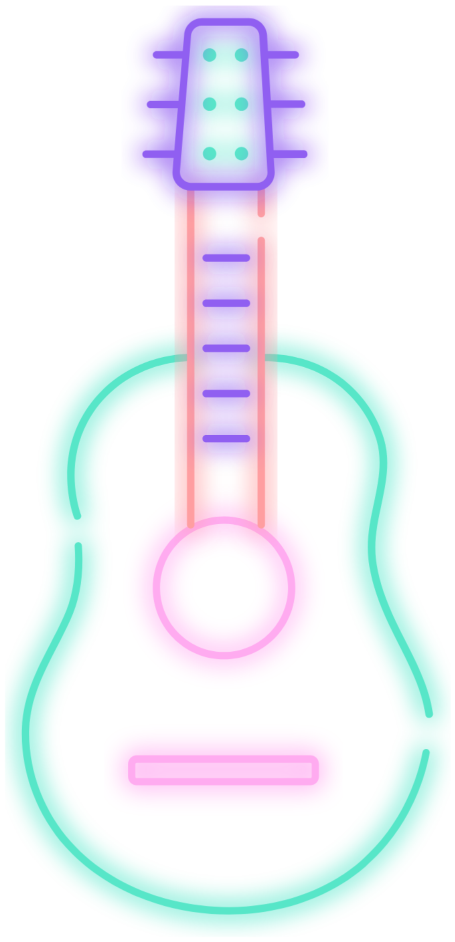
DLD (Digital Logic & Design) PROJECT REPORT

Project Name: TURN UP THE GUITAR



TEAM: MILOGANG

Members (chronologically): Bismaa Behlim, Iqra Azfar, Rabia Shahab, Zoya Suleman

Introduction

Our final project aimed to create a user-friendly PC version of the popular mobile phone game called "Piano Tiles". We call this game "Turn up the Guitar". This idea was chosen because all the digital design concepts that we have learned in the course could be effectively applied to tie together the different components that the game requires, and further enhanced by our own intuition to make the game fully functional. The idea of our game is to let players match musical notes that scroll on-screen at the correct instant, by using the colored buttons in front of them. Consider the buttons to be the guitar strings for the player, wherein he/she is trying to play the musical notes as they are generated by the computer in order to score points, and become a rockstar!

How does the game work?

The game starts as a grid made up of squares. Each column of the grid will be designated one color, owing to the color of the blocks that appear in that column. For instance, the first

Column 1: RED

Column 1: YELLOW

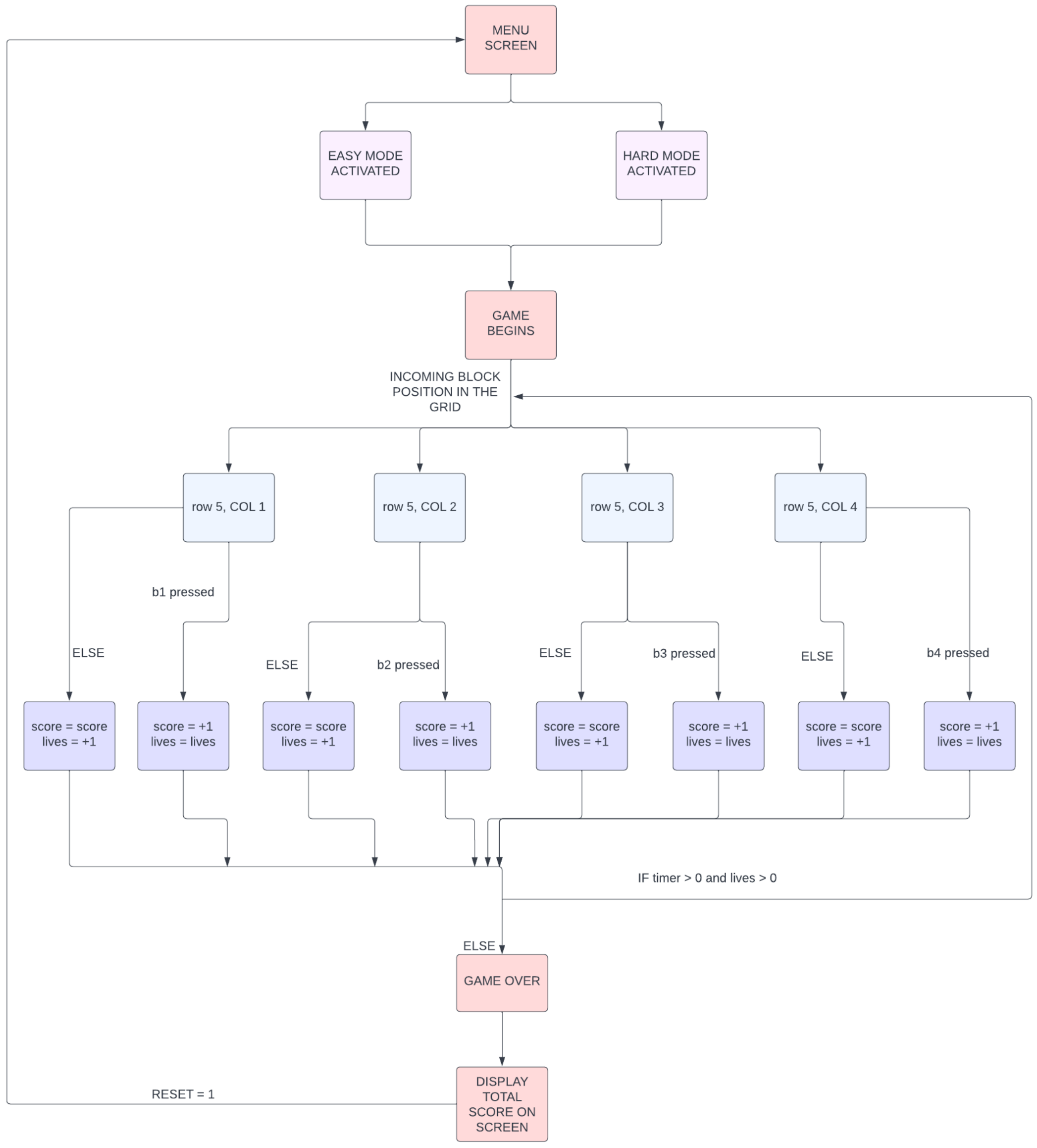
Column 3: BLUE

Column 4: GREEN

When the player presses a button, that button's color shows up on the last row of the grid (let's call this row A) in its respective color column. A colored block appears at the top of the rail, and moves sequentially, block by block, down the rail, toward row A. The user (you) tries to coincide as many input blocks with the system-generated blocks.

Every time the user succeeds in aligning their input with the incoming blocks in the last row, they gain a point. The score is being counted on the 7-segment display and a certain music plays as the game progresses. If the user misses, he/she loses a life. After four misses, the game ends, and the total score is displayed on the screen.The duration of each iteration of the game is 3 minutes, after which the game ends.

User Flow Diagram:



Implementation of the game:

System Overview:

The game uses **Verilog** as the hardware description language and is divided into major modules, which are as follows.

The first module is our **Input Module** which contains 4 external buttons for the user input and handling, as well as for choosing to enter easy and hard mode according to the user's preference.

Adding on, the second and most important module is our **master module** which is implemented with the FPGA Basys-3 board serving as a central hub of the system and it is responsible for controlling the flow of the game. It is used for the implementation of the seven-segment display to keep the count of the score, give the timer countdown for the game, and is also the means of programming the display unit of our menu and game screens.

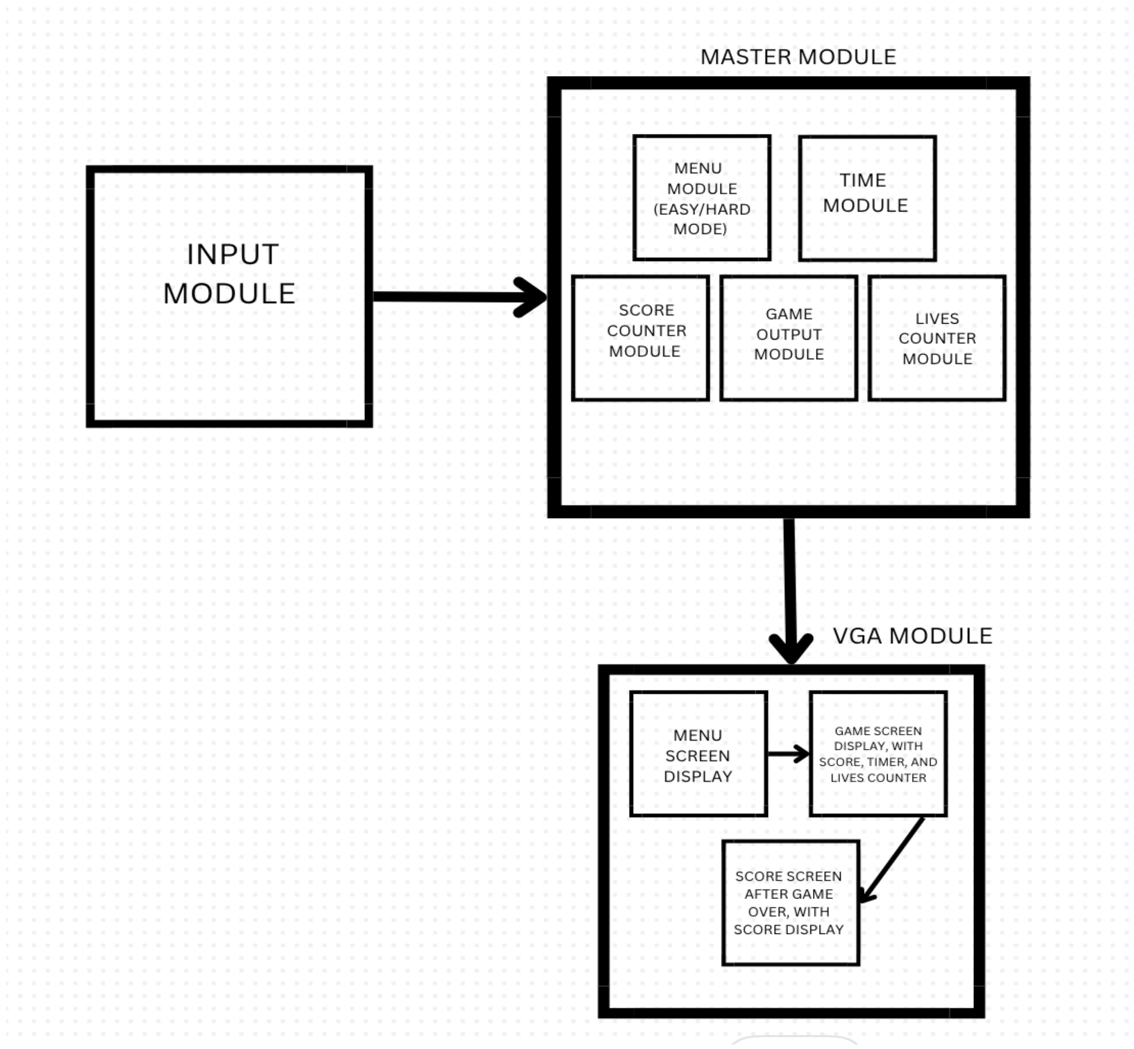
Lastly, our game also contains the **VGA module**, just as important as the master module, wherein the interface of the menu and game screens of our game are displayed on the monitor via the FPGA.

Our game fundamentally follows the **Mealy FSM**, since our output display, which is the score screen, is determined by the user's inputs and the present state, which in our case are the falling blocks on the game screen.

Since our game requires multiple modules, which need to function individually first before the entire system is amalgamated together, we have divided the tasks according to the following rubric:

- Bismaa Behlim: Input Module and Score Module(partial)
- Zoya Suleman and Iqra Azfar: Master Module(Menu Module), Timer Module, Score Module(partial), lives module
- Rabia Shahab: Master Module(Game Screen) i.e movement of incoming blocks

Block diagram representing FSM factoring and division:


MASTER MODULE
INPUT MODULE
MENU MODULE (EASY/HARD MODE)
TIME MODULE
SCORE COUNTER MODULE
GAME OUTPUT MODULE
LIVES COUNTER MODULE
VGA MODULE
MENU SCREEN DISPLAY
GAME SCREEN DISPLAY, WITH SCORE, TIMER, AND LIVES COUNTER
SCORE SCREEN AFTER GAME OVER, WITH SCORE DISPLAY

Input Module:

The game takes the player's input through external push buttons. Adding on, four external buttons are assigned each color respectively,So when a button is pushed, that button's color shows up on the last row of the game's screen and helps the player to match the color of the falling block with it.

Explanation:

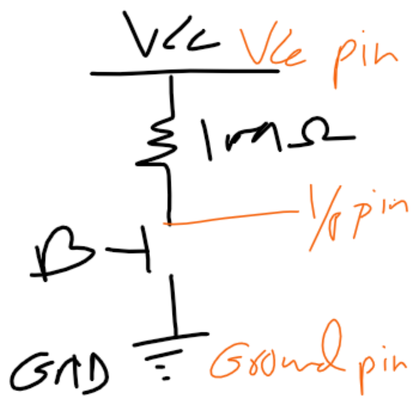
The primary function of each button is to allow the user to play the game through matching tiles. The secondary functions are as follows:

- B1=Red: Resets game after it is over
- B2=Green: Starts easy mode of game

- B3=Blue: Starts hard mode of game
- B4=Yellow: No secondary function

Implementation:

The buttons are arranged on a breadboard with each button having a painted wooden structure on top to make a more easy-to-understand user interface. The buttons are wired as follows:

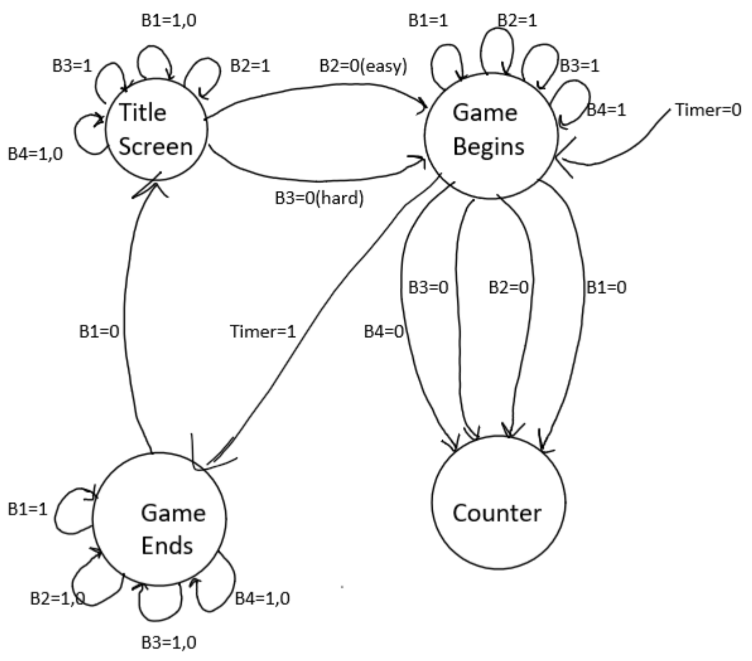


All buttons will share the same Vcc and GND pin , provided by the JA PMOD of the Basys-3 FPGA. The Input pins will be as follows:

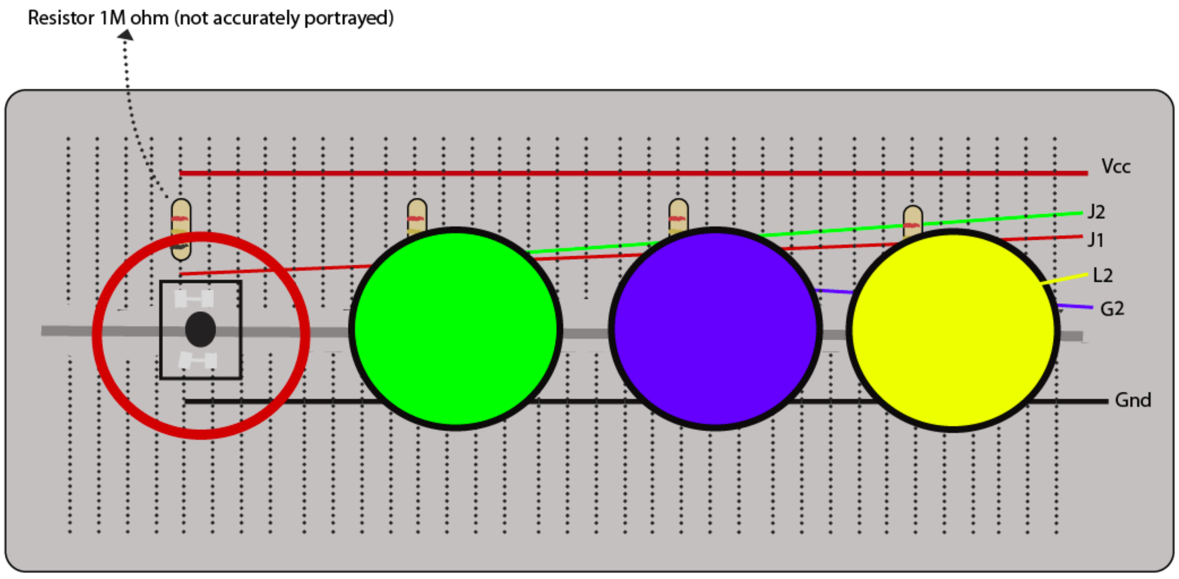
- Red: JA1:J1
- Yellow: JA2:L2
- Green: JA3:J2
- Blue: JA4:G2

The hardware is implemented as follows: Each button is connected to a pull-up resistor of 1M ohms, with the input pin connected right after the resistor. When a button is not pressed, there is some voltage V being supplied to the i/p pin. When the button is pressed, the connection is complete and the i/p pin voltage reduces to 0V. Thus, the whole system is being implemented in active low configuration. The code is a simple conditional statement where i/p pin HIGH gives no signal and i/p pin LOW gives signal.

State Diagram:



Visualization Diagram:



The Master Module:(FPGA)

The master module is further divided into the following modules:

1) Menu Screen module

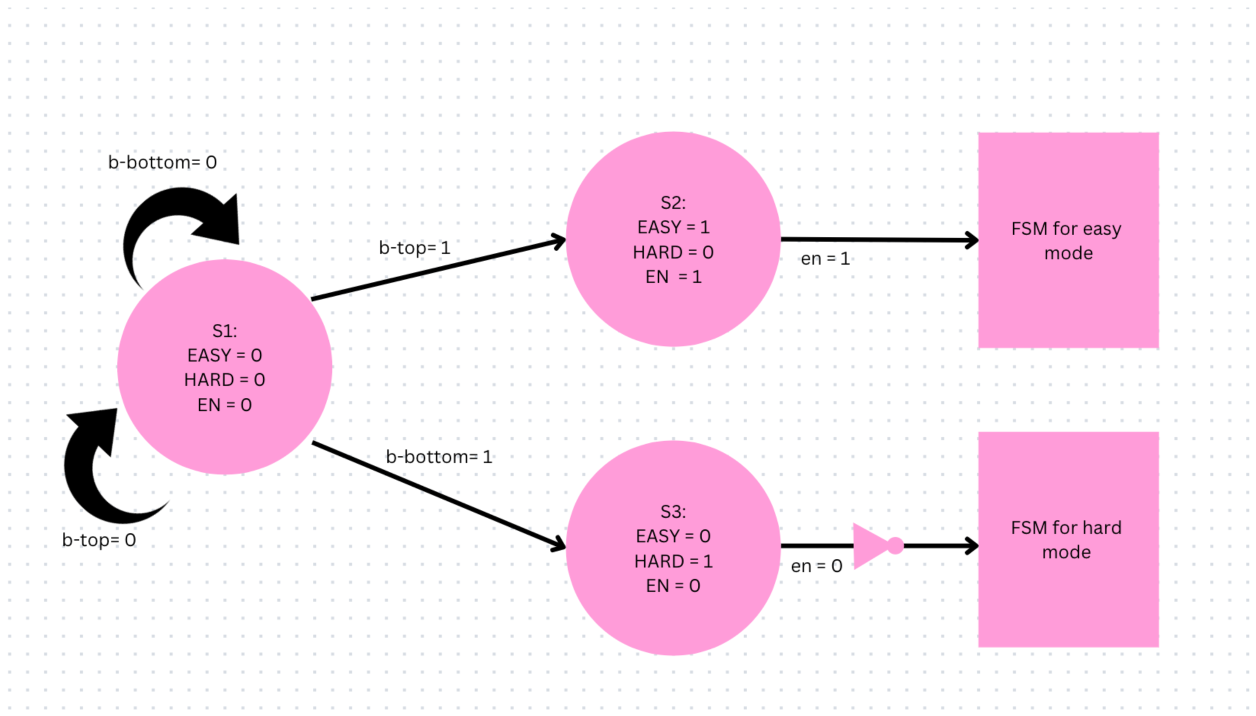
<u>Explanation:</u>

The title screen is the first screen that appears on the screen once the fpga is connected. We have implemented the pixel generator and VGA syncing module to generate a colorful screen, wherein we display two options to the user:

- Easy mode (letter E at the top)
- Hard mode (letter H at the bottom)

The user makes these selections via 2 buttons on the FPGA board. We plan to use the top and button buttons of the FPGA to correspond to the top and bottom options on the menu screen respectively.

<u>Menu Screen State Diagram:</u>



State assignments in the above state diagram for Menu screen FSM:

Easy refers to easy mode whereby the blocks generated by our random block generator will come onto the screen with a given speed. Hard refers to hard mode whereby the blocks generated by our random block generator will come onto the screen with relatively faster speed than easy mode.
1 refers to high/on, 0 refers to low/off
b-top refers to the top fpga button, b-botton refers to the bottom fpga button
En refers to the enable pin of the FSMs after the menu screen.

For this particular FSM:
- Easy and hard are my states
- b-top and b-bottom are my inputs
- En is my output

2) Game Screen Module

Explanation:

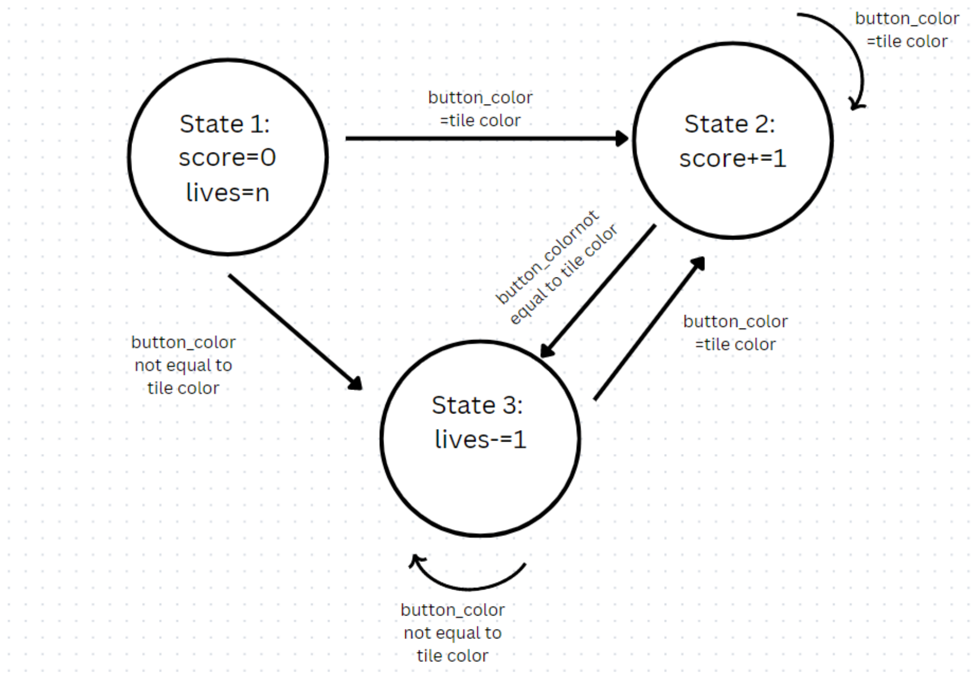
Once a user selects the level of difficulty they want to play at from the menu screen, they are taken to the game screen. The screen would display a timer and 5 by 4 grid, each column would represent the tile color and upon their chosen difficulty, the tiles would drop at the difficulty screen across their column.

As soon as the tile reaches the last row, the user would have to press the corresponding button in time(the 4 buttons will be assigned for each column). If done so, the user's point increases by 1; else, the user's lives will decrease by 1. As mentioned above, an FPGA Basys-3 board would be connected that would increment the score of the user as the correct button is pushed at the right time for the corresponding tile.

Game Screen Output Module

The output is controlled using the FSM displayed below. The change in state depend on the user input and the score display is shown on the FPGA board, it is controlled by the modules that are shown in the score block diagram below.

<u>Game Screen Output State Diagram:</u>



3) Lives Counter Module:

<u>Explanation:</u>

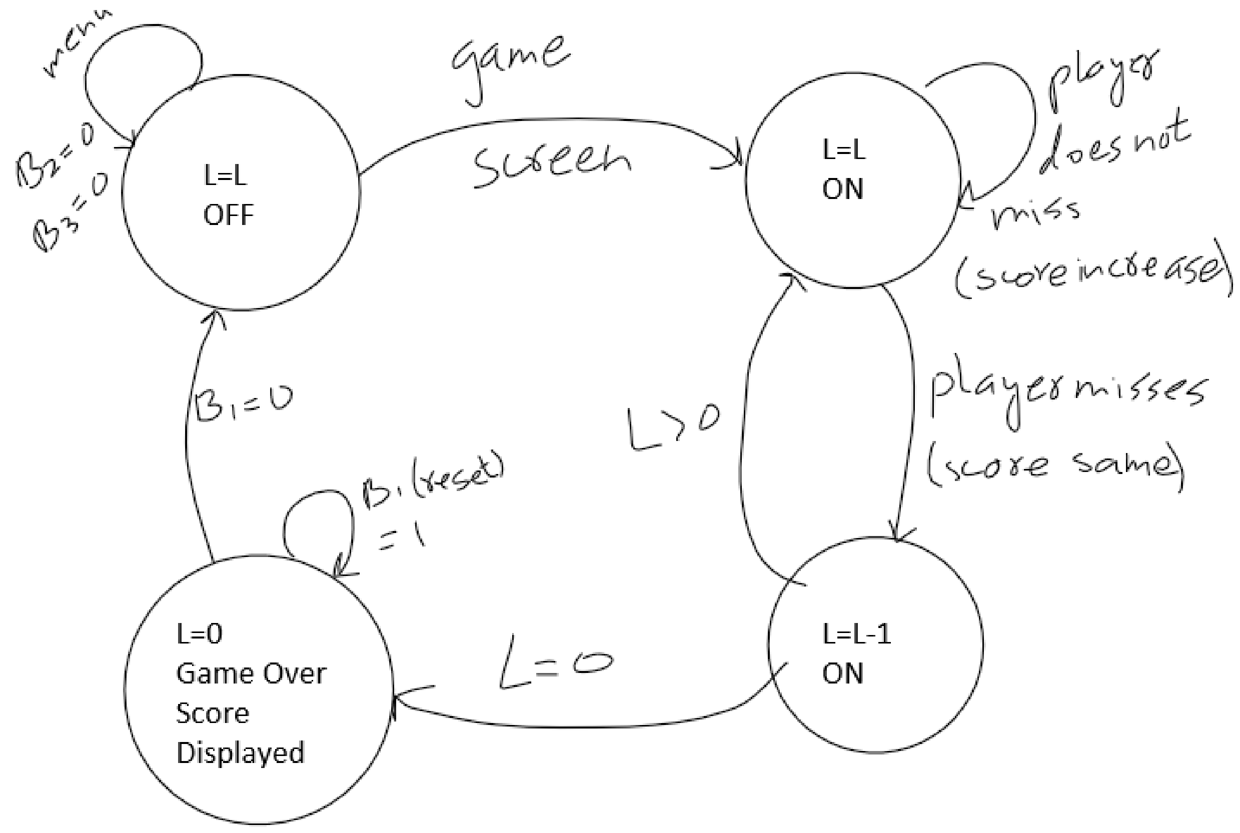
The lives module fundamentally has the same functionality as the countdown timer described above. The difference is that the lives counter will have the initial value 4, and the number of lives are reduced by 1 only if the player presses a button at the wrong time. For example:

- Case 1: Random block in row 5 column 1, user pressed button 1 at that instant -> lives remain as it is
- Case 2: In any other case, the number of lives decreases by 1 if the wrong button is pressed or the correct button is pressed at the wrong time.

So case 2, basically acts as the trigger or the enable pin to activate the deduction of lives while the game is being played.

When the number of lives becomes zero, the game is terminated and the total score gained by the player is displayed on the screen.

FSM State Diagram:



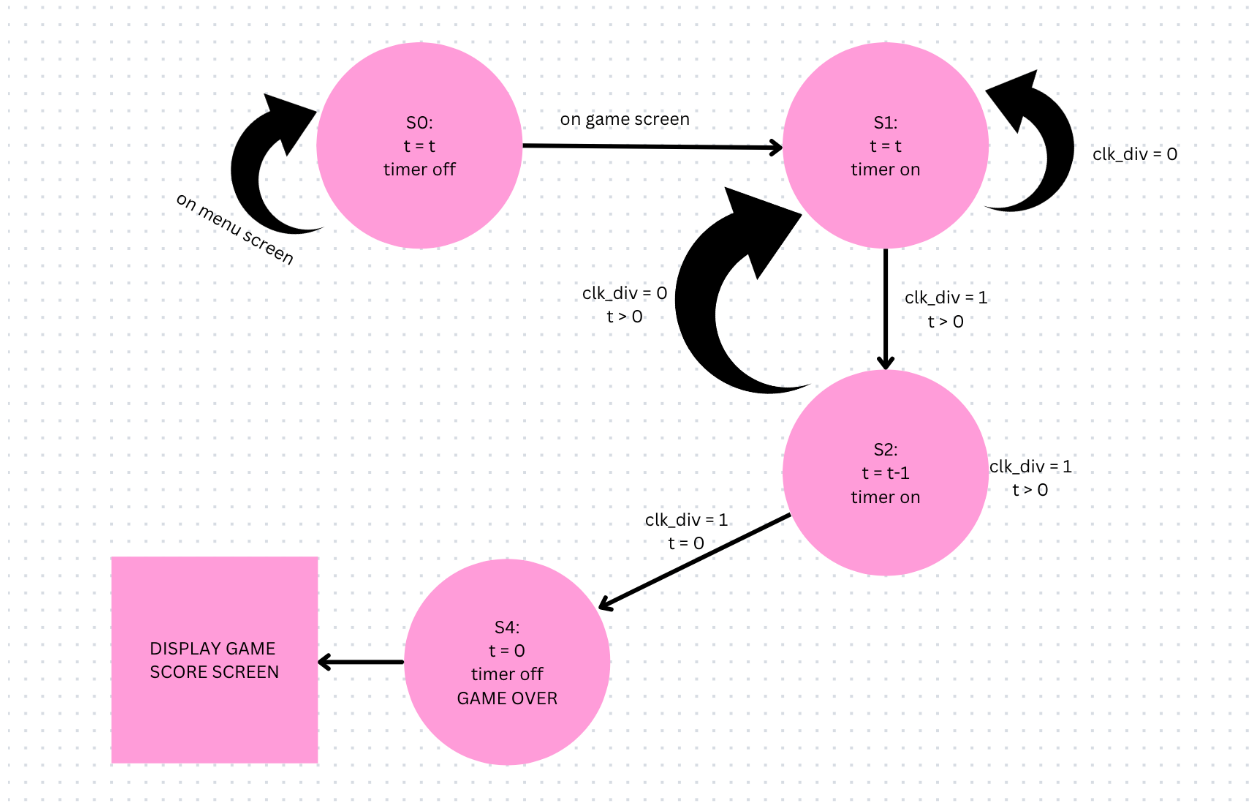
4) Timer Module:

Explanation:

The duration of our game is 3 minutes as mentioned above. We aim to implement a counter, which operates in the reverse direction. This means that instead of an increasing counter, we will have a decreasing counter from starting at 180 (3 minutes = 180 seconds).

We will use a clock divider module (similar to the one we designed in our labs), to reduce the frequency of the clock signal to 1Hz, so that our clock signal is in synchronized with the seconds in a regular clock. At every clock signal of the update clock, the counter reduces by 1. This continues until the counter eventually reaches zero, at which point, our system will terminate the game and display the player's total earned score on the screen.

Input state diagram:



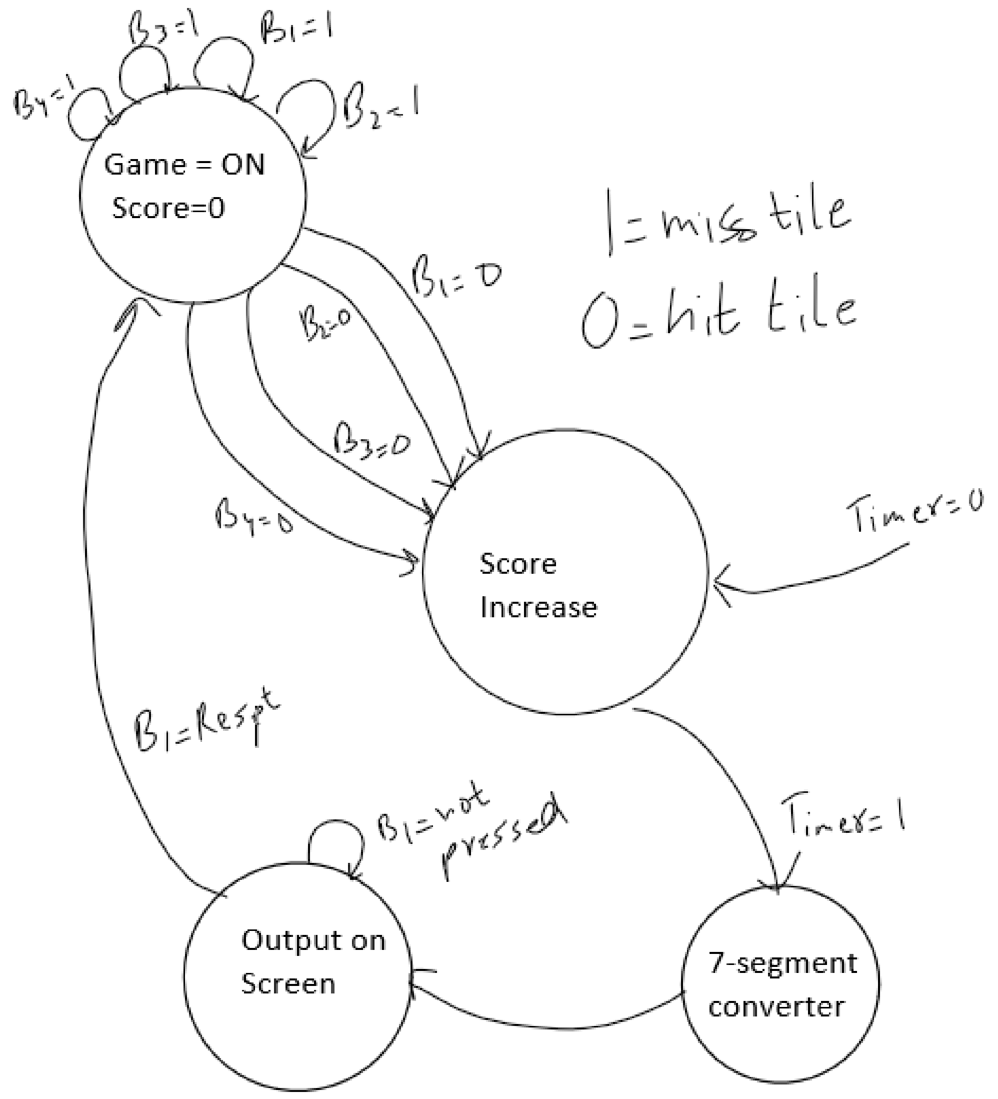
5) Score Counter Module:

Explanation:

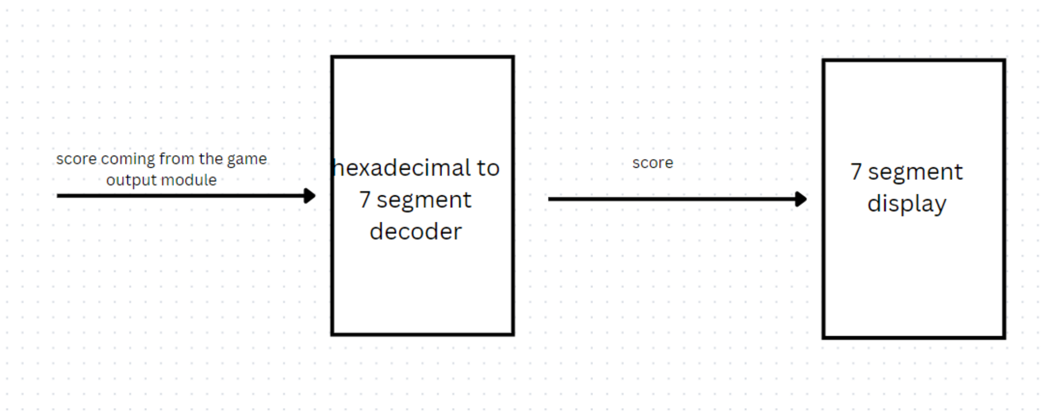
As the user presses the correct button corresponding with the incoming block in the respective position, the score should increment by 1 and the updated score will be displayed on the FPGA board. There is no deduction of score in a case where the player misses the block. The modules needed to do so are shown in the block diagram below.

The counter module will keep track of the score every time a button pressed matches the incoming tile. Its maximum value will be 250. After the timer ends, the value stored in the counter will be displayed on the screen. When the game resets, the counter will also reset.

FSM State Diagram:



The modules called/instantiated for the score counter will be:

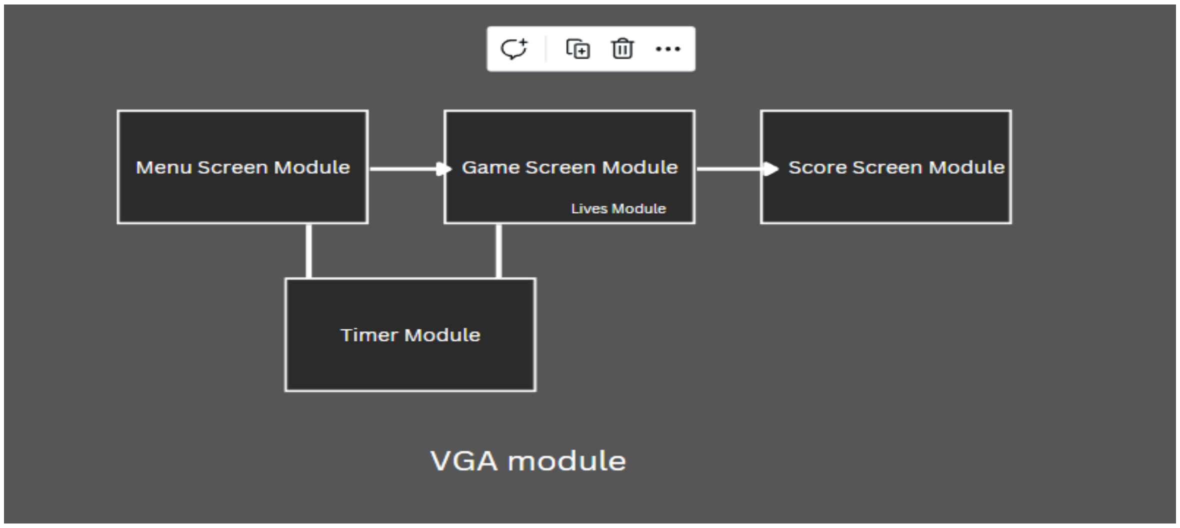


score coming from the game output module
hexadecimal to 7 segment decoder
score
7 segment display

VGA Module:

Pixels are displayed on the video screen using the VGA. It calculates 640 pixels horizontally across our screen using a counter, then there is a pause while the cathode ray tube moves to the next row from the top of the screen. The cathode ray tube then resets to the upper left-hand corner of the screen during the vertical blank, which lasts until all 480 rows have been drawn. The clock signal (clk) is taken in by the VGA module, which then displays the horizontal and vertical positions of each pixel. The **Menu Screen Module**, **Game Screen Module,** and **Score Screen Module** receive these location signals in the form of v count and h-count, which it uses to assign the proper color values to each pixel.

Block diagram showing the functionality of the VGA module:

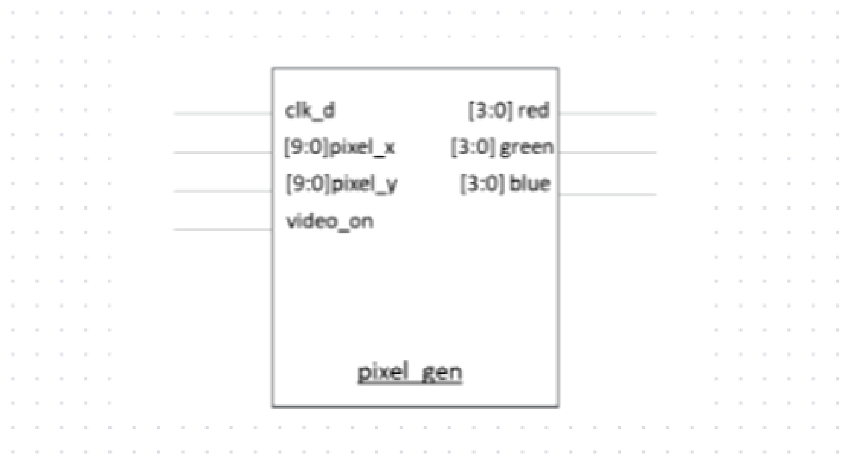


Pixel Generator:

Explanation:

The pixel generator module is of great significance in our game, and is being used at many instances in our VGA module. It will be used to display the user an output screen where they can see the tiles appearing. The module does this by generating the 4-bit color signals- RGB signal. Each coordinate of the pixel (x_location, y_location) will be given this RGB signal so they know what color to display for that specific pixel. The module is show below:

Module Diagram:



We have used the pixel generator module in displaying all the different screens of our game (menu, main game screen, score screen)

Conclusion

In conclusion, the aim of our project Turn Up the Guitar is to learn about digital hardware design and be able to implement it into an entertaining game for all ages. We are using our knowledge obtained throughout the course of using breadboards, Vivado, FPGA Basys-3 board, and many other concepts to implement this project. For our project, we have successfully found a way to take user input through buttons and are currently working on creating a GUI for this game. The next step will be to display the score and a counter and to integrate the input output modules together. We aim to design a project that can be easily tested and works successfully without any errors whatsoever. We are enthusiastic to see what challenges may pop up in the future and the creative ways we can find to solve them and continue our game implementation.