ECE241 Digital Systems
Final Report

Piano Tiles

Bassem Shaker 1001190169
Yusra Syeda 1001466585

Introduction

As the ECE241 final project, we decided to make the game Piano Tiles. This is a one player game which consists of a screen divided into five columns, each column corresponding to a different switch on the DE1 board. Each column has a tile falling down towards the bottom of the screen. The objective of the player is to turn on the corresponding switch of the correct tile when it reaches the bottom of the screen, this increments the score by one However if the player misses the tile, the score is decremented by one. The player has one minute to accumulate as many points as they can, and achieve a high score. Furthermore, hitting the tile at the bottom of the screen results in a sound output with each column having a different sound frequency.

We brainstormed several possible project ideas, and decided to take on this project because it had a reasonable scope, it provided us a good challenge, and we believed we would be able to complete it within three weeks. After choosing the project, we set the following goals for every week:

For week one our goals were to:

- Develop a detailed structure for the verilog code, including the function of each finite state machine and various modules
- Understand how to use on chip memory, to allow us to smoothly move five objects on the screen simultaneously
- Generate tiles on the screen at random, using a linear feedback shift register

For week two, our goals were to:

- Add animation to the tiles by having them move down the screen and reappear at the top of the screen
- Implement the finite state machines and make sure they were working perfectly
- Load the background image and tile image from memory respectively when erasing and drawing tiles

For week three, our goals were to:

- Make the tile disappear if it has reached the bottom of the screen and the player has turned on the corresponding switch
- Implement the score functionality (adding one if hit, and subtracting 1 if not hit)
- Implement the high score functionality
- End the game in one minute, and allowing the game to start over upon a key click
- Implement the sound
- Output the score and high score

The Design

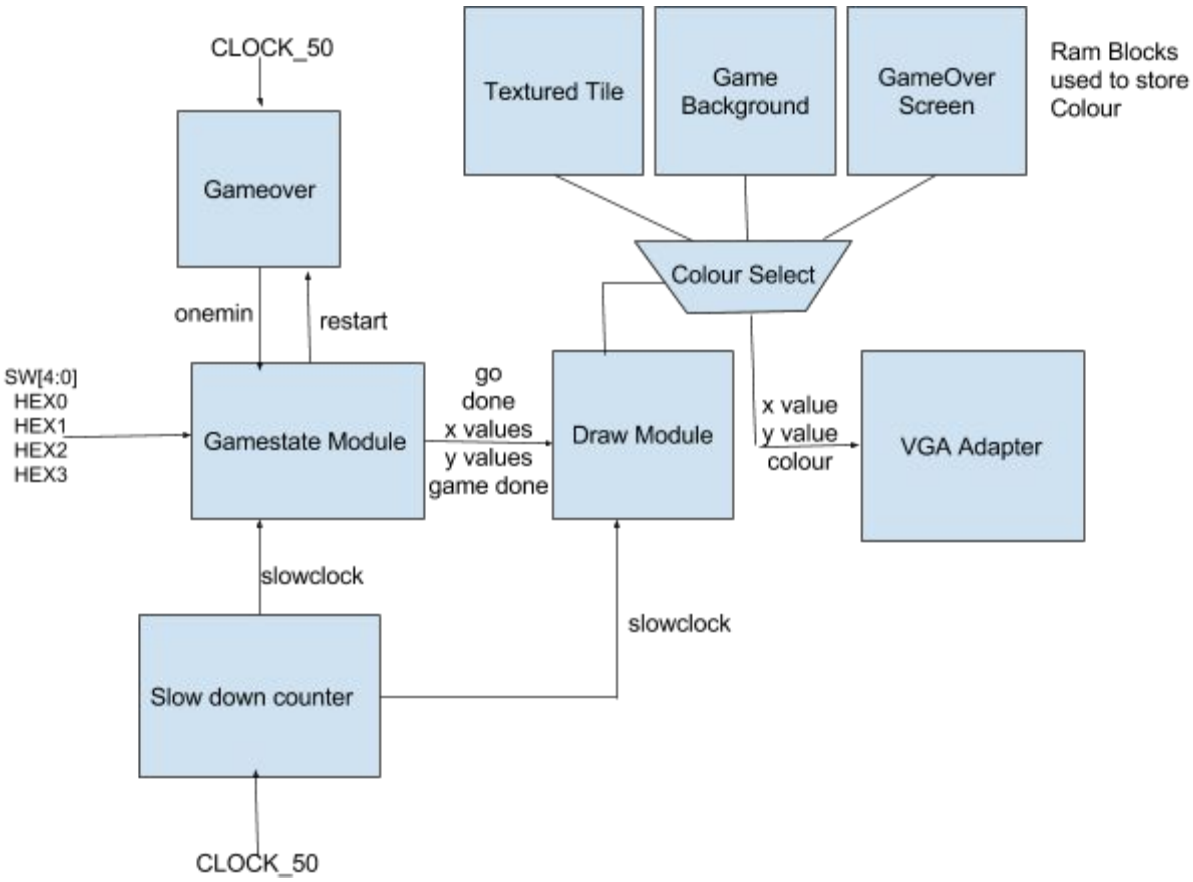

Figure 1: High level block diagram

Overall Game State Description

This hardware game consists of two FSMs that manage both the overall game state and the drawing portion of the game.The Game State FSM will first start by sending a go signal to the Draw FSM to draw 5 tiles in predetermined x and y positions. Then, as long as the game has not completed one minute of gameplay, it will continue to cycle between Supdate and Sdraw. Once a minute has been reached the Game FSM will proceed to Sgameover that once again sends a go to the Draw FSM to draw the gameover screen and waits for the user to hit KEY[0] in order to restart the game. Refer to figure 2.

Gamestate FSM

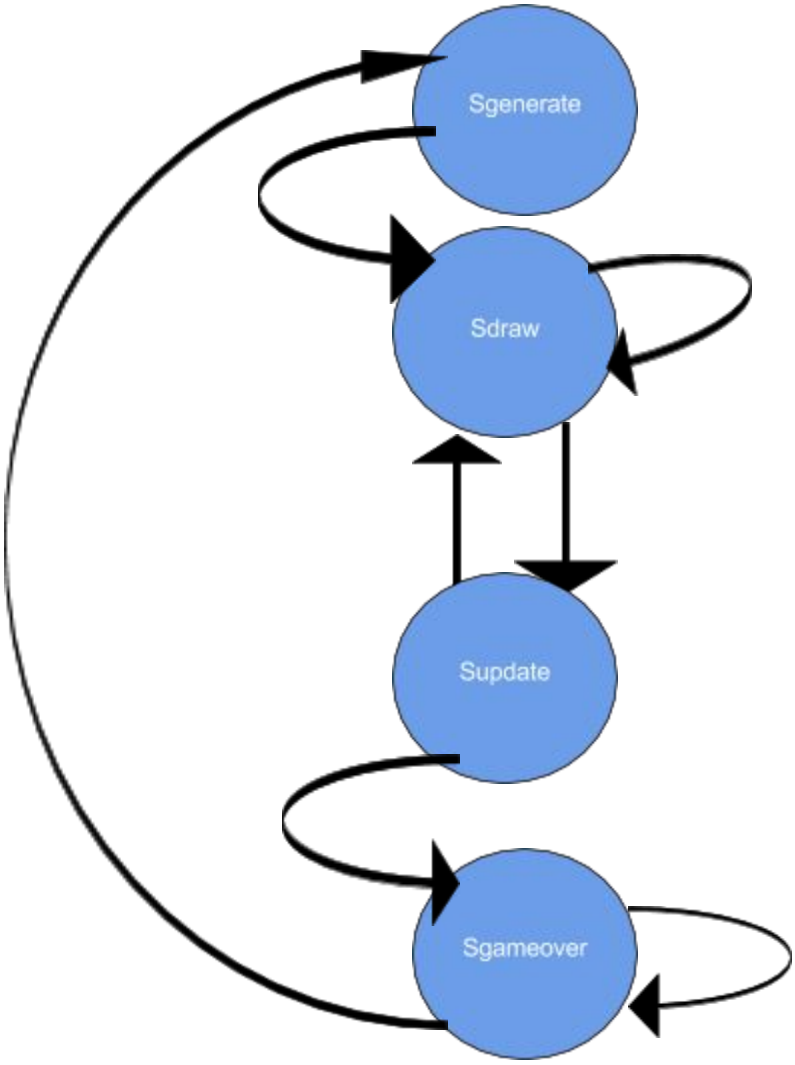

Figure 2: Gamestate finite state machine

Draw FSM Description

The Draw FSM starts at Sreset. Upon receiving a go signal it will go to Scheck which cycles through each pixel on the screen and checks if the x and y coordinate is of the predetermined set of 5 coordinates for each tile. Once it has found a tile will erase (draw background at specific tile position) it and redraw it. This cycle of check, erase, and draw continues until all 5 blocks have been detected and drawn. Once done, the Draw FSM will send the Game State FSM a done signal to notify it that it has completed drawing the 5 tiles. The Draw FSM wait for a go signal from the Game State FSM to redraw the tiles again after every update state. Refer to figure 3.

Draw FSM

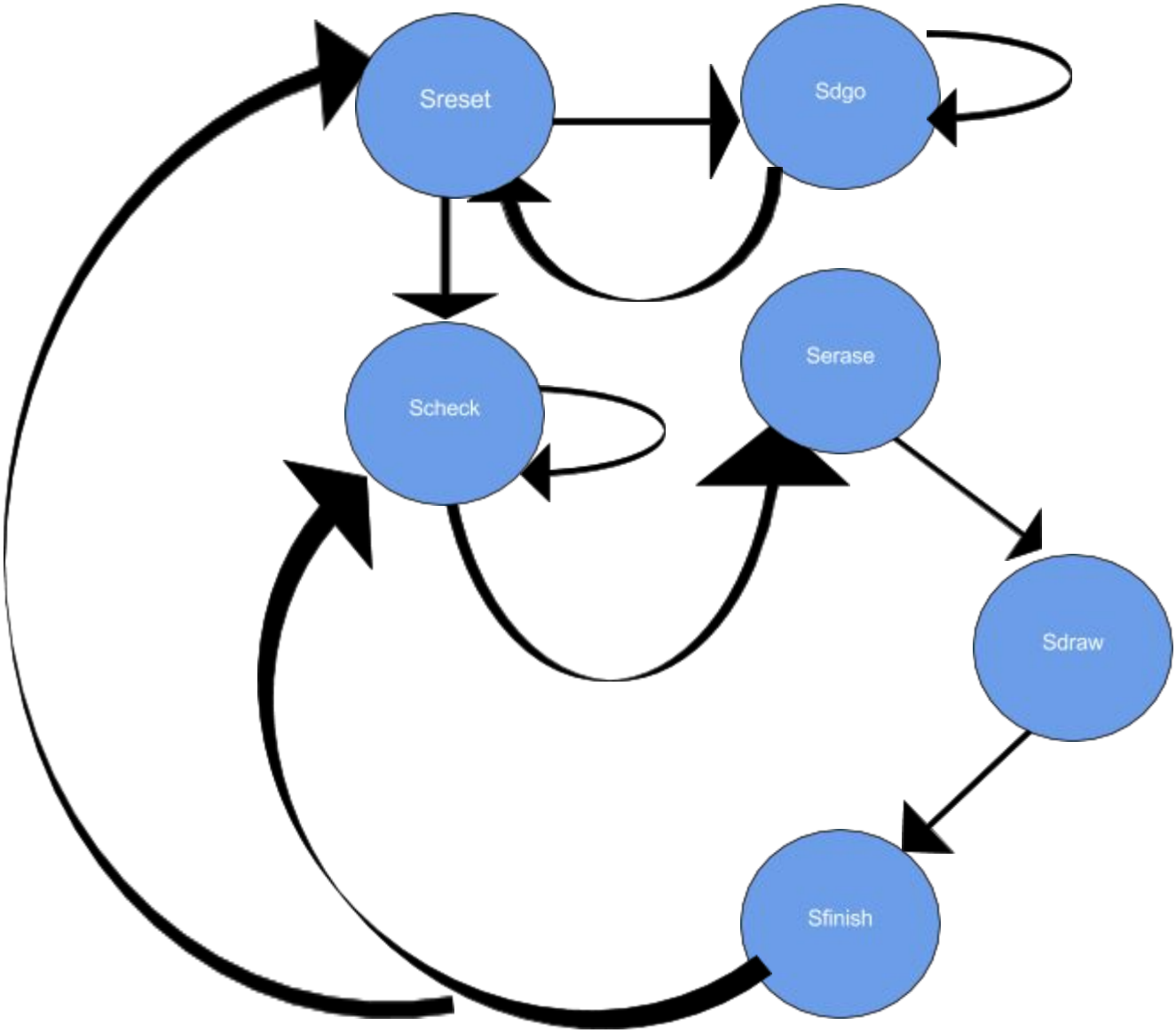

Figure 3: Draw finite state machine

Report on Success

Overall the project worked as we had planned. However, some components did not work due to time management issues. The components are described below:

Tile X position randomization

At the beginning of the project, we spent most of our time attempting to create a good randomizer using a Linear Feedback Shift Register using 3 D flip flops and one XOR gate to produce a number from ranging across 5 values. We then used case statements to assign each value to an X coordinate that corresponds to a lane position. This however did not work very well as values were still predictable. I think a better approach to creating a better randomizer was to create an LFSM with more D flip flops and XOR gates then pick 3 arbitrary bits to represent the X coordinates. This method results in a more random selection of values.

Outputting specified notes

For our audio portion of our project, we used the Audio Demo's implementation of the audio for our game without fully understanding the module's functionality. To generate a note we created an edited version of a clock by changing the delay value, thus changing the frequency of the square wave.Using case statements we tried to implement this idea of changing the delay value, but it did not really work.I think a better understating of the audio module would have made it much easier to properly implement sound into our game. So we just made a small adjustment to the available audio module to implement it into our code.

What would you do differently

If we were to take on this project again, there are many things we would do differently. This project taught us that it is much more difficult to debug a hardware project than it is to debug a software project. Thus, if we did this project again, we would work on smaller components and test them thoroughly before building on bigger parts. If the smaller components work perfectly, bigger parts that depend on them would be easier to build. During the first week, we tried to build big components without fully testing if the smaller components worked. This caused a lot of error and wasted time. Furthermore, we would also use ModelSim more effectively as a debugging tool.

Additionally, this project taught us the importance of time management. Under a three week project, it is important to work diligently and meet all internally set deadlines. During this project, we felt we did not use our time in the first week effectively as we wasted a lot of time trying to figure out how to store and retrieve memory from the on chip memory. To manage our time better, we should have set daily goals instead of weekly goals. This would allow us to break a bigger problem into a smaller and more manageable problem.

Lastly, if we did this project again, we would try to make the game more user friendly. We would implement better control such as input from a keyboard. We would also display the score and high score on the screen. We would also display the time remaining in the game on the screen.