ES204 Digital Systems

Project Overview and Objectives

March 3, 2024

Project Name: '*Dodge Through Bars*'

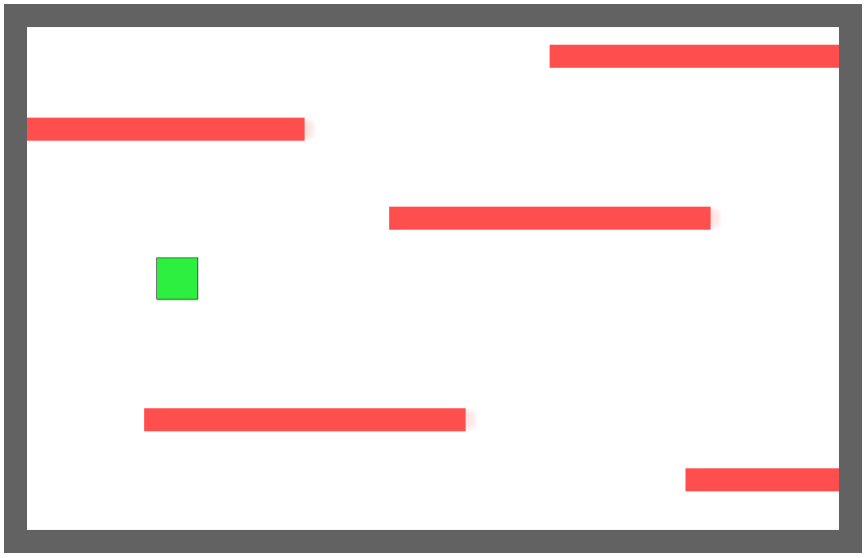

Fig.: 1

Group Details:

1. Vipul Sunil Patil (22110189)
2. Raj Patel (22110214)
3. Ranit Biswas (22110217)
4. Shreel Chawla (22110243)

Project Overview:

In *'Dodge Through Bars,'* our project aims to develop an engaging arcade-style game using the Basys 3 FPGA board. Players can control a block character through a series of upcoming obstacles, represented as rectangle bars on the screen. The objective is to avoid colliding with these bars while accumulating points to navigate the obstacles successfully.

We will implement the game logic through Verilog programming, including player movement, obstacle generation, and the controlling system. Using the Basys 3 board's buttons, players will be able to control the block character's movement on the up-down axis.

We will utilize VGA communication to display the game graphics on an external monitor, making the game '*Dodge Through Bars'* playable.

This project will provide hands-on learning in digital logic, Verilog, and FPGA programming and real-time graphics for entertainment and educational purposes.

1. About The Game

1.1 Playable Boundary:

The game boundary screen is visually represented as a rectangular display monitor area delineated by distinct borders or edges. Within this defined closed area, players can observe the movement of the block character and obstacles as they interact with the game environment. The player cannot move the block character outside this boundary.

Inside the boundary, the player will aim to move the block up and down to dodge the obstacles coming towards it and save from any collision to gain a point.

1.2 The Block Character:

The block character is the player's avatar in *'Dodge Through Bars,'* representing their presence in the game world. It is typically depicted as a solid-coloured square against the background screen, with distinct visual characteristics distinguishing it from other in-game elements. Despite its minimalistic appearance, the block character signifies the player's focal point of interaction within the game environment.

Player control of the block character is achieved using the directional buttons on the FPGA board. Players can navigate the block character vertically across the 'playable boundary' on screen, dodging obstacles and manoeuvring through narrow gaps to progress further in the game.

1.3 The Obstacle Character:

The obstacle character represents the challenges players must navigate to progress in the game. It appears as blocks that move horizontally across the screen from the right side towards the left. The obstacle character is visually depicted as solid-coloured rectangular blocks against the background screen.

The obstacle character moves continuously from the right of the screen to the left at a defined speed. The movement of the obstacles adds unpredictability to the gameplay.

1.4 Control Buttons:

Players can control the movement of the block character using two of the five push buttons available on the Basys 3 board. As mentioned, the upper and bottom push buttons will vertically navigate the block character across the screen. Pressing the upper button will move the block character upwards while pressing the bottom button will move it downwards.

These buttons will be the only input from the player to interact with the game.

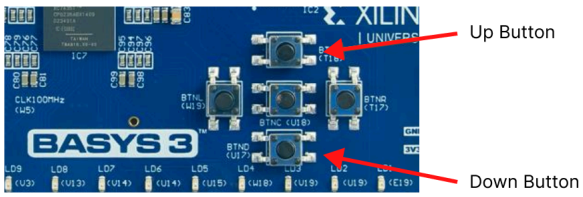

Fig.: 2

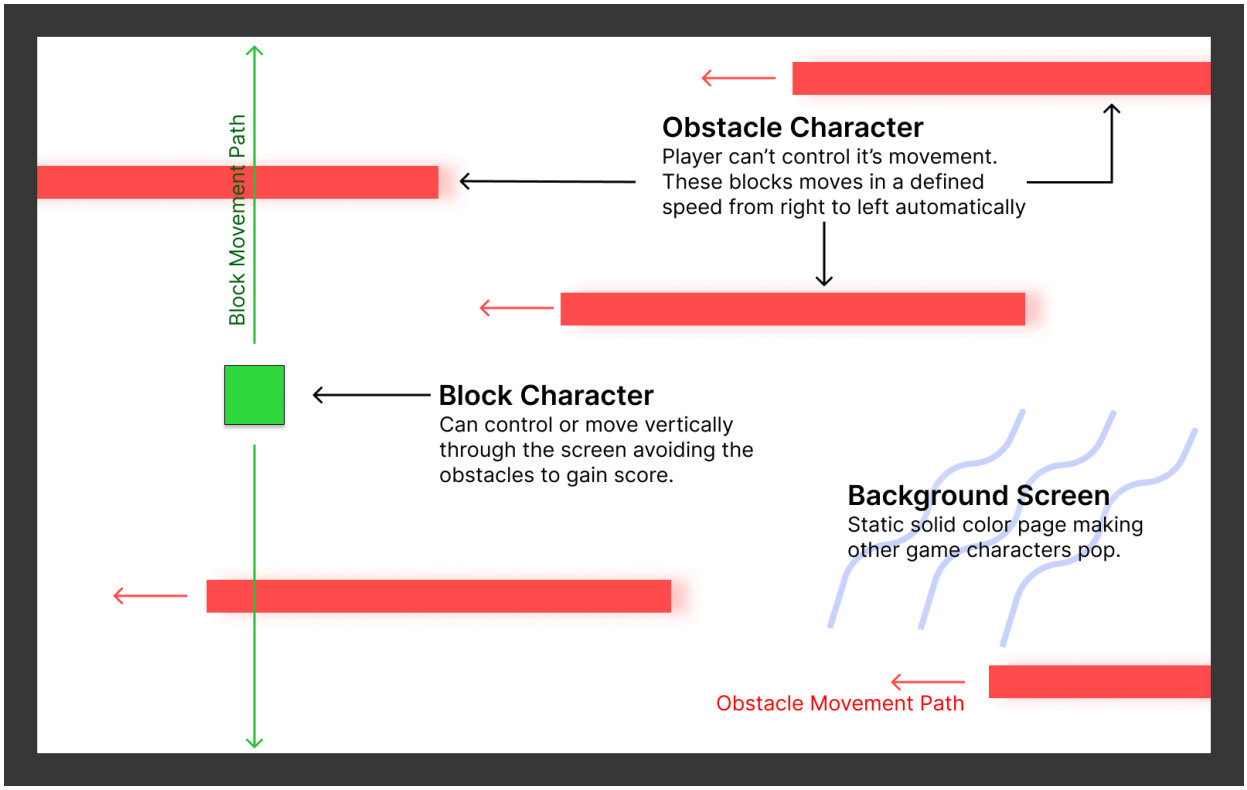

Fig.: 3

2. Game Mechanisms & Logics:

2.1 Visual Output (Display):

VGA communication will be utilized to display the game on a monitor. VGA (Video Graphics Array) is a standard for displaying video signals on computer monitors. Verilog modules will be created to generate VGA signals, including horizontal sync, vertical sync, pixel clock, and colour signals (red, green, and blue). The VGA cable will transmit these signals from the Basys 3 board to the monitor. We can create a rectangle by setting the upper and lower ranges on the X and Y axes and giving it a different RGB value to make it distinguishable.

Using the described method, we can display the block and obstacle characters on the screen.

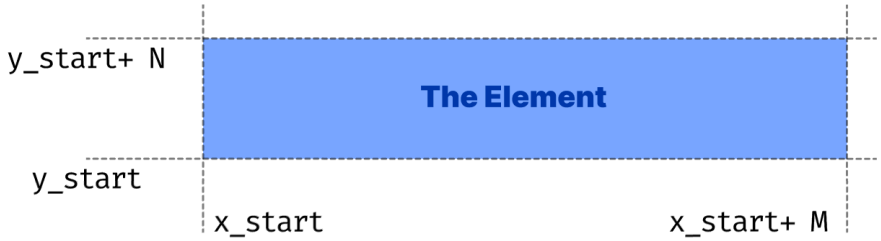

Fig.: 4

2.2 Making Obstacles Move:

Each obstacle initially starts at specific coordinates (figure 4) on the screen, typically towards the right side. To simulate movement from right to left, we decrement the X-coordinate of each obstacle block. By decreasing the X-coordinate value at a predefined speed, we create the illusion of obstacles moving towards the left side of the screen. While Y coordinates will be kept unchanged as our game does not allow obstacles to move up and down,.

2.3 Block Character Controls:

To facilitate responsive vertical movement of the block character (figure 3), we employ a variable, i.e., **`up_ctrl`**, which consists of 2 bits. Initially, **`up_ctrl`** is set to **`2'b00`**, indicating that the block character is stationary.

When the player presses the up button (figure 2), the value of **`up_ctrl`** is set to **`2'b11`**. While up_ctrl remains at **`2'b11`**, the block character's vertical position (Y-coordinate) is incremented by a certain rate, causing it to move upwards on the screen. This mechanism allows players to control the block character's ascent by simply holding down the up button.

Conversely, if the player presses the down button (figure 2), the value of up_ctrl is changed to **`2'b10`**. While **`up_ctrl`** remains at **`2'b10`**, the block character's vertical position is decremented by the same rate, resulting in downward movement. This intuitive control scheme enables players to navigate the block character vertically with ease.

2.4 Counting the Score:

We implement a scoring mechanism that rewards players for successfully navigating the block character through obstacles. Each element in the game, including obstacles and the block character, is assigned a starting X-coordinate (**`x_start`**) and an ending X-coordinate (**`x_end`**), which is calculated as **`(x_start+ M)`**, where **`M`** is a constant.

When the game detects that the ending X-coordinate (**`x_end`**) of an obstacle is less than the starting X-coordinate (**`x_start`**) of the block character, it indicates that there is no further chance of collision between the two elements. In this situation, the score counter variable is increased by 1.

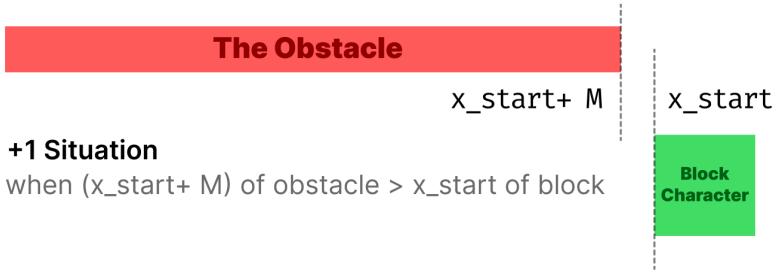

Fig.: 5

2.5 Displaying the Score:

On the Basys3 board's 7-segment display, the player's score will be shown in real-time during gameplay. The score variable will be taken as input to the module, and the current score will be shown.

2.6 Score-Based Level Functionality:

The x-axis speed of obstacles will be adjusted as a function of the player's score to introduce dynamic gameplay progression. As the player's score increases, the speed of obstacles will slightly increase, creating a smoother and more challenging level-up system.

$$speed = f(current\_score) \quad : speed \uparrow \; current\_score \uparrow$$

This adjustment will be implemented by modifying the rate at which the x-coordinate of obstacles decreases based on the player's score. As the score rises, the decrement rate will increase proportionally, causing obstacles to move faster across the screen. This dynamic scaling ensures that the game's difficulty gradually ramps as players achieve higher scores, providing a more engaging and rewarding gameplay experience.

2.7 Game-Over Situation:

In *'Dodge Through Bars'*, the game-over occurs when the block character collides with an obstacle. Collision detection is implemented by continuously monitoring the positions of the block character and obstacles on the screen.

When the game detects that the boundaries of the block character intersect with those of an obstacle, it triggers the game over sequence. This sequence includes displaying a game-over message on the screen, halting further movement of the block character and obstacles, and presenting the player with their final score.

3. Weekly Plan: Four-Week Timeline

1. **Week 1:**
    a. Establish VGA communication between Basys 3 and the monitor. (2.1)
    b. Verify that the display output meets expectations.
    c. Making stationary Block-Characters and Obstacle-Elements on the display.
    d. Integrating the controlling buttons to the system as inputs.
2. **Week 2:**
    a. Enable vertical movement of the Block Character based on player button input. (2.3)
    b. Making the Obstacle Elements appear at the right end of the screen.
    c. Developing Obstacle-Elements Move at a constant speed towards the left of the screen. (2.2)

3. Week 3:

a. Implement score counting functionality and level based on player performance. (2.4),(2.6)
b. Introduce game-over conditions, triggering when the Block Character collides with Obstacle Elements.(2.7)
c. Fine-tune gameplay elements for balance and challenge.

4. Week 4:

a. Showing the score on the 7-segment display on the basys3 board.(2.5)
b. Test and debug the entire system, ensuring smooth functionality.
c. Prepare documentation and presentation materials for project demonstration.