ADLD Course Project 2023

TEAM 15

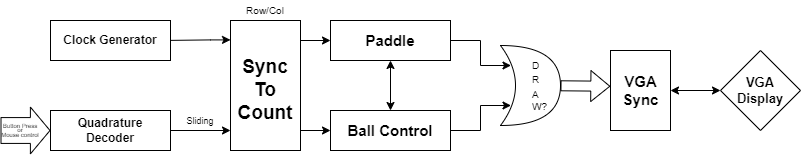
Guide: Dr.Saroja.V.Siddamal

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.no | Name |  | USN |
| 1 | RAMA B.GONDA |  | 01FE20BEC174 |
| 2 | BAKKESH V A |  | 01FE20BEC236 |

**PROBLEM STATEMENT**

PING PONG BALL GAME: The Ping Pong Ball Game is a project that involves creating a simple game using an FPGA (Field Programmable Gate Array). The game involves two paddles that are used to hit a ball back and forth across a screen. The objective of the game is to score as many points as possible by hitting the ball past the opponent's paddle.

**ARCHITECTURE**

****

The architecture we designed for ping pong ball game is shown as above. Our architecture consists of modules such as Clock generator, Quadrature Decoder which helps in the movement of paddle, Counters to draw the pixels, Ball control to control the direction of the ball, The or gate in the architecture defines where the ball and paddle should be drawn and a VGA sync which helps in syncing the horizontal and vertical pixels in the monitor which is finally connected to a display.

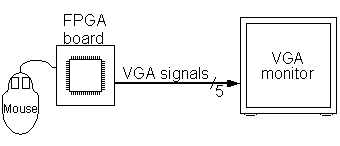
**THEORY**

To implement the game on an FPGA, the first step is to design the game logic using a hardware description language such as Verilog or VHDL. The game logic includes the ball and paddle movement, collision detection, and scorekeeping.

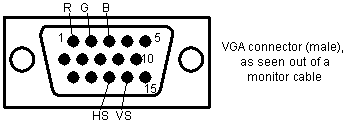
Next, the designed logic is synthesized and implemented on the FPGA using a development tool such as Xilinx or Vivado. We have used a Spartan 3 FPGA with Xilinx tool in our project. The FPGA is connected to a VGA display and input devices such as buttons or switches, which are used to control the movement of the paddles.

Once the hardware design is complete and implemented on the FPGA, the game can be played by users. The game can be further improved by adding features such as sound effects, different game modes, and increasing the complexity of the game logic.

The pong game consists of a ball bouncing on a screen. A paddle (controlled from a mouse or a button) enables the user to make the ball bounce back up.



A VGA monitor requires 5 signals to display a picture i.e R, G and B (red, green and blue signals).HS and VS (horizontal and vertical synchronization).The R, G and B are analog signals, while HS and VS are digital signals



The sync generator is best rewritten to be used as an HDL module where we generate R, G and B outside. Also the X and Y counters are more useful if they start counting from the drawing area.

use a mouse to move the paddle left and right on the screen.

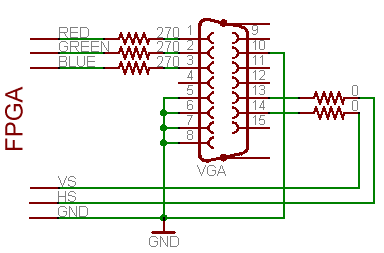
The ball needs to move around the screen, and bounce back when it touches an object (border or paddle).

First we display the ball. It is a square 16x16 pixels. We activate the drawing of the ball when CounterX and CounterY reach its coordinates.

We could check the coordinate of the ball against each object on the screen and determine if there is a collision. But that would become quickly a nightmare as the number of objects increases.

Instead we define 4 "hot-spots" pixels, one in the middle of each side of the ball. If an object (border or paddle) redraws itself at the same time that the ball draws one of its "hot-spot", we know that there is collision on that side of the ball.

Creating a video signal from FPGA pins:



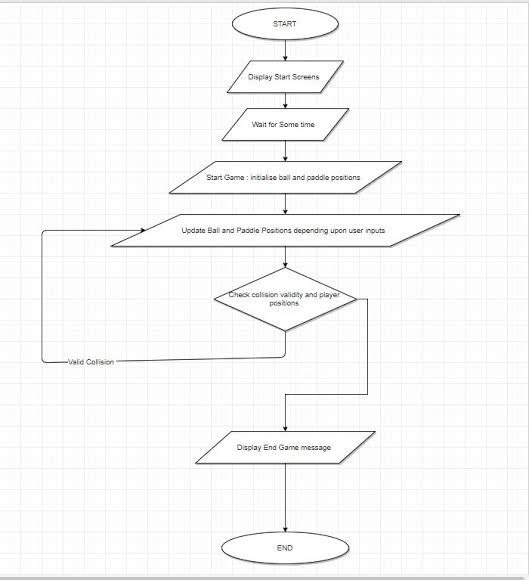
* Pins 13 and 14 of the VGA connector (HS and VS) are digital signals, so can be driven directly from two FPGA pins (or through low value resistors, like 100Ω).
* Pins 1, 2 and 3 (R, G and B) are 75Ω analog signals with nominal values of 0.7V. With 3.3V FPGA outputs, use three 270Ω series resistors. The resistors form voltage dividers with the 75Ω resistors in the monitor inputs so that 3.3V become 3.3\*75/(270+75)=0.72V, pretty close to 0.7V. Driving these RGB pins with different combinations of 0's and 1's gives us up to 8 colors.
* Ground pins are pins 5, 6, 7, 8 and 10.

A monitor always displays a picture line-by-line, from top-to-bottom. Each line is drawn from left-to-right.

But you specify when the drawing starts by sending short pulses on HS and VS at fixed intervals. HS makes a new line to start drawing; while VS tells that the bottom has been reached (makes the monitor go back up to the top line).

For the standard 640x480 VGA video signal, the frequencies of the pulses should be calculated or set to a standard of 25 Hz i.e 25 pulses per second.

**ALGORITHM**



**1.**The game starts by initializing the game window or screen where the gameplay will be displayed.The game screen is divided into two halves, representing the player's side and the opponent's side.A paddle is placed on each side, typically vertically aligned in the middle of the respective halves.A ball is positioned at the center of the screen.

**2.**The game begins with the ball moving in a specific direction (usually horizontally by increasing counter) towards one of the paddles.The player controls one paddle, while the opponent (computer or another player) controls the other.The player's objective is to hit the ball with their paddle, trying to make it pass the opponent's paddle and score a point.The opponent's objective is to do the same, trying to prevent the ball from passing their paddle.

**3.**The player controls their paddle using input devices such as a keyboard, mouse, or some keys.The paddle can typically move vertically within a defined range on the player's side.The player's input determines the paddle's movement, allowing them to position it strategically to hit the ball.

**4.**The ball moves continuously, usually in a straight line with a constant speed.In each frame or iteration of the game loop, the ball's position is updated based on its current direction and speed.The ball's direction can change when it collides with the paddles or the boundaries of the game screen.

**5.**Collision detection is crucial to determine if the ball hits a paddle or the game screen boundaries.When the ball collides with a paddle, its direction is altered, typically resulting in a bounce off the paddle.The angle of the bounce depends on the relative position at which the ball hits the paddle.If the ball collides with the top or bottom boundaries of the screen, its vertical direction is reversed.

**6.**The game keeps track of the score for both the player and the opponent.When the ball passes the opponent's paddle and goes beyond the game screen's boundaries, the player scores a point.The score is updated accordingly, and the ball is reset to the center for the next round.

**7.**The game continues until a specific winning condition is met, typically reaching a predefined score limit.Once a player reaches the winning score, the game ends, and the winner is declared.

**8.**The game logic is implemented within a game loop, which runs continuously until the game is over.The game loop handles updating the positions of the paddles and the ball, detecting collisions, checking for scoring events, and rendering the game's visuals.

**THANK YOU**