**Habib University**

****

**Dhanani School Of Science and Engineering**

**Digital Logic and Design**

**EE/CS 172/130**

**T3**

**(Milestone 3)**

**گولی مار**

(A Shooting game)

Hijab Fatima Siddiqui (hs09195) Faiza Salman Khatri (fs09192)

Iffat Fatima (if09270) Vishal Raj Sundrani (vs09221)

Professor Areeba Rajput

11th May 2024

**Table Of Contents:**

1. **Abstract**

Introduction

1. **Implementation**

2.1.InputBlock

2.2 Output block

1. **User Flow Diagram**
2. **Control Block**

1. Start Screen

2. GameScreen

3. EndScreen (if you win)

4. EndScreen (if you lose)

1. **FSM**
2. **Major Challenges Faced**
3. **GitHub Link**

**Abstract**

This project presents a shooting game designed for the Basys3 FPGA board, leveraging joystick input to maneuver a red dot on the screen and a button to shoot the characters. Players navigate through a dynamic environment featuring two character types: enemies and friends. Shooting enemies earns points, while mistakenly shooting friends makes you lose the game. The VGA output displays real-time gameplay and score progression, with victory achieved upon reaching a score of 9. This succinctly engineered game offers an engaging experience that challenges players' strategic thinking and reflexes.

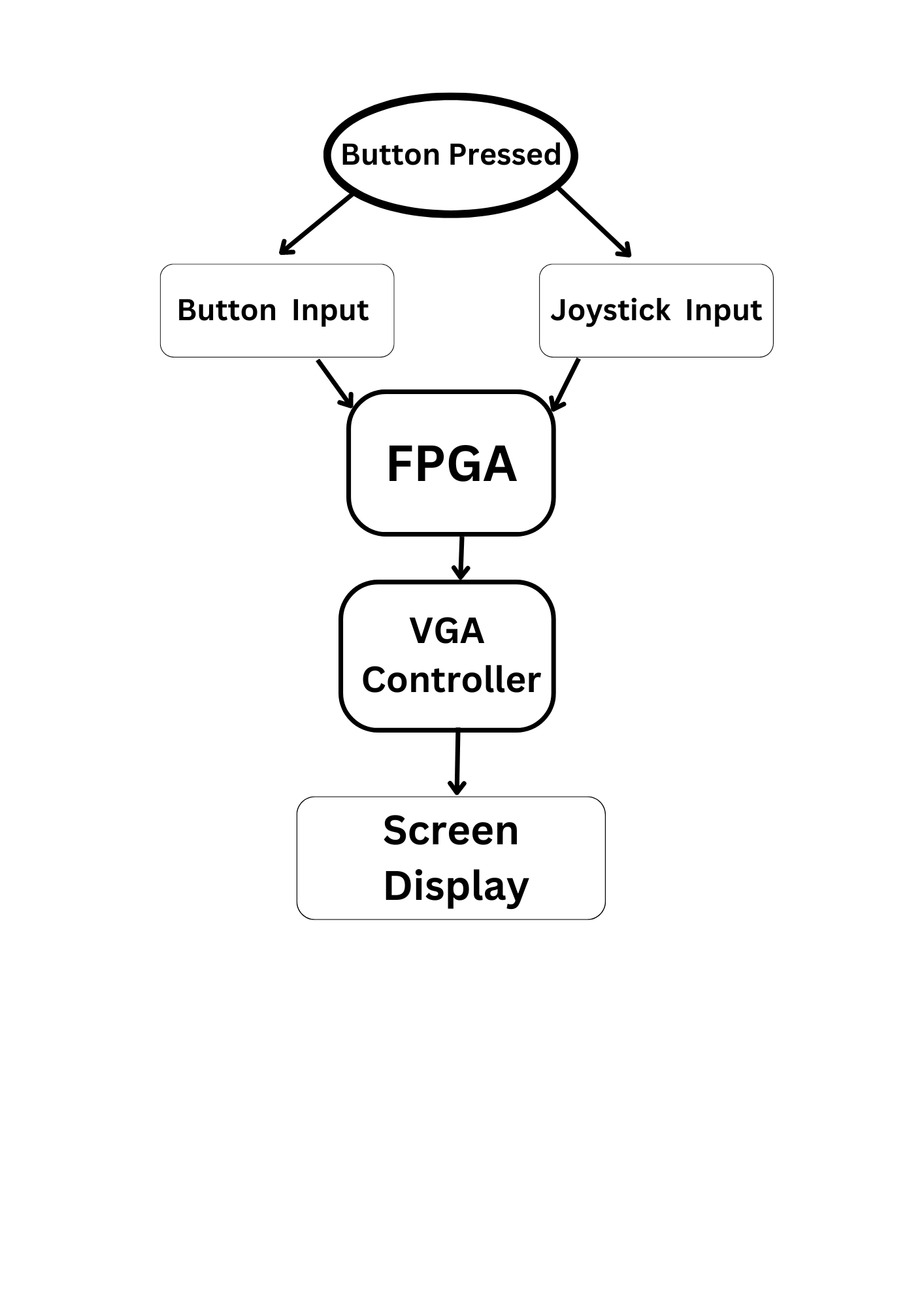
1. **Introduction:**

Welcome to the immersive world of our FPGA-based shooting game. As players engage with the Basys3 FPGA board, they are greeted with a sleek default screen displaying the game's title. Upon initiating gameplay by pressing the designated button, the digital arena comes to life, with characters dynamically appearing on screen. At the game's outset, the player's score stands at zero, poised to escalate with each strategic maneuver.

Central to the gameplay experience is the intuitive joystick interface, seamlessly integrated with the Basys3 FPGA board. Players wield the joystick to command a red dot across the screen, tactically positioning it amidst the ever-changing landscape of friend and foe. A press of the button while targeting an enemy culminates in a satisfying defeat, duly rewarded with a point increment, and targeting a friend makes you lose.

For Joystick FPGA’s jxadc ports are used to seamlessly translate joystick movements into fluid red dot navigation. The button used the U18 port. This meticulous engineering ensures a responsive and immersive gaming experience, where every twitch of the joystick translates into decisive on-screen action.

1. **Implementation:**



**2.1 Input Block:**

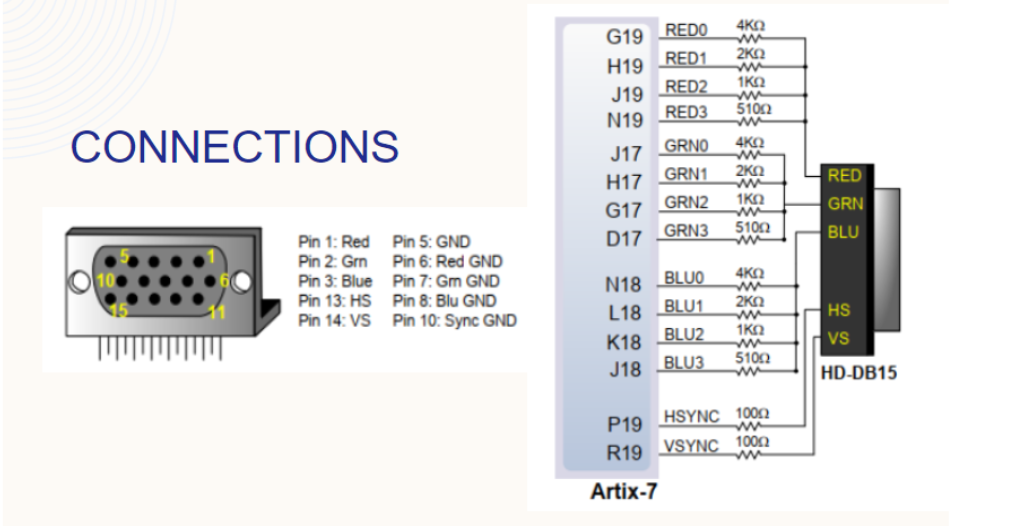
| **Input** | **Functionalities** |
| --- | --- |
| FPGA Button U18, T18 | U18 Button to attack and to start game.  T18 Button to reset. |
| Joystick jxadc | To manage the target |

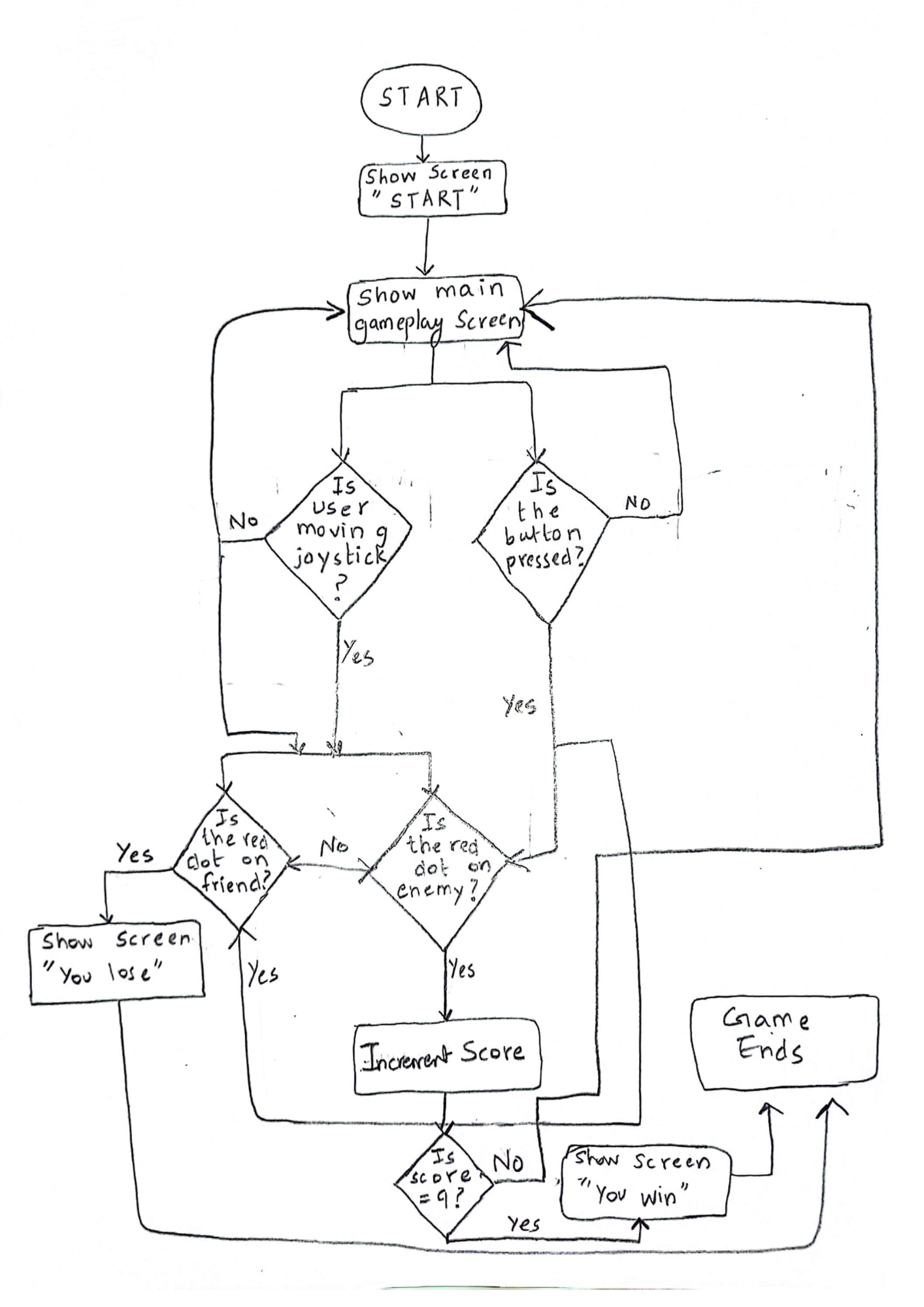
**2.2 Display Screen (Output Block):**

The display screen is generated using the VGA (video graphic array) connector. The display which we are using is standard 640 x 480 pixels.

For imaging on the horizontal axis, we turn the video on from 0 to 640 pixels, and for the vertical axis the video is on from 0 to 480 pixels as that is the main display, after that, the video is turned off for borders and retracing.

To make the final screen we first used a clock divider to reduce the frequency from 100 MHz to 25 MHz for FPGA to work, then we used h\_counter to count the pixels of the horizontal axis and the v\_counter for the vertical axis. Then Vga\_sync was used for turning the video on on the required part of the screen and finally pixel gen was used to create our desired screen. To control the input clock we use default clk that has a frequency of 100MHz. To control and display the score on the display screen, we used the score module.



**3. User Flow Diagram:**

**4.Control Block:**

The game consists of four main states which are represented by four different screen displays:

1. Start Screen

2. GameScreen

3. EndScreen (if you win)

4. EndScreen (if you lose)

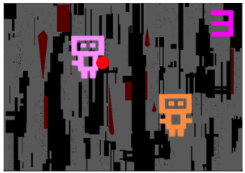
**4.1.Start Screen:**

****

The modules controlling the start screen are clock\_divider, h\_counter, v\_counter, vga\_sync, and pixel\_gen.

State Transition: The start screen appears when the control block has the starting state. The player is supposed to press the start button which will result in the transition from the start screen to the game screen.

**4.2.Game Screen:**

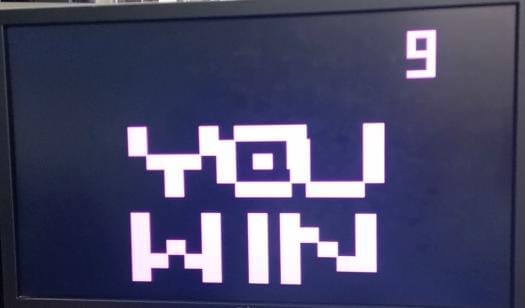
****

This module is controlled byclock\_divider, h\_counter, v\_counter, vga\_sync, background module, animation\_module, score module, and joystick module.

After the start button is pressed, this screen appears.

State Transition: If the score is 9, the user wins and the end screen “you win” appears. If the user hits a friend, the game is over and the end screen “game over” appears.

**4.3.EndScreen (if you win):**

****

The modules controlling the end screen are clock\_divider, h\_counter, v\_counter, vga\_sync, pixel\_gen, animation\_module, and score module.

This screen appears if the score is 9 and the user did not hit any friend.

State Transition: The reset button is pressed and the start screen appears.

**4.4.End Screen (if you lose):**

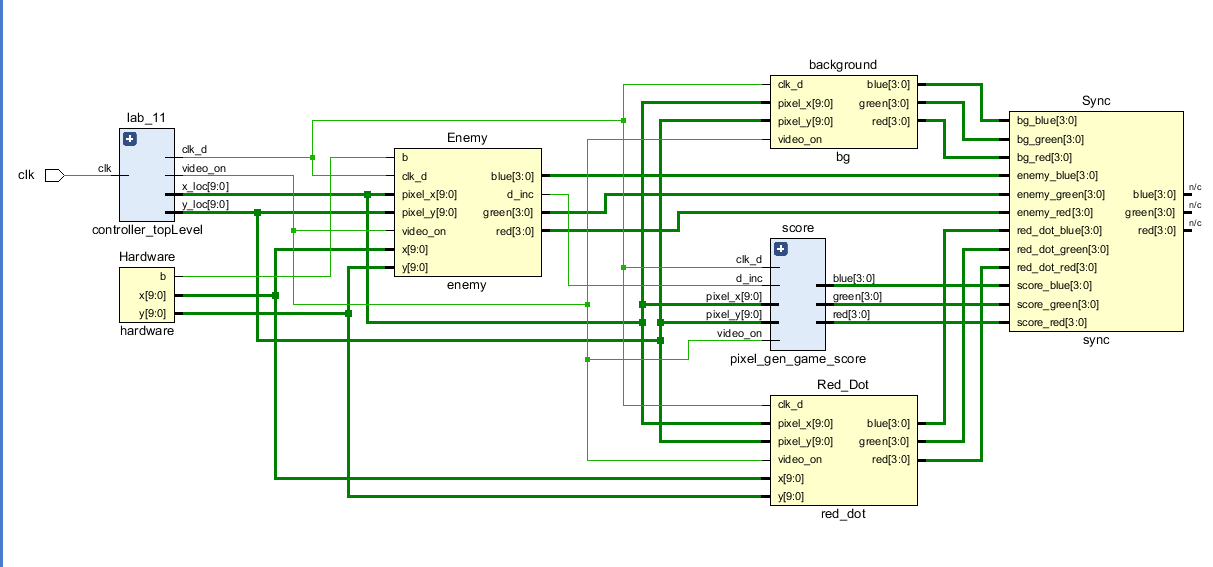
****

The modules controlling the end screen are clock\_divider, h\_counter, v\_counter, vga\_sync, pixel\_gen, animation\_module, and score module.

If a user hits a friend, this screen appears.

State Transition: The reset button is pressed and the start screen appears.

**Top module:**

****

**5. FSM**

The start and end states of the game, dictated by the start game and end game screens, take the push buttons of the FPGA and joystick as inputs for state transitions. Furthermore, our main game itself gives outputs based on any kind of input from the user (the next attack user does on the enemy or friend) or instead terminates automatically when the friend is hit, moving to the “Game Over” screen. This output can be seen as dependent on the current state and current inputs hence, we can conclude that the FSM implemented in our game is a Mealy Machine as the output of our game is dependent on the input block and current states.

**6. Major Challenges Faced:**

The major challenges faced in making the project were as follows:

1) Implementation of a joystick: It was very hard for us to integrate the joystick within our project using the breadboard and connecting it with the FPGA.

2) Writing the code for Animation: We had no experience in writing the verilog code for creating screens with animation and random movement of the characters.

3) Score counter: Creating the score counter was harder than it seemed. We had to keep incrementing it as the user attacks enemies and the score should appear on screen.

4)Creating the top module: The hardest part of our project was integrating all the modules that every member had created in the top module. To manage all the input output ports, to make every module compatible with one another, and to display the game play screen with all modules integrated in it.

**7.GitHub Link of our project:**

<https://github.com/vishalrajsundrani/goli_maar>