Team 07 Information Processing Coursework Report

1. System Purpose

Overall, the system is implemented in the form of a structured battle plane game. The system has three stages: security verification, the actual game, and results display.

In the first stage, a welcome window is displayed, and it asks the user to either sign up with a new username and a password or login in with existing information. Once the user has successfully signed up with a new username or passed a security check, the user will enter the actual game with the surname displayed on the FPGA.

The game allows the user to control a battle plane with three lives in the middle of an air battlefield. There are several types of enemy planes coming randomly from all directions. With each enemy being taken down, the user will gain points corresponding to the types of enemy planes, and the level of difficulty accelerates as points increase. In addition, there will be bombs and machine guns with more power that users can collect and choose when to use. As the points collected reach a certain threshold, the "boss" in the form of a giant bomber with extremely high health points will appear on the screen.

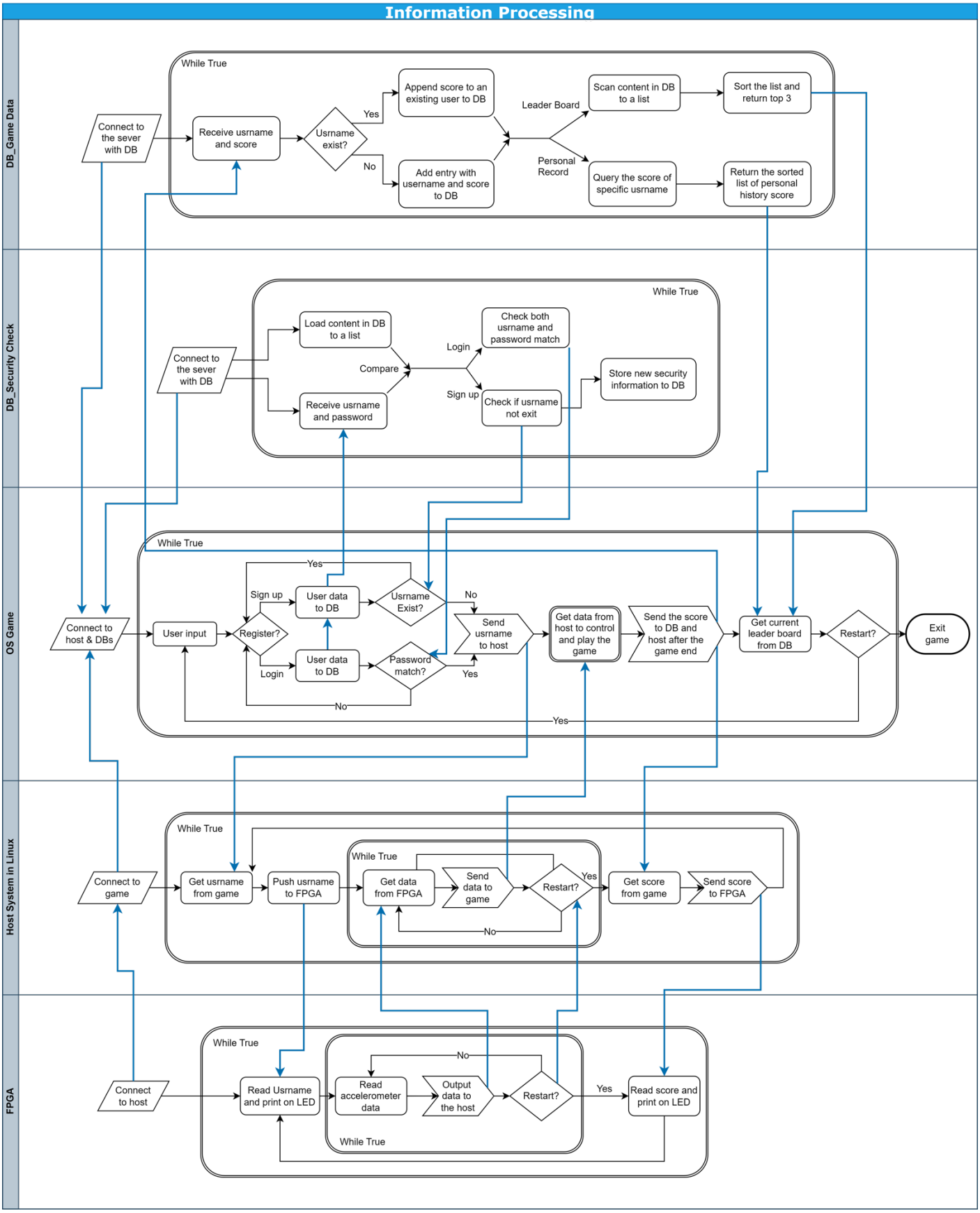
The game ends when the boss is defeated, and a leader board will be displayed with the three top scores and their corresponding usernames. Also, the user's best score will be shown on the screen as well as on the FPGA board. When the game has finished, the user can choose to restart or terminate the game.

2. Overall Architecture

The system comprises four components: the FPGA block, the host in Linux, the game OS, and the database block. They are individual blocks but are interconnected by TCP connections for data transmission purposes.

- The FPGA block acts as the controller to move the battle plane in four directions, and the two buttons on the side are used to drop the bombs and pause/resume the game. It is also responsible for displaying the username as well as the score.
- The host connects with FPGA through UART using nios2-terminal to establish str communication between the host and the FPGA. Then, it communicates with the game for data transmission through the TCP portal.
- The game block is the actual battle plane game, and it communicates with the host and the cloud databases through TCP connections
- The database block uses two separate tables on two different servers to store the security information (username and password) and the game results (username and score). In both tables, the username is stored as the primary key, and the password and the score are stored as the attributes.

Team 07 Information Processing Coursework Report



3. Performance Metrics

- **The delay between the game and FPGA**

  This metric assesses the performance of the interaction with the user and the game. This delay should be as low as possible, or users can barely play the game if it takes more than 2~3 seconds to give a response to users' actions.

- **Sensitivity of using FPGA to control the game(vertical/horizontal)**

  Same as above, this metric directly indicates how well users can control the game. If users already move the FPGA board with a considerable amplitude but just cause a slight movement in the game, the user experiment would be terrible.

- **The delay between game & database**

  The connection between the game and the database stored in the AWS cloud is built by a TCP socket. The local game sends user information (username, password, and score) to the database and waits for a response from the database. Therefore, if there is a significant delay, users have to wait a relatively long time to log in/sign up for the game and see/her best history score and the leader board.

- **Speed of the 7-segment display shows the username at the beginning and score at the end**

  Another factor that will cause a long waiting time is how fast the 7-segment LED can display the data. We expect the LED can show the username and score as soon as it receives this information.

- **Amount of data stored in the cloud database**

  The whole system is expected to be as light as possible, which means do not occupy too much storage and is easy to establish. For this purpose, we should only select the most important data to store in the cloud.

- **Accuracy and reliability of outputting and displaying the leader board**

  The information displayed by the LED should be accurate and understandable enough. Besides, the game uses two physical buttons on the FPGA board press the upper one for using the bomb, the lower one for pausing the game, and press both to finish and see the scoreboard. So, the FPGA is expected to be programmed well to make these functions reliable enough, or users may press buttons but with no reaction.

- **Degree of automation**

  The whole program should achieve some basic automation. For example, for continuity of playing the game, it ought to do the restart, log in, sign up, and display scores actions automatically, instead of manually doing these steps.

4. Design Decisions

- **Security verification & score performance in separated tables**

  Instead of storing the username, password and scores in one table, we have chosen to store the username and password in one table as "security check" and store the username and score in another table named "game". In this case, it is much simpler when writing python scripts to conduct operations to the tables without using *FilterExpression* to locate the exact attribute. However, it is with the cost of creating an extra table.

- **Having 2 databases on different servers to reduce complexity & increase reliability**

  We decided to use 2 servers for the security check & score database because they won't have any conflicts when the username is only a unique key to both. Besides, we're using TCP connection rather than UDP, so that we'd like fewer data to be transmitted each time to ensure transmission speed and reliability. Moreover, in this way, the server-side does not need to determine which database to access, which saves the processing time and reduces complexity. The time to transfer data between the server (database) and client (game) within a delay of 0.05s is 150% faster than using 1 single server but can be further improved by using UDP connections.

- **Use of Python and PyGame Library**

  Instead of using Unity 3D or other more advanced game engines, we have chosen to write the game using IDLE (python 3.7, 32-bits) and PyGame, which we are more familiar with. This is because PyGame can already meet all the needs of our game, such as graphics and collision detection and given the time limit and the amount of work devoted to learning new software, it is more time-efficient to just use PyGame.

- **Pop-up Window for User Input & Display**

  For security verification, leader board and history score display, and restart show that require user input to proceed, we have chosen to use a pop-up window outside the game instead of terminal input or an embedded display inside the game. An embedded display is difficult and time-consuming to achieve, and the terminal input lacks an exemplary user interface and interaction. Therefore, a pop-up window is used to balance the design complexity and good user interface.

- **Choice of Display Content of 7-Segment LED on FPGA**

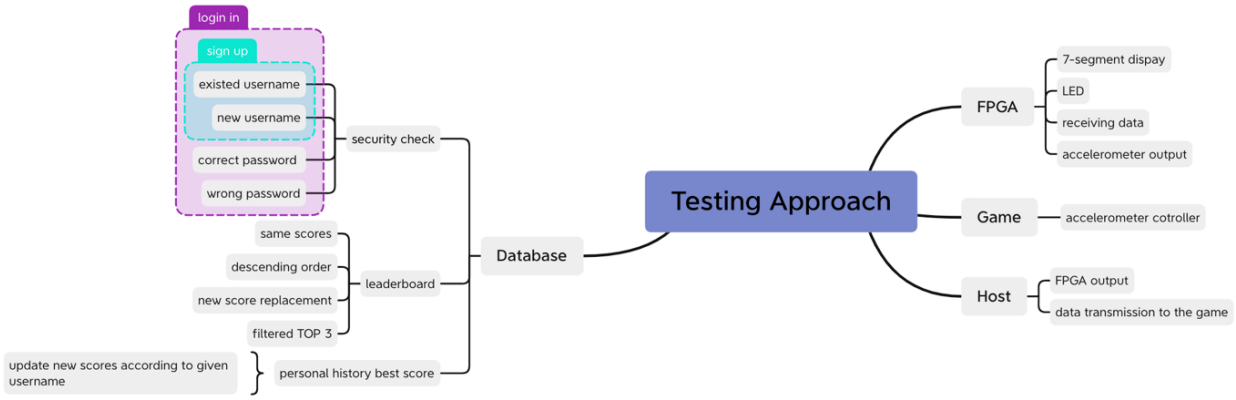
  We optimize delay between game and FPGA by reducing unnecessary functions. For example, we delete the part for printing the real-time value of the accelerometer, so the delay from the aspect of the game can be reduced dramatically. Besides, we attempt to display the real-time score on the seven-segment LED display only send

- **Coding Optimization**

We decided to code the classes of each object in their sub-files and created these objects inside the main code file to improve the readability and the comprehensibility of the project.

- **User Gaming Experience Optimization**
    1) To improve the user experience, we designed eight levels of difficulties, including a final boss and the level of difficulties increases as the user score increases.
    2) The whole game relies on collision detection to run, for example: by detecting the collision between the bullets fired by the aircraft controlled by the player and the enemy aircraft to reduce the enemy's health points and determine whether the enemy is destroyed to increase the player's total score.

5. Test Approach



- **First Phase of Testing**
    1) FPGA & Game

       Testing of the game controlled by FPGA board to ensure the user can play without noticeable delay
    2) Database on EC2 server

       Test of a temporary client that simulates the communication between game & server to ensure the server can respond promptly to the client's message with the database modified
- **Second Phase of Testing**

  Integrate FPGA, game and database on EC2 server to ensure each stage connects smoothly
- **Third Phase of Testing**

  Integrate UI to the game to ensure user's input, pop-up windows and display of data as expected

6. Resources utilized for the DE10-Lite from Quartus

- **CPU**
- **On chip memory**
- **Jtag_uart**
- **LED**
- **Timer (for interrupt)**
- **Accelerometer**
- **Seven segment display (0-5)**
- **Button**