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 Sivanantharuban
Department 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 the player is able to pick up coins. 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: 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

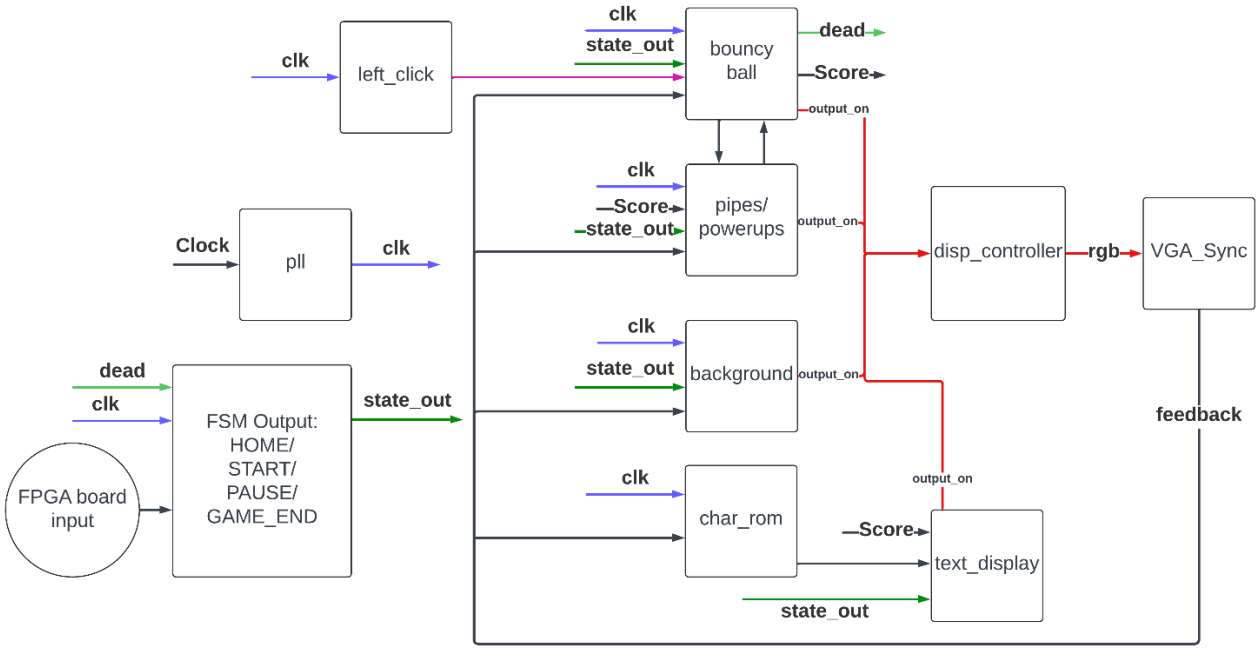

Fig 3 – Block diagram

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

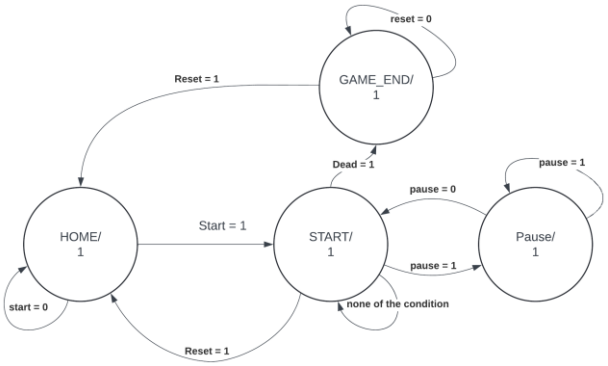

Fig 1 – FSM game state

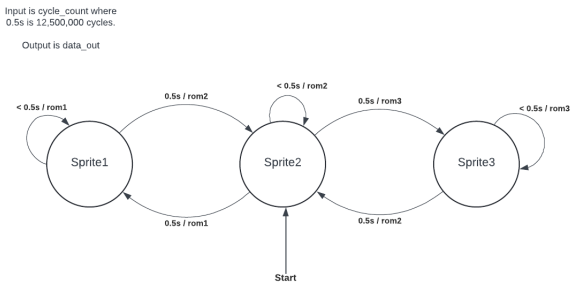

Fig 2 – FSM bird animation

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. For easier management of each state, we created a package to make the states more readable, instead of memorizing “0001” as “HOME”.

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.

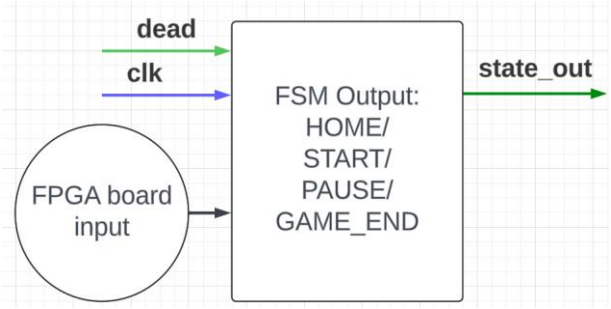

Fig 4 - FSM

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.

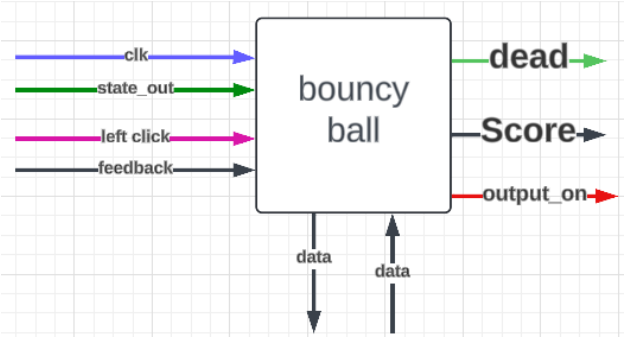

Fig 5 - Bouncy

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.

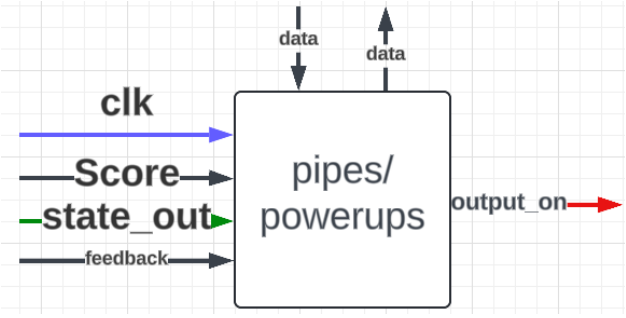

Fig 6 - Pipes

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. LFSR is implemented with reference to GPT.

Fig 7 - background

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

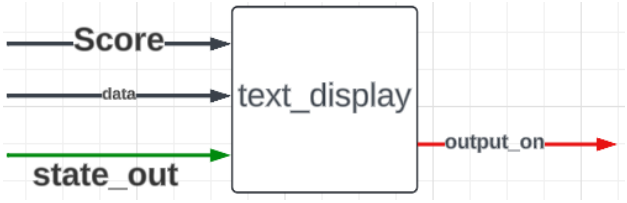

Fig 8 - text

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.

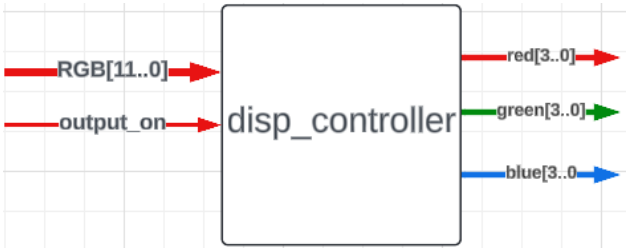

Fig 9 – Display Controller

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.

Design decision & trade offs

We have decided not to add a background using MIF file, this is due to lack of time management. Instead, we have daytime as just a plain background. For nighttime we added stars that are stored as register values in a black background. This has saved us a lot more time. In trade off, the game is not very aesthetically appealing, and the stars take up a lot of resources.

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.

Device	5CEBA4F23C7
Timing Models	Final
Logic utilization (in ALMs)	1,318 / 18,480 (7 %)
Total registers	732
Total pins	42 / 224 (19 %)
Total virtual pins	0
Total block memory bits	283,648 / 3,153,920 (9 %)
Total DSP Blocks	1 / 66 (2 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 4 (25 %)

Fig 10 – Bad optimization

Device	5CEBA4F23C7
Timing Models	Final
Logic utilization (in ALMs)	1,381 / 18,480 (7 %)
Total registers	494
Total pins	44 / 224 (20 %)
Total virtual pins	0
Total block memory bits	283,648 / 3,153,920 (9 %)
Total DSP Blocks	1 / 66 (2 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 4 (25 %)

Fig 11 – Better optimization

With reference to Fig 10 & Fig 11, both codes serve the same logic. However, Fig 10 uses a lot more registers compared to Fig 11. Initially Fig 10 is pure spaghetti code and Fig 11 we have implemented FSM. This drastically reduces the number of resources used, since most of the unnecessary signals are removed.

Results

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.

Flow Summary

<<Filter>>

Flow Status	Successful - Tue May 28 14:15:03 2024
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Standard Edition
Revision Name	flappybird21
Top-level Entity Name	flappybird21block
Family	Cyclone V
Device	5CEBA4F23C7
Timing Models	Final
Logic utilization (in ALMs)	1,463 / 18,480 (8 %)
Total registers	497
Total pins	44 / 224 (20 %)
Total virtual pins	0
Total block memory bits	199,168 / 3,153,920 (6 %)
Total DSP Blocks	1 / 66 (2 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 4 (25 %)
Total DLLs	0 / 4 (0 %)

Fig 12 – Compilation Report

Fig 12 shows a flow summary for this project, which uses 8% of ALMs and 44/224 (2%) pins.

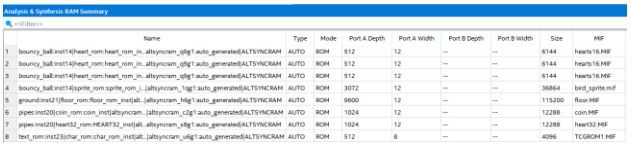

Fig 13 – RAM summary

There's a total of approximately 6% of memory bit used for storing the MIF files of the bird, power ups, ground, text.

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	51.74 MHz	51.74 MHz	inst6|pll_inst|altera_pll_i|general[0].gpll~PLL_OUTPUT_COUNTER|divclk	
2	60.91 MHz	60.91 MHz	VGA_SYNC:inst10|vert_sync_out	
3	296.3 MHz	296.3 MHz	MOUSE:inst1|MOUSE_CLK_FILTER	

Fig 14 – Fmax summary

Figure 14 shows the maximum frequency of our design, which is 51.75 MHz, the VGA sync requires a minimum of 25 MHz. Our design is far beyond 25 MHz, therefore no display bugs are expected from data transfer error.

Conclusion

In conclusion, this project aims to familiarize students with the FPGA board making a similar game to flappy bird in VHDL. The game is being displayed through a VGA monitor with 640x480 and 60 Hz refresh rate. The game is controlled by a PS/2 mouse and DE0 board. The game has unlimited attempts and the pipes gap and power up position are randomly generated through LFSR.

Future improvements

We ran short on time hence many features are not up to standard. For future improvements there are a few things we could improve and implement.

We can add sprite for pipes, this way the game would look a lot more aesthetically appealing.

We could implement background using MIF file, this way the M9K memory is used instead of using registers and have two different backgrounds. One of them is daytime and one of them is nighttime and when the player reaches the score of 25 the background would change from daytime to nighttime. When another 25 points passes, it'll be back to daytime.

Add not just power ups, instead traps as well. For example, meteorites, the bird will lose a single life if comes in contact with the meteorite.

Add some easter eggs, such as reaching the score 99 will unlock a special outfit for the bird. E.g. a space helmet for the bird.

Acknowledgement

We would like to express our gratitude to Dr. Morteza Biglari-Abhari and Dr. Maryam Hemmati; Teaching Assistants Ross Porter, Dhruv Joshi, Ehsan Behravan, Robert Sefaj. A quick shout out to ChatGPT 4/4o. The completion of the project would not be possible without our beloved lecturers.

References

[1] M. B. Abhari and H. Hemmati, "COMPSYS305," [Online]. Available: https://canvas.auckland.ac.nz/courses/104416/modules [Accessed: May 28, 2024].

[2] OpenAI, "ChatGPT: GPT-4," OpenAI, 2023. [Online]. Available: https://chatgpt.com/ [Accessed: May 28, 2024].