EE/CS 172/130 Digital Logic & Design (DLD)

PROJECT REPORT

Project Title

Flappy Bird Game

Team Members

Ali Hashir (ah05433)

Sakina Shabbir (ss05562)

Novairah Abbasi (na05886)

Mohammed Haider Abbas (ma06418)

Home Section: **T-3**

Table of Contents:

1. About the Project:

Introduction:

For our project of the course Digital logic and Design, abbr DLD, we have brought the famous Flappy Bird game to life. We believe that given a basys-3 board along with Verilog as the programming language of choice, we can showcase the maximum implementation of the course learnings through a game. Flappy Bird is a timeless game which each of us have played uncountable times, so we thought of making our very own flappy bird game. It is a single-player game and consists of two inputs - up and down which move the bird up and down respectively. The player must jump through the gap between two randomly sized hurdles in the form of pillars as in the original game, also shown in Figure 1. The goal is to score as many points by crossing the hurdles as possible and try to beat your personal best and most importantly, don't forget to have fun!

Player Instructions:

Start the game by toggling the V17 pin on the fpga board. Prepare yourself for the first set of pipes to appear. Immediately after starting the game, the bird starts moving towards the right hand side. Every up button pressed represents a wing flap which makes the bird go up vertically & similarly the down button represents a wing flap which will make the bird go down vertically. The faster the bird jumps, the higher it goes and vice versa. The player has to make the jumps such that the bird stays in a position that crosses through the gap between the two pipes. If you hit a pipe, the sky or the ground, the game ends. Find your rhythm for higher and lower pipes. It is important to determine when you need to go higher or drop, or else you will hit a pipe. Try not to go high. You can still bump into a pipe. Don't forget to have fun while playing it!

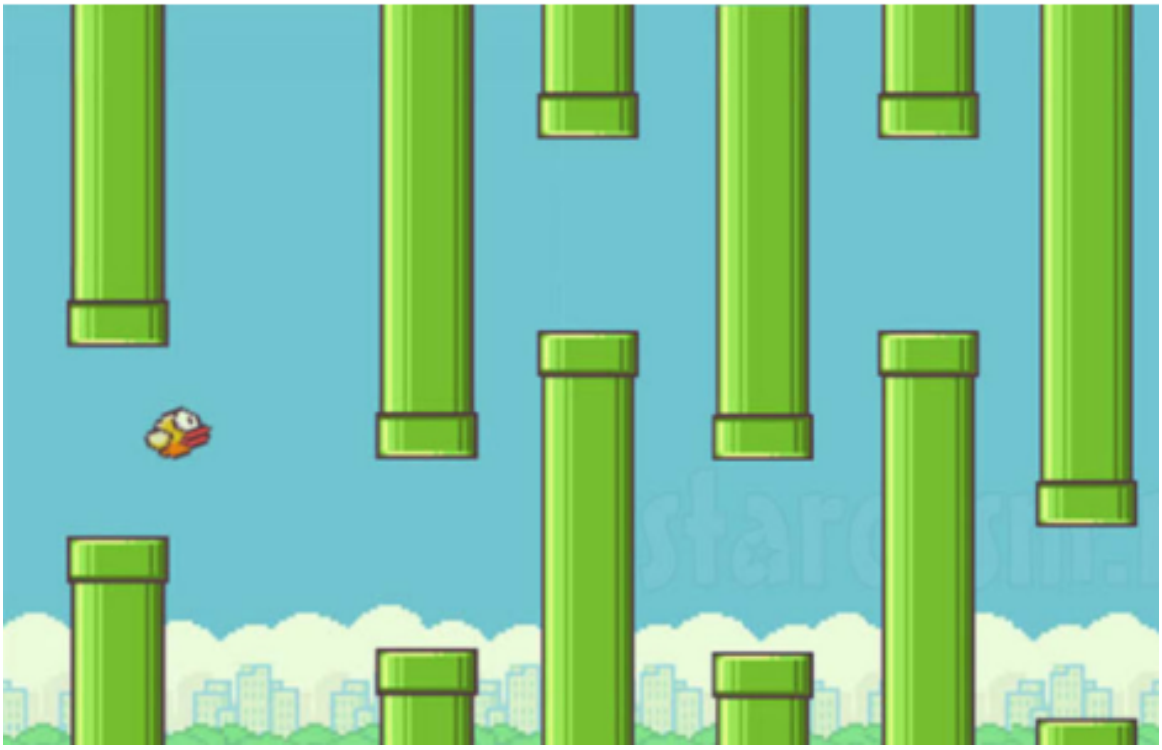

Figure 1. A Sample game snippet from the original Flappy Bird Game

Software & Hardware Requirements:

In order to run this game, the user must have the Xilinx Vivado design suite preferably a 2020 version or newer installed on their machine (Windows & Linux only, Vivado is not available for macOS as of writing of this report dated December 2022). As for the hardware requirements, the user must have a Basys 3 Artix-7 FPGA Board and a monitor for the display with a VGA cable.

2. User Flow Diagram:

User-Flow Diagram:

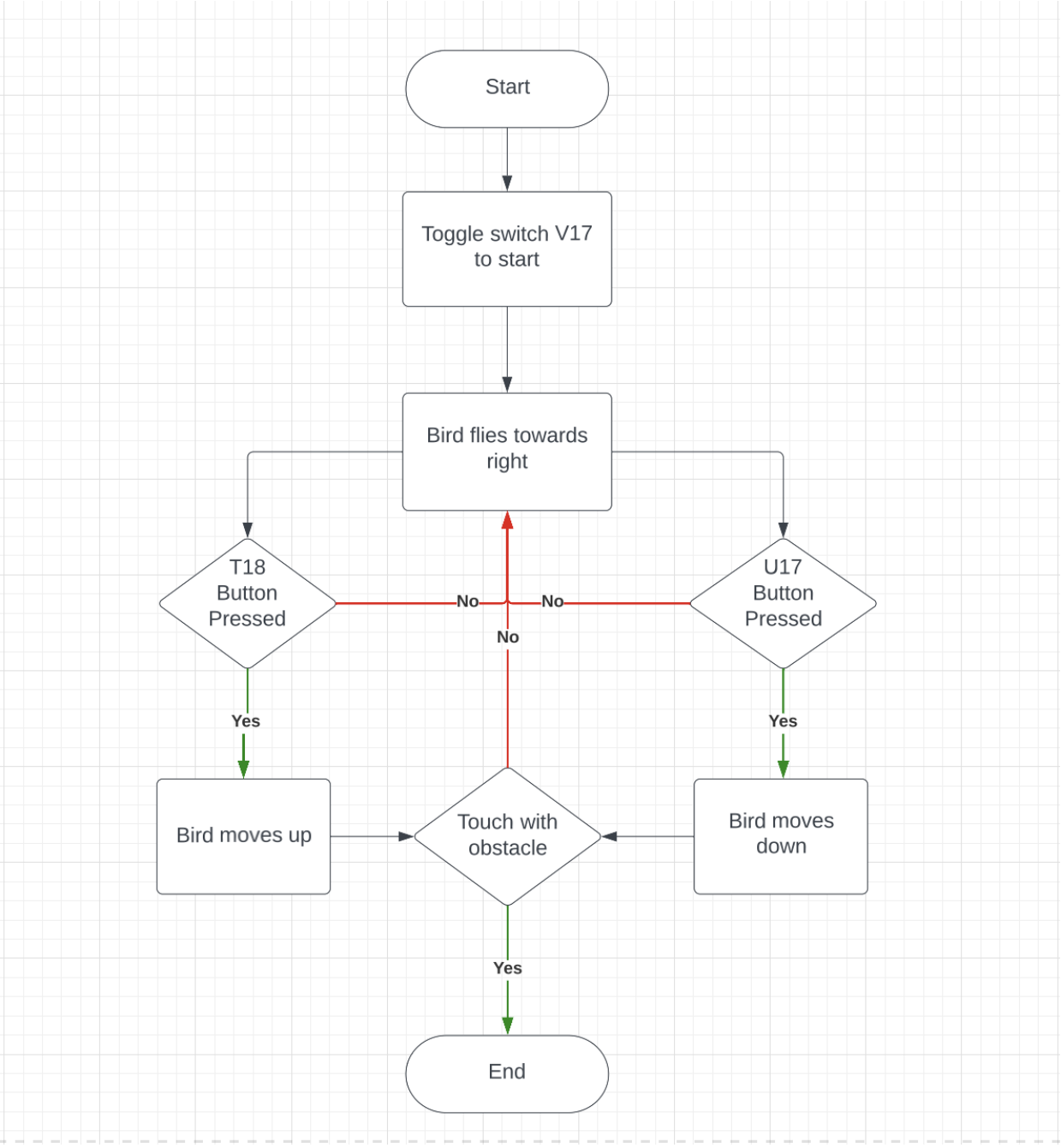

Figure 2. User Flow Diagram

Description:

All of the options available to a player in this game are neatly laid out in the user-flow diagram above. The user can control the up and down movement of the bird through the FPGA board. Navigation of the bird without colliding with the tubes is crucial to the player's success in the game. The user can ascend through the T-18 button and descend through the U-17 button on the FPGA board. The tubes will continue to appear at random until the user hits the bird by colliding with the tube. Every time it successfully completes a set of tubes, the user's score will increase automatically and will be displayed on the FPGA board.

Division of Task amongst Blocks:

The tasks are divided amongst the 3 blocks as follows:

1. Input Block: The input block provides the control block with an input for it to perform the task on. For example when we press T18 or U17 it will send it to the control block to act accordingly.

2. Control Block: The control block basically verifies the input received from the input block and ensures the output is carried out.

3. Output Block: The output block is primarily responsible for displaying the output screen which involves all pixel generation, animation effects etc.

3. Block Diagram:

Block Diagram:

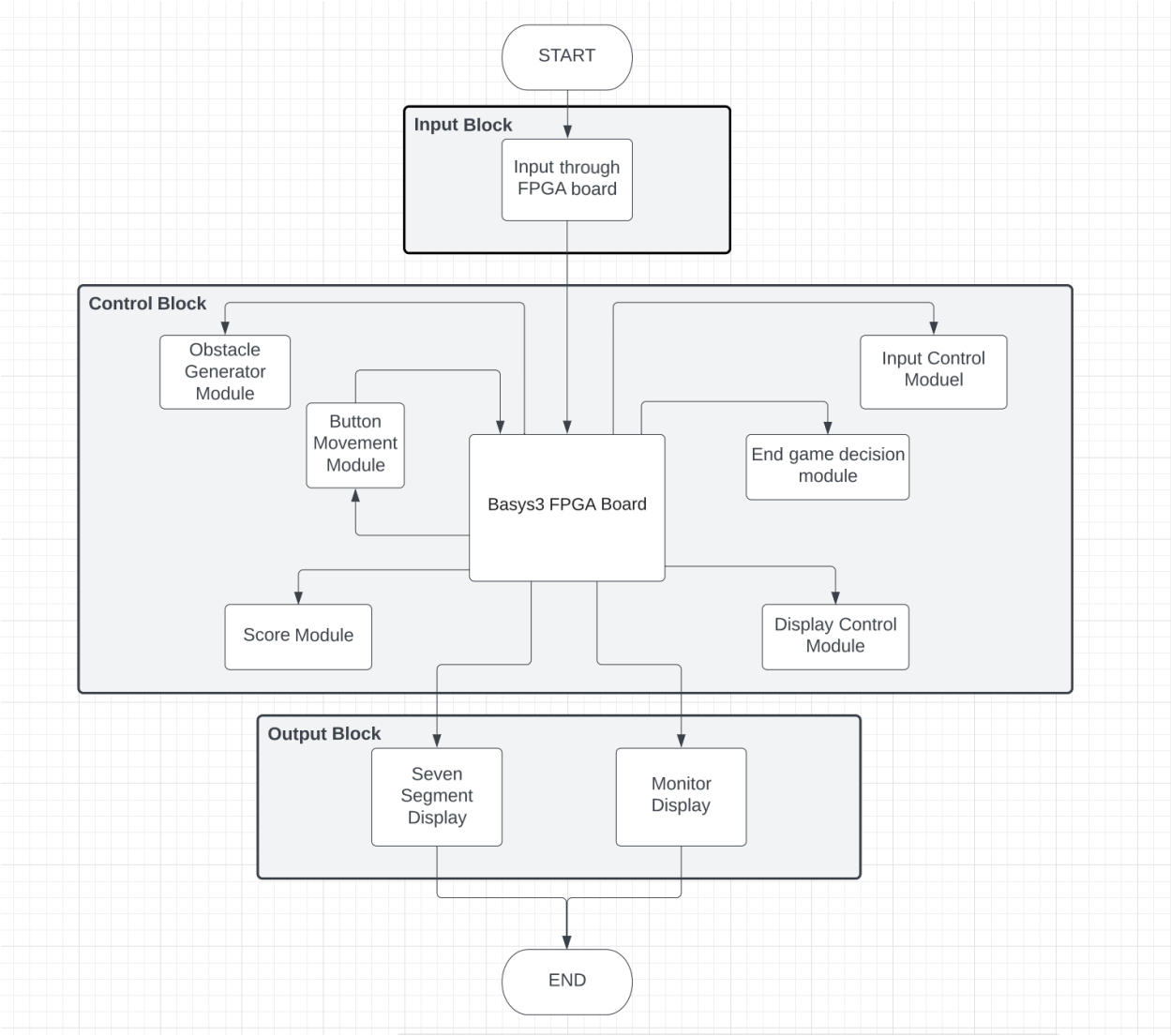

Figure 3. System Block Diagram

Description:

Our game is implemented using the 3 blocks i.e. input, control and output. The input block receives the input from the FPGA board and sends it to the control block.

The control block then using the game logic acts appropriately to ensure the output is sent to the output block. The output block manages the output and delivers it in appropriate manner.

4. Input Block:

About the input peripheral:

As the project outline clearly stated that the keyboard is not allowed as an input device, therefore we thought of using the basys-3 FPGA board itself as our input device. The FPGA board fulfills all the necessary requirements for our project and was the best pick out of all the available options since our mode of input consists of buttons only.

Inputs to the Project:

Total inputs are 3 used but we mainly have 2 inputs in our project. All 3 are buttons in the FPGA board. The T-18 button on the FPGA board acts as the up key. On pressing the up key (T-18) the state of the bird shall change to up and will move upwards while moving towards the right hand side, i.e. north east direction. Similar to the up key, the second input is the U-17 button on the FPGA board which acts as the down key and is basically the total opposite of the up key. On pressing the down key (U-17) the state of the bird shall change to down and will move downwards while moving towards the right hand side, i.e. south east direction. These 2 are the main inputs for the game. As for the third input, it is the reset button which we assigned to the U-18 button on the FPGA board. At any point in the game if the reset button is pressed then the game immediately starts over along with the score.

Input I/O ports:

Figure 4 highlights the input constraints.

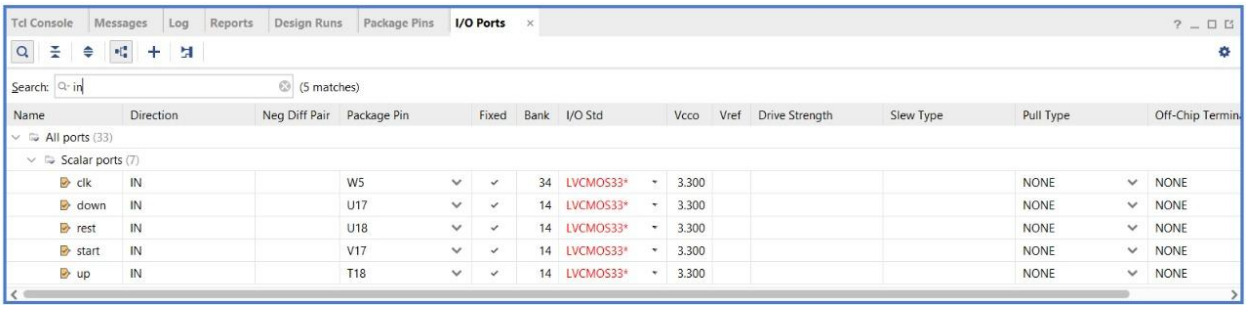

Figure 4. Pin configuration for input block

Here, the toggle switch V-17 on the FPGA board acts as a start display switch. On toggling V-17 off the VGA display is turned off therefore it is kept on by default to immediately power up the VGA display.

Input elaborated design diagram:

Figure 5 shows a schematic diagram of the elaborated design for the input block.

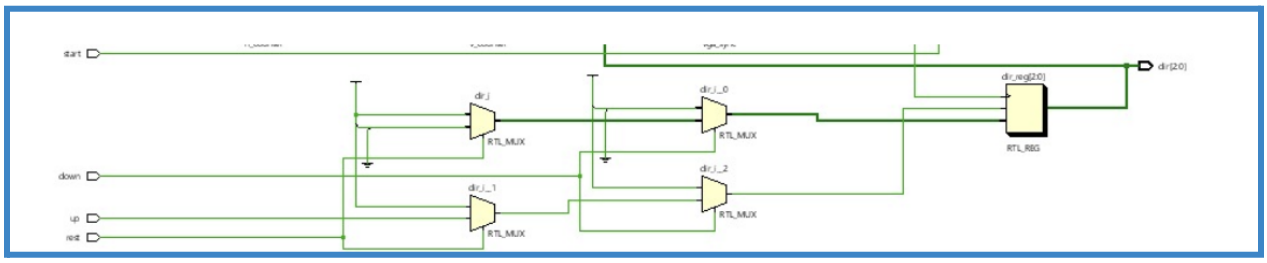

Figure 5. Schematic diagram of the Input Block

5. Control Block & FSM Design Details:

State transition diagram:

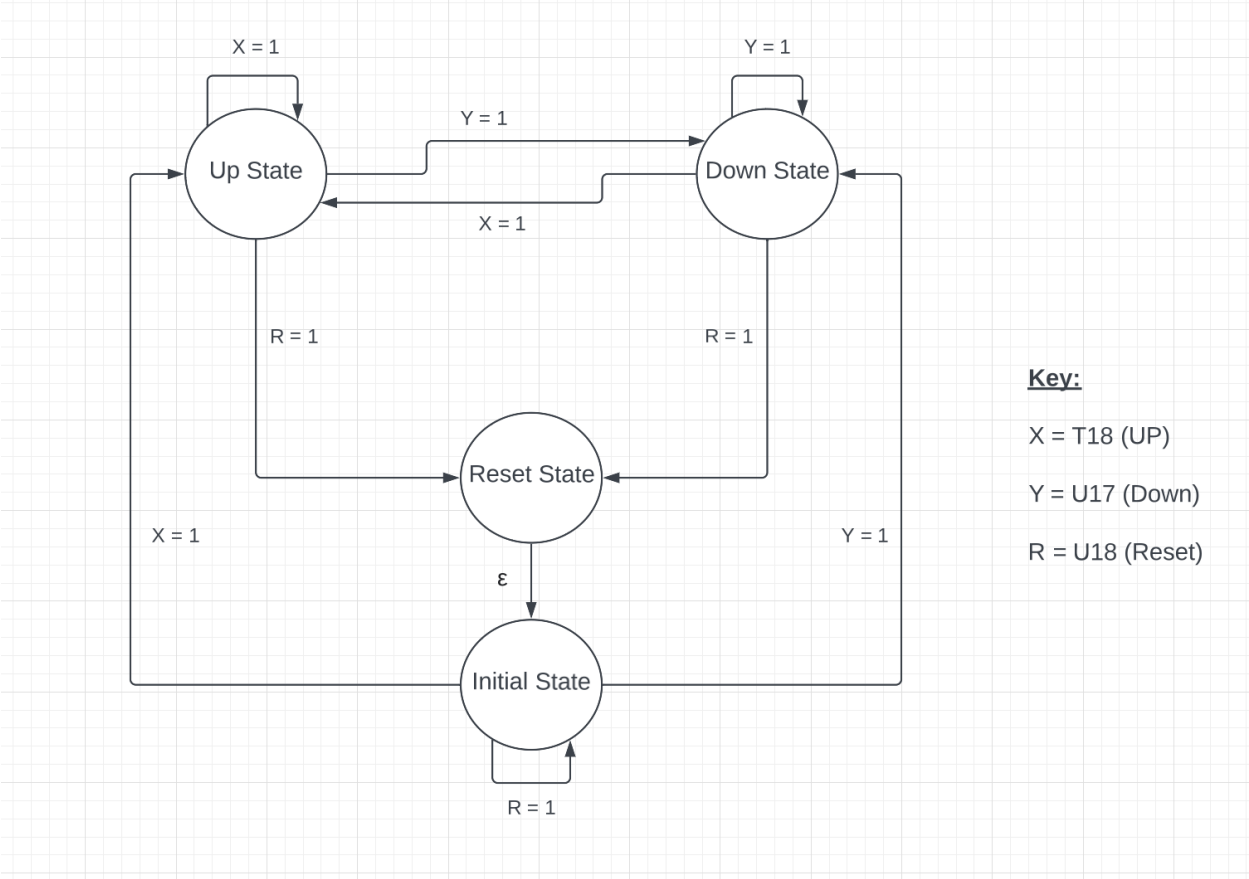

Figure 6. State Transition Diagram (FSM)

FSM design & state details:

We will mainly be having 3 states; the reset/initial state, up state and the down state. The initial state is technically the reset state but we have categorized it separately for ease in understanding and implementing our game logic. So technically there are 4 states but since reset and initial are the same so we shall consider reset in place of them both.

The first state is the reset or the initial state. When the game starts, we are in this state and will continue to do so until the up or down button is pressed. If at any point in the game the reset button is pressed then we are immediately returned to this state or if the player loses the game by hitting an obstacle, then too we are returned to this state. Also an important thing to notice here is that in the FSM, the transition from reset state to initial state is an epsilon transition which is denoted by ε. In very simple words, an epsilon transition allows a machine to change its state without consuming an input symbol. Therefore the assumption we made for reset and initial state holds true.

After the reset state we have the up state. In this state, the bird will continue to move in the upward direction if the up button is pressed. It will stay in this state until the player loses by any of the rules, resets the game or presses the down button which will change the state to down.

The down state is the exact opposite of the up state. When the down button is pressed, the bird will continue to move downwards and the player will stay in this state unless the player ends the game by hitting an obstacle, resets the game or presses the up button which will change the state to up.

Type of FSM:

This is a Moore machine since the transition to the next state is determined entirely by the present state.

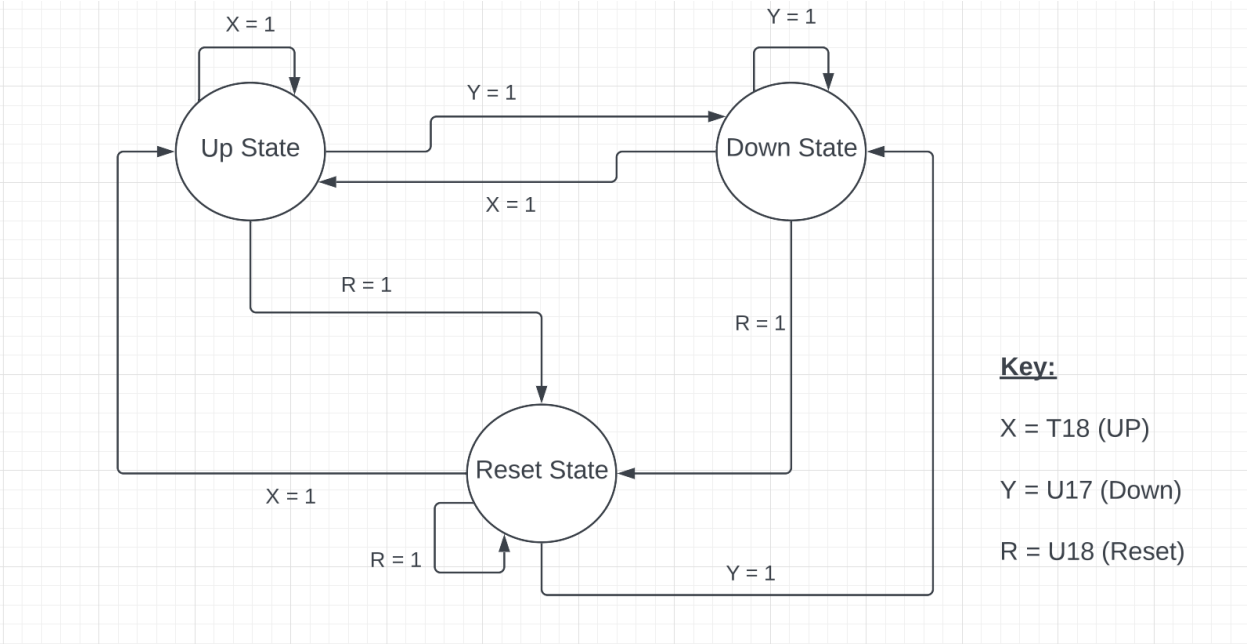

Figure 7. Another State transition diagram (FSM) - Basic Version

Top level module:

Figure 8 comprises of the top level module for our project.

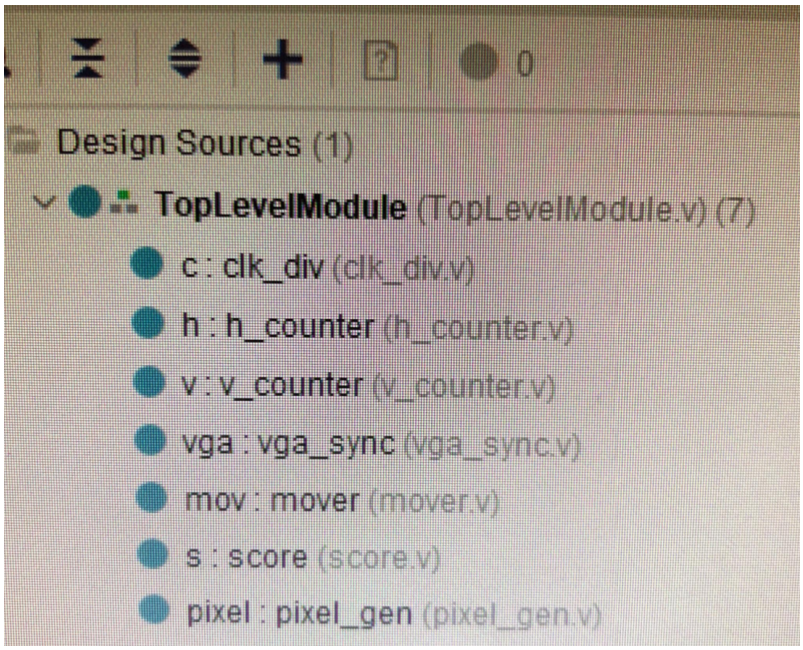

Figure 8. Top Level Module

State Table:

State Table

Current State		Input			Next States		Output Play
A	B	X	Y	R	A(t+1)	B(t+1)	
0	0	0	0	x	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	0	1	1	1	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	1	1
1	0	0	0	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	0	1	0	1
1	0	1	0	0	0	1	1
1	1	x	x	0	1	1	0
1	1	0	0	1	0	0	0

Figure 9. State Table

From figure 6 we know that X is for up, Y is for down & R is for reset.

States key:

State → Logic

Initial state→ 00

Up state → 01

Down State → 10

Reset state → 11

State equations:

Using figure 9, we have the following state equations.

$A(t+1) = ABR' + A'X'YR' + AB'X'Y' + A'BX'Y'R$

$B(t+1) = XY'R' + ABR' + A'BXY' + AB'X'Y'R$

$Output = A'B + AB'$

6. Output Block:

About the output peripheral:

The primary output device for our project is the VGA display and the score will be displayed on the FPGA board.

Output generation:

Some important points to highlight here are the statistics of the project like sizes, dimensions and movement speed control etc which are an important part of the display in the output block. The size of the bird is 20 x 25 pixels, which means 20 pixels wide or the x component being 20 units and 25 pixels tall or the y component being 25 units. Another important aspect is the x and y (upward and downward) movement of the bird. For both upward and downward movement, there will be a shift of 30 pixels in the screen. Other than that the obstacle tubes will be created on random but will have a constant gap of 80 pixels between the two opposite pillars to ensure that the game remains possible to play with enough gap for the bird to pass through.

Output I/O ports:

Figure 10 shows the output constraints.

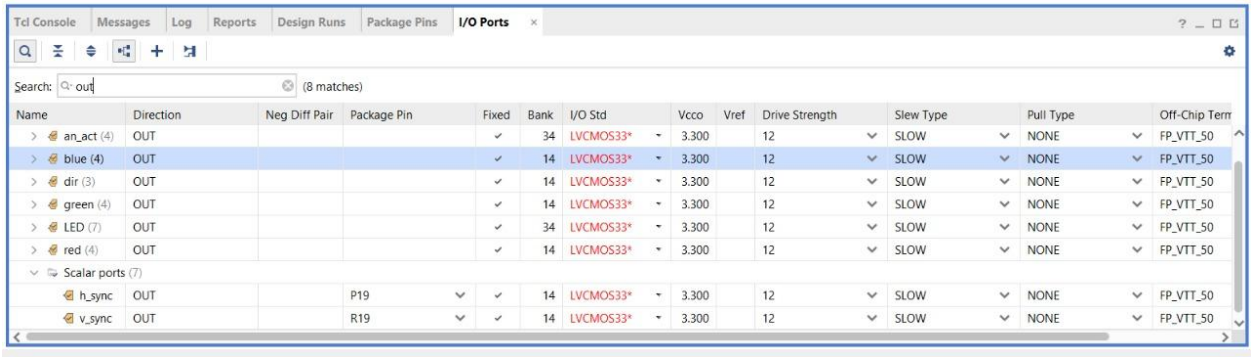

Figure 10. Output I/O Ports

Output elaborated diagram:

Figure 11 and 12 show the schematic diagrams for the output block.

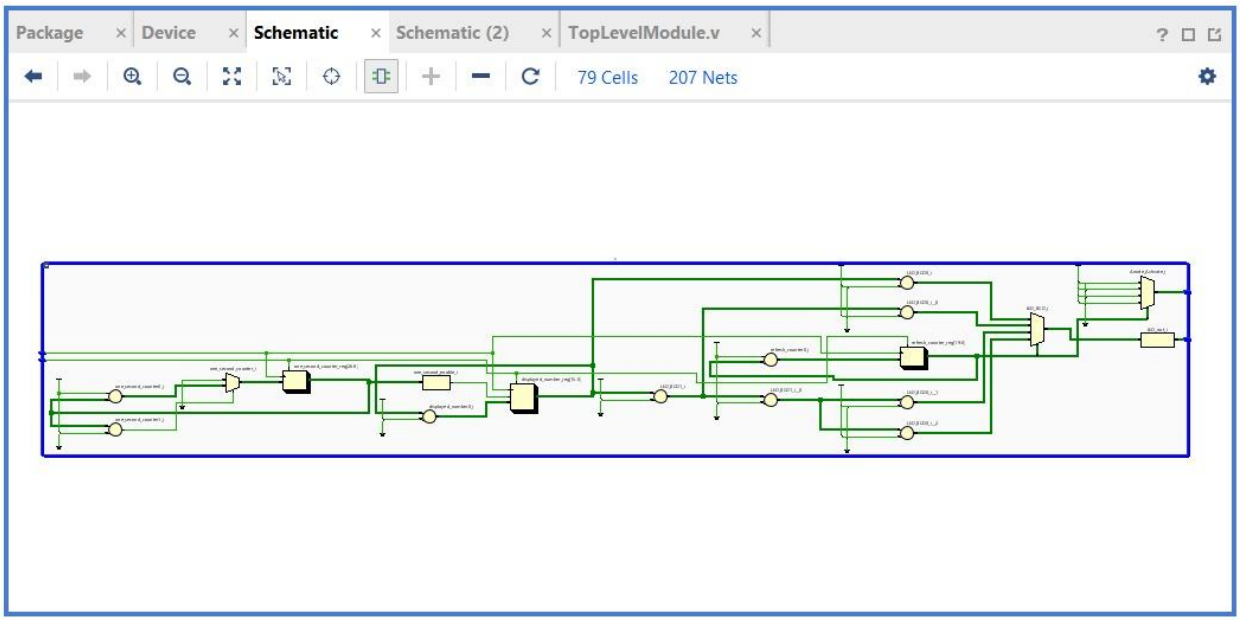

Figure 11. Schematic Diagram for the score of the game from the Output Block

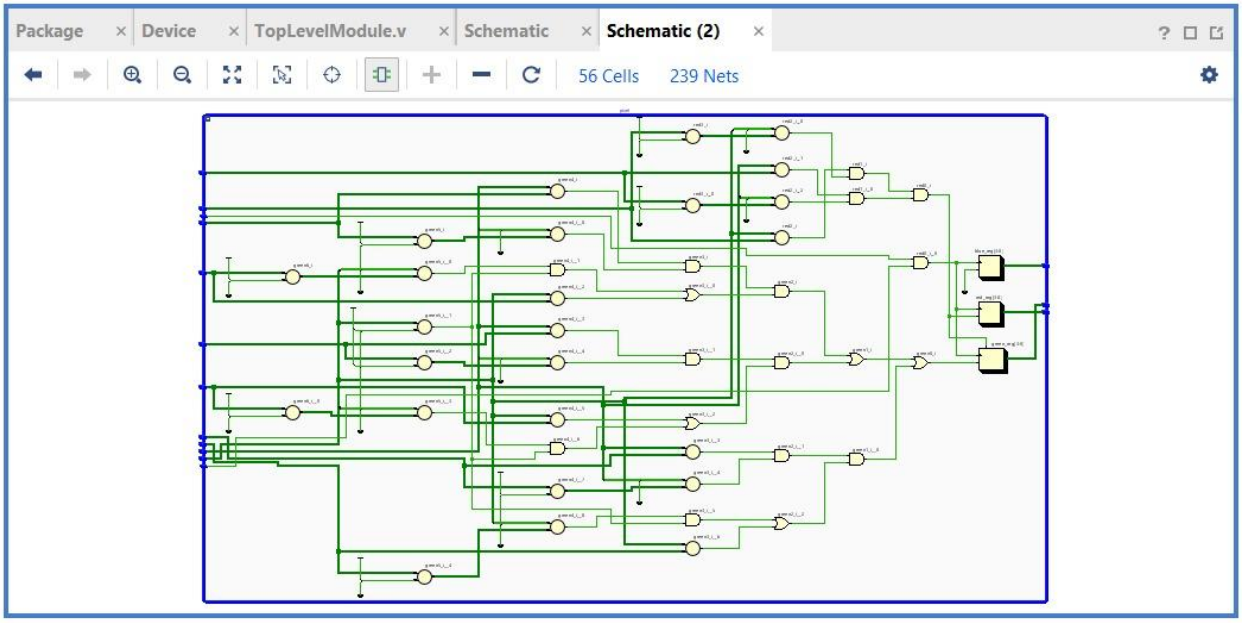

Figure 12. Schematic Diagram for pixel generation from the output block

VGA output:

Figure 13. Working output of VGA Display

7. Conclusion:

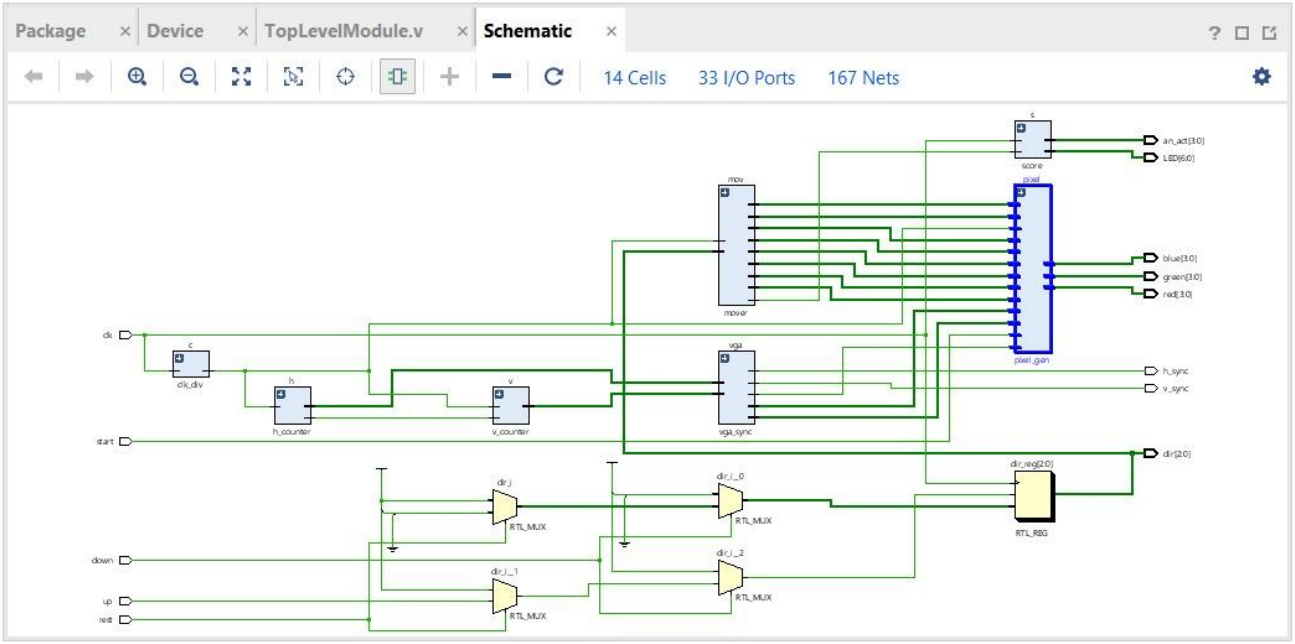

Figure 14. Elaborated Design for the Game

To conclude the report, here is figure 14 showing the elaborated design for the entire game. This report covered the entirety of our project in depth, not just the philosophical side but the theoretical wise as well. Through this report, one can easily get full insights to our project, both as a user and as a developer looking to work in this domain. The report extensively covers the user flow mechanism, system blocks which are input block, control block and the output block. Along with that it also has an in depth explanation on the FSM that we constructed along with the state table. Last but not the least we tried to incorporate almost all schematic diagrams generated by Verilog to give the reader an immersive experience. Having fun playing Flappy Bird!

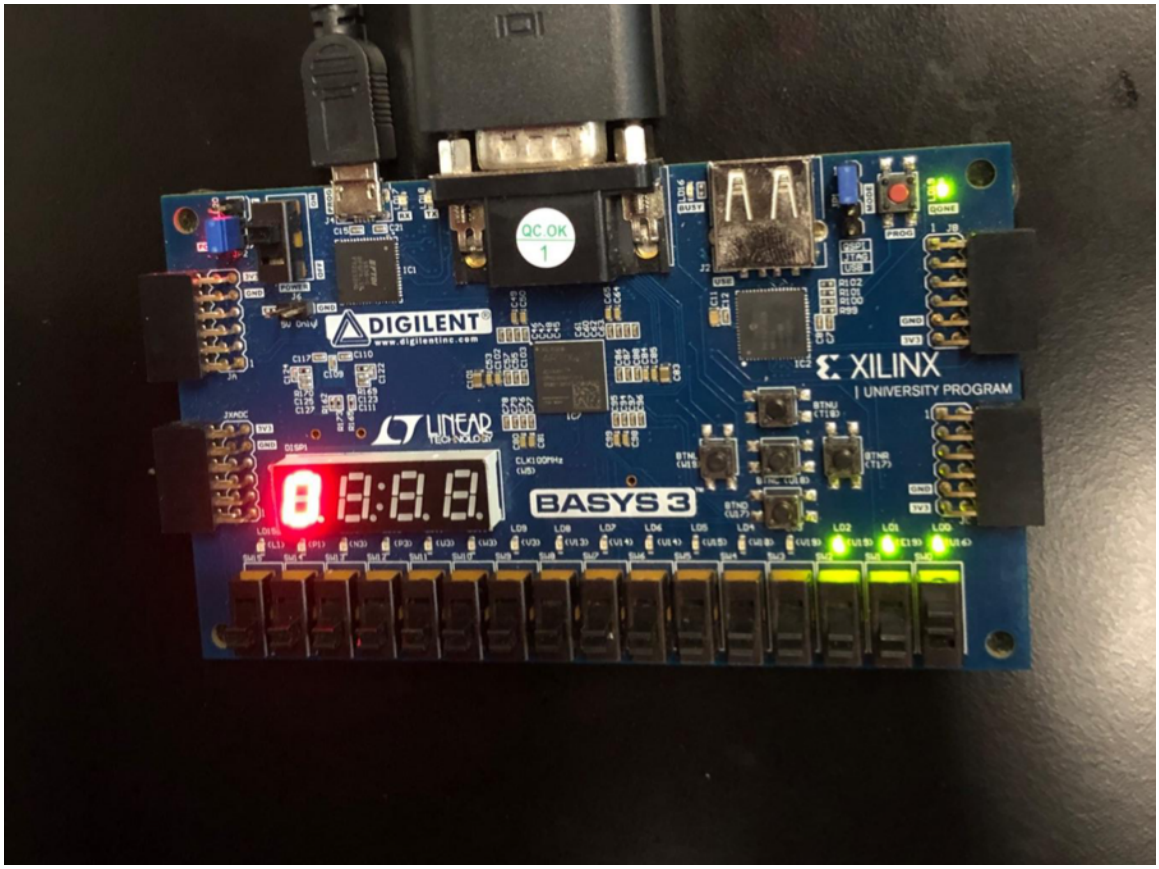

Figure 15. A Basys 3 Artix-7 FPGA

8. References:

1. https://en.wikipedia.org/wiki/Flappy_Bird

2. https://www.chipverify.com/verilog/verilog-tutorial

3.https://www.elprocus.com/finite-state-machine-mealy-state-machine-and-moore-state-machine/

4.https://psmag.com/economics/flappy-bird-candy-crush-still-making-much-money-75048

5. https://flappybird.io/