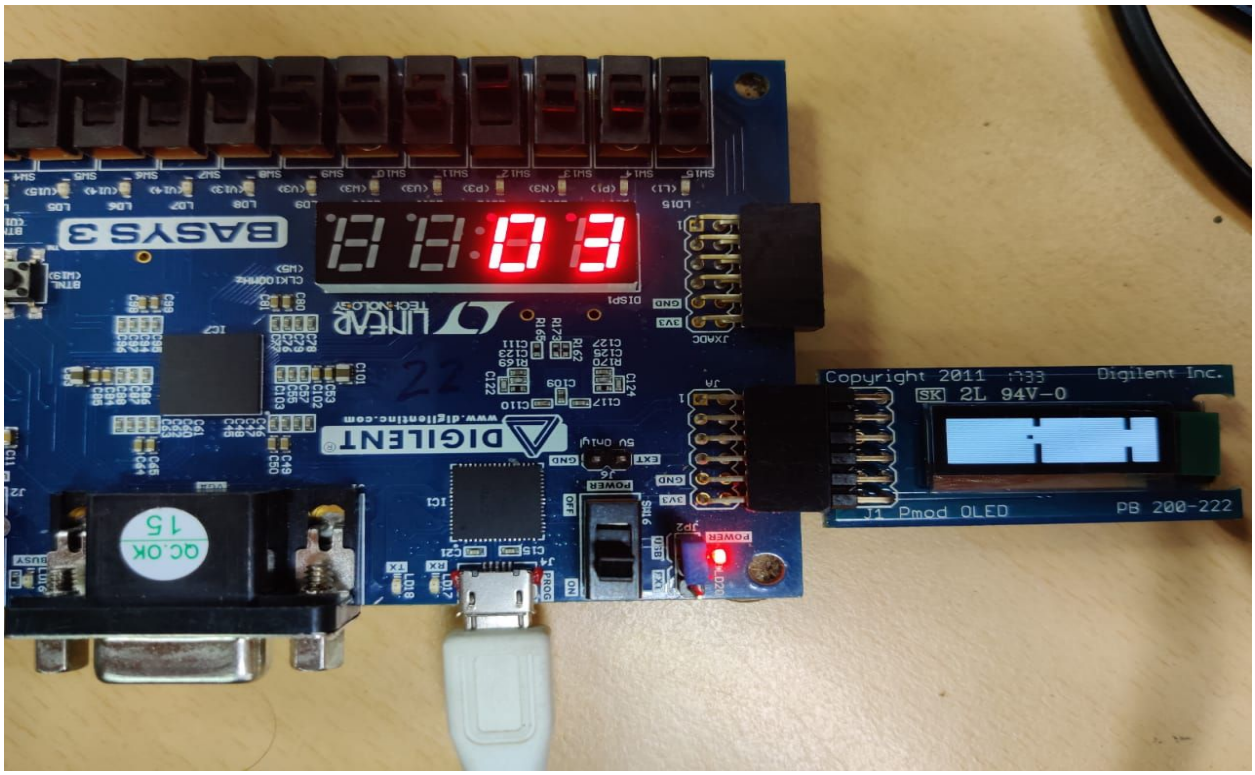
COL215

Flappy Bird using PmodOLED

Abhyuday Bhartiya
Sushant Sondhi



Introduction

Flappy Bird is a mobile game developed by Vietnamese video game artist Dong Nguyen, under his game development company dotGears.[1] The game is a side-scroller where the player controls a bird, attempting to fly between columns of green pipes without hitting them. If the player touches the pipes, they lose. Faby(the bird protagonist) briefly flaps upward each time that the player taps the screen; if the screen is not tapped, Faby falls because of gravity; each pair of pipes that he navigates between earns the player a single point. Due to it's immense popularity and addictive nature , it was pulled off Apple Store and play Store by the creator Dong himself.

Controls-

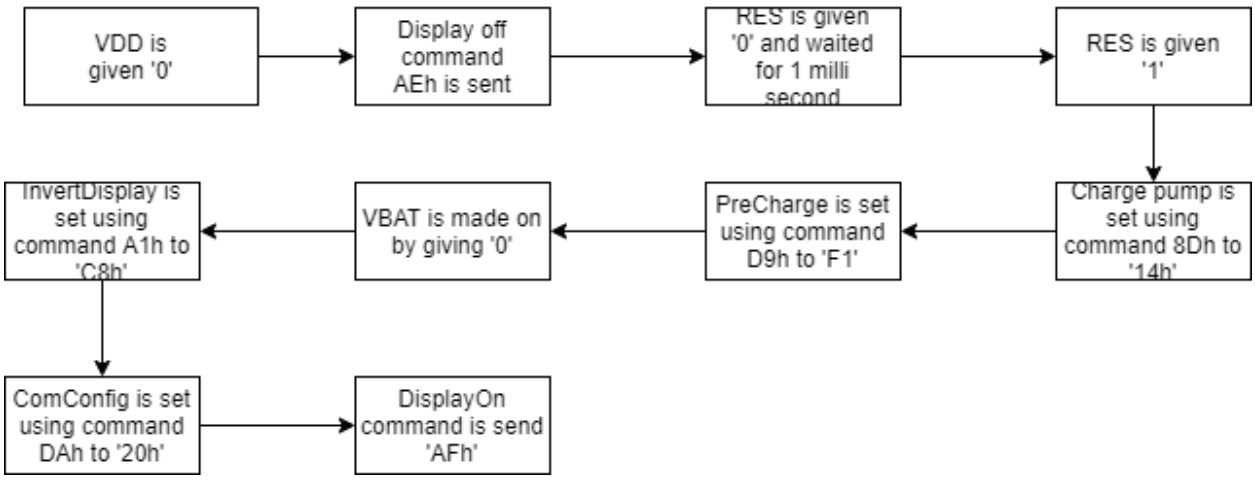
1. Start/Jump Button (BTNC) - starts the game and also makes the bird jump when playing the game.
2. Pause Button (BTNR) - pauses the game during gameplay. If pressed again after pausing, it ends the game.
3. Resume Button (BTNL) - resumes the game if the game is paused.
4. Reset Button (BTNU) - resets the game during any point of time when playing the game or even when the game gets over. After pressing reset button, the game can be started again by pressing Start button

Our Project

We have recreated the game using FPGA board and Pmod OLED. The code for the project was written in VHDL. The player can start and pause the game anytime he/she wants and also quit using buttons on the FPGA board. The score of the player is displayed on the seven-segment display and it gets updated as the game progresses. The project can be broadly classified into 2 sections:-

1. Starting OLED.
2. Updating OLED as the game progresses

Starting OLED



Updating OLED

The game begins with the bird at the centre of the screen and its position gets updated after every 0.1 seconds as moving one vertical block down. Similarly, the pillars are moving from the right to left with their position being updated every 0.05 seconds. After every 10 points scored, the game becomes a notch more difficult, by increasing the speed 1.25 times. Also, the pillar heights are different to make the game more enjoyable and challenging.

If the player presses the middle button on the FPGA board(bTnU) then the bird jumps up and avoids falling to the ground. This way the user can navigate a path across the gaps between the pillars and keep increasing their score. Every time the user passes through a pipe its score gets incremented by 1 and this can be viewed on the seven segment display.

The updates to the bird and the pillars are done in a separate process than the one in which the core state cycle of the game is run. In the cycle we check for pause and play inputs by the user and also update a 128 X 32 matrix which is sent to the PMod OLED for display.Following is a brief description of the state cycle:-

Idle - The program checks if the pause button(bTnR) has been pressed and if yes, puts the game in a frozen state. Otherwise, the program moves to checkNegative state

checkNegative - We check if the player has hit the top or bottom of the screen which will cause the game to end and the player to lose. If not, we move to the state updateMatrix.

updateMatrix - Here based on the position of the bird and the pillars the Display Matrix is updated. From hence worth, we first set the configuration for the PModOLED to receive data. We have used **horizontal addressing mode** because it required a command to be set only once for each display update. Then we move to the sendMatrix state.

sendMatrix - We send the data byte-by-byte using the component SpiCtrl and then enter the state checkCollision.

checkCollision - In this state, after sending the data, we check whether a collision has occurred between the pillar and the bird. Our condition for collision is the overlapping of the coordinates of a pillar and the bird. The bird has been modelled as a 3 X 4 rectangle while the pillars are set at varying heights but with a uniform width of 5 pixels and gap of size 12 pixels.

Paused- The game remains in the paused state, and the program waits for the player to press either the pause or the play button. In the scenario that the player presses the pause button(bTnR), the game is quit and a blank screen is displayed. On the other hand, the game resumes if he/she presses the play(bTnL) button.

Game_over- When there is a collision with pillars or with with the top or bottom of the screen, the game is sent to this state where the screen is initialised to blank.

This essentially describes the entire state cycle.

Auxiliary components of the project include :-

Delay - allows us to wait for a certain time where the time is sent as a parameter.

Spi_comp2 - Sets the signals such as SDO, CS, SCLK etc to be sent to the PModOLED.We communicate with the PMod at a frequency of 3,125,000 Hz i.e 10MHz /32.

DeBounce - For debouncing the input received from the buttons JUMP, PAUSE, PLAY.

SevenSegDisplay - Contains map from integer to 7-bit sequences to be sent to the Seven Segment display on the FPGA display to display the pattern corresponding to the matching integer.

ReducingFrequency - Allows us to synthesized a clock with a frequency reduced from 10MHz which is useful for executing updates to the bird and pillars at set intervals.

Innovations:--

1. By pressing reset button (BTNU) at any point during the game play or when the game gets over, everything is set to reset state and then pressing jump button (BTNC) starts the game again.
2. After every increment of 10 scores, the game levels up and the speed of game increases to increase the difficulty.

Conclusion

All in all, this project has been a one of a kind experience for us and we have learned a lot in the process of making this game both in terms of digital logic and the designing and customization of game user interface. We thank the professor for giving us this opportunity.