different game states, we discovered that there was insufficient memory on the DE0 board available to achieve this.

V. Conclusion and Future Work

This project’s goal was to create a version of “Flappy Bird” in VHDL that, while not replicating the original in its entirety, still needed to be recognisable as a parody of the original. The implementation was on a FPGA, programmed with VHDL. Menu elements containing relevant information such as a player’s score are displayed on a VGA-compliant monitor, with the target resolution being 640x480 and using 4-bit colour. The final game has 3 levels, with each increasing the difficulty as the player progresses through the game. The design of the game was such to allow some challenge but also to be accessible to a wide audience.

In regards to future improvement, we believe that adding music and sound effects would greatly enhance the experience of playing the game. Additionally, if a board with more available memory could be acquired then higher-resolution assets could be implemented. Futhermore, randomly generated items would be a desired gameplay addition as this would add another dimension of play.

VI. Appendix

The ASM for the vga_sync block will go here.

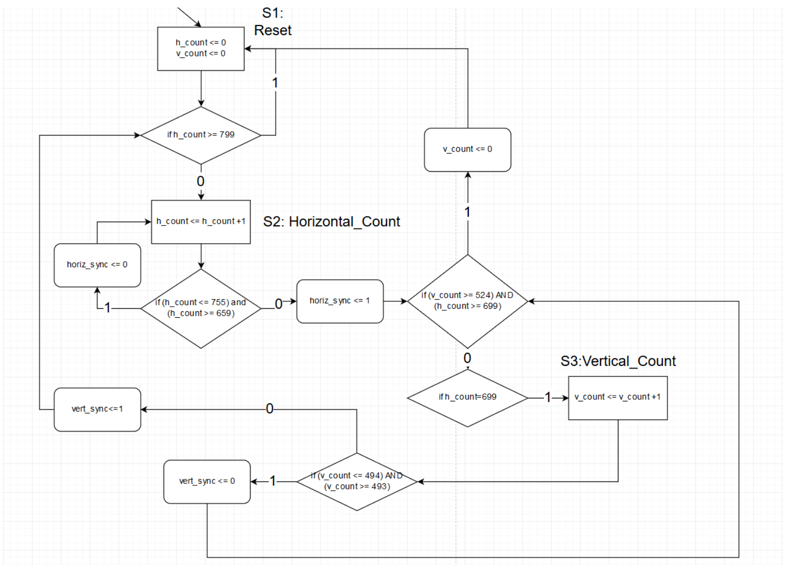


Fig. 5. An ASM for the VGA Sync component.

Acknowledgement

This team gives its thanks to Dr. Morteza Biglari-Abhari, Dr Muhammad Nadeem, Krystine Sherwin, Marijam Hemmati and Nick Harvey, as well as the University of Auckland's Electrical and Computer Engineering department for providing us with laboratory space and necesssary components.

References

[1] B. Singh, Background Artist.
[2] M. Eden, Background Model.
[3] M. Kishimoto, owns intellectual property rights to the character "Naruto Uzumaki" from the "Naruto" franchise, which was used as the inspiration for the background.
[4] Altera, DE0 User Manual (http://esca.korea.ac.kr/teaching/FPGA_boards/DE0/DE0_User_Manual.pdf).
[5] M. Nadeem, COMPSYS 305 course notes.
[6] Image resize tool (https://picresize.com/).