Pong Game using Basys 3
Project Report

Samar Ahmed-900222721
Omar Leithy-900221663
Habiba Elsayed-900221264

Fall 2024
CSCE230101-Digital Design I
Dr. Mohamed Shalan

1. Introduction

Pong, a groundbreaking electronic game, was released in 1972 by the American game manufacturer Atari, Inc. As one of the earliest video games, Pong became immensely popular and played a crucial role in launching the video game industry (The Editors of Encyclopaedia Britannica, 2019). The original game features two paddles controlled by players, used to hit and block a small ball that bounces across the screen. The first player to reach a score of 10 wins!

The conceptual groundwork for Pong was laid by German engineer Ralph Baer in 1958 when he proposed creating a simple video game that could be played on home television sets (The Editors of Encyclopaedia Britannica, 2019). Building on this idea, Atari adapted the concept into an arcade game through extensive innovation and effort. This led to the production of over 8,000 Pong arcade machines, and by 1975, Pong had evolved into a console game, marking a significant milestone in the gaming industry.

In this project, we explore the process Atari undertook to create Pong, utilizing modern tools such as the Basys 3 board. Through components like a VGA connector, the 7-segment display, switches, and push buttons, we have developed a fully functional Pong game featuring multiple states. This report details every aspect of the project, beginning with the design approach, followed by the modules used, the challenges faced, and the contributions of the team in bringing this project.

2. Design Approach

Hierarchical Implementation

The system is broken into multiple interconnected modules, each handling a specific functionality. We have 9 modules. This is mainly to simplify debugging and allow the module to be reused. These modules are found in a hierarchy. First, we have the Top Module, which acts as the central hub, integrating all the submodules to achieve the overall functionality. We then have our submodules. The controller manages the VGA signal generation and ensures the proper timing for display rendering. The graphics_Gen handles all the visual elements, including the ball, paddles, and border. The textGen, which creates text displays such as “SCORE” or “GAME OVER”, is also where the asciiRom module is used. Finally, we have the Push_button module that allows us to start the game which contains multiple modules including the PosEdgeDetector, the debouncer, and the synchronizer.

Inputs and Outputs

Inputs include the clock, paddle movement controls (up1, down1, up2, down2), and the synchronization inputs (x, y, video_on). The outputs will include the RGB signals for rendering graphics, paddles (pad1On, pad2On), ball (ballOn), score display (score1, score2), and game states using winner and buzzer_output.

Parameters

Dimensions and positions are parametrized to allow more flexibility. Thus the screen limits are: X_MAX = 639, Y_MAX = 479. The paddle size and velocity, (padHeight = 90, padVelcoity = 2). The ball dimensions and initial positions (ballSize, BALL_CENTER_X, BALL_CENTER_Y) and finally the border thickness (BORDER_THICKNESS).

Game Logic

Finite State Machine using OHE

The three states are Start, Play, and GameOver. The start clears the ball, score, and paddle display. Stopping any scores and movement of the ball, and it is waiting for the player to push a button. We then have the Play where there is active gameplay. The 2 players can now move their switches to move their paddles. It will continue until one of the players reaches a score of 10. While this is happening on the screen there is also a sound effect for when the ball collides with a wall, and their current score is displayed on the seven-segment display. Finally, the game over is the endgame state with the option for the user to restart, and a display of which player has won. The main purpose of this state machine is to ensure a clear and controlled transition between the gameplay phases and enhance system robustness.

Score Tracking and Winner Determination

When the state is at “Play”, the scores would be incremented depending on when the ball touches the left or right borders, with win conditions at 10 points for either play. The “winner” output is set when a player achieves the score threshold. Thus if player 1 wins it is given an encoding of 2’b01, whereas if player 2 wins the encoding is 2’b10. Furthermore, the score is output to score1 and score2 which is then used in the Top Module to give the 7 segments their required binary sequence to output the correct digits on display using specific anodes.

The 7-segment Display uses a clock divider that generates a slower clock signal for multiplexing. It then Displays alternate values rapidly between the two scores, creating the illusion of simultaneous display.

Buzzer Sound System

The buzzer provides a sound system to the game for a significant event. In this case, it is when the ball collides with the ball. It utilizes the WALL_FREQ and PADDLE_FREQ for adjustable frequency and durations HALF_SECOND) to create a distinct tone for the different events.

Graphics Rendering

Borders are mainly designed using the border logic to occupy a fixed-width perimeter for the screen. The paddle's positions are tracked and updated dynamically based on the input controls from (up1, down1, up2, and down2). Then the ball motion would follow the xDelta and yDelta vectors, updating its position every refresh cycle. The collisions with walls, paddles or

borders would modify its velocity and direction. Finally, to display this all on the screen there is a conditional hierarchy that determines the color rendering, where priority is given to the text, ball, paddles then the borders. There is a refreshTick signal that synchronizes updates to game elements based on the refresh cycle of the display.

Text Rendering

The textGen would calculate the pixel regions for characters using the lookup table (asciiRom). The text alignment is then handled dynamically based on the pixels coordinates and the game state giving us different messages and displays at different states throughout the game.

Push Button Logic

We implemented a debouncing logic to ensure reliable detection of the user inputs, and we added a synchronizer to ensure accuracy, such that when the button is pressed down it is entered long enough to be noticed as input to start the game and at the same time, it is not held for a very long time where there could be an issue when switching states.

Game Reset Logic

Reset can be accessed by flipping a switch at any time throughout the game. On reset, the scores, ball position, and paddle position are reinitialized. Furthermore, all states return to their defaults.

3. Modules Used

VGA Controller:

The VGA controller is in charge of the time signals sent to the pixels in addition to the pixel positioning on the display. The module has two sets of timing parameters: horizontal and vertical. The horizontal timing parameters include H_DISPLAY, H_FRONT, H_BACK, H_SYNC while vertical parameters include V_DISPLAY, V_FRONT, V_BACK, V_SYNC. The H and V displays are the number of visible pixels in one row or the number of visible rows as per the VGA display standards. The front and back porches (V_FRONT, V_BACK, H_FRONT, H_BACK) are the times before and after a sync pulse respectively. Lastly, the V_SYNC and H_SYNC are the durations of the vertical and horizontal sync pulses. So, this means that the total time it takes to parse a single row of pixels is H_TOTAL = H_DISPLAY + H_FRONT + H_BACK + H_SYNC and the total time it takes to parse all the rows of pixels on the screen is V_TOTAL = V_DISPLAY + V_FRONT + V_BACK + V_SYNC. Two counters, namely hCounterReg and vCountReg, rely on these timing parameters to track the horizontal (x) and vertical (y) pixel positions on screen so that each pixel on the screen is allocated an appropriate color based on if this part is a ball or the score or even the borders of the game. Another important variable in the controller module is the video_on signal which detects if the pixel being parse is within visible range that is the 640x480 pixel screen. Additionally, this module executes a clock divider that divides the 100 MHz clock signal from the FPGA into 25 MHz which is suitable for display on the monitor.

Top Module

The top module is where most of the other modules are instantiated and where the main game logic takes place. For instance, graphicsGen and textGen are instantiated in this module to generate bitmap and geometric graphics on the screen including the ball, paddles, border, score text, start and game over text, and the pong game logo. Additionally, the finite state machine that is responsible for deciding which state the game is operating in is in this module. The finite state machine depends on the player scores that are incremented in the graphicsGen module. Based on the output state of the FSM in this module, the corresponding output is managed in the graphicsGen and textGen modules. This module is also responsible for the 7-segment decoder which takes the scores of the players and transforms the numbers into a 7-segment display. To display the correct color for each component on the screen, this module utilizes multiplexing based on the video_on signal and the state of the game to decide on pixel colors. Thus, it assigns fuschia to the ball, green to the left paddle, purple to the right paddle, red to the score, white to the pong text, and black to everything else.

Ascii ROM

The asciiRom module includes the bitmap for all the letters and numbers used to display scores or state texts in the game. These bitmaps were used mainly in the textGen module to display the score, the start message, the game over message and the winning message.

Graphics Generator

This module is where the balls, paddles, and the borders of the screen are drawn using bitmaps. After outputting the components, this module manages the functionality of all three components (the two paddles and the ball). The ball, for instance, moves with the velocity controlled by the xDeltaReg and yDeltaReg variables and bounces off the paddles and all of the borders. The paddles on the other hand are controlled by variables such as up1 and up2 and are constrained to the perimeters of the screen by collision with the borders. As collisions are handled in this module, this is also where the sound logic is implemented. An FSM is implemented here to decide which sound the buzzer needs to output depending on the type of collision the ball is encountering. Additionally, if the game is in the game-over state, a different sound logic is implemented.

Text Generator

This module instantiates the asciiRom model to draw bitmaps based on the state of the game that is passed to it by the top_module. It also controls the positioning of the text on the screen, the color of the text and its size.

Push Button

Manages the button used when the player wants to start the game. This module also instantiates other sub-modules which include a synchronizer, a positive edge detector and a debouncer.

4. Challenges Faced

One of the challenges we faced in the beginning was to get an output on the screen. The different clock rates of the Basys 3 and the required clock rate for the VGA were a problem; but we managed to use a simple clock divider, vertical sync, and other technologies following the handout provided by the Professor to fix these problems. Also, we faced some challenges implementing the ball's movement, and collision with the paddles and walls. These challenges were eventually overcome by fixing logical errors in the code, which required a lot of debugging. Finally, we faced challenges using the buzzer to output sound, and outputting a nice rhythm when a player finally wins. After a lot of debugging and trial and error, we were able to get the buzzer to work as intended and output different frequencies.

5. Hardware Optimization

A clock divider is used to reduce the main clock frequency to fit VGA timing and for the 7-segment display. We also take into consideration the color depth, where we use a 4-bit color for each channel (RGB) to allow a balance between quality and resource usage. Finally, there is resource sharing, where we combine overlapping modules in the graphics_gen for streamlined rendering.

6. Contributions

7.

Group Work: The entire team has done research on this topic before starting it, and started with a group meeting to implement the block diagram and get a basic display on the monitor using the FPGA. Every member has also participated in debugging and fixing errors or bugs in the game.

Habiba Elsayed: Worked on outputting the ball on screen and the ascii ROM module which has all the bitmaps for the displayed words. Also worked on displaying the score correctly on screen using the bitmap and the making of the FSM and the conditions and the encoding of each state.

Omar Leithy: Worked on ensuring the ball's and paddles' movement is smooth, and that collisions were handled properly. Also worked on incorporating sound effects into the code using a buzzer as an output, utilizing an FSM to map the sound to every state the ball is in.

Samar Ahmed: Worked on the PONG display, the score system, such that it increments during collision and outputs on the 7-segment display, She has also worked on the FSM, such that the play is a state, and when every text display should be outputted, and when to disable required objects.

8. Future Improvements

Through this logic, we believe that this project can be extended and improved in the future. For instance, future features can include: adding more players/balls, introducing different

game modes with different themes, enhancing the visual effects or sound cues, including levels to the game such that the ball speeds up as the scores increase.

9. Acknowledgment

We would like to express our sincere gratitude to Dr. Shalan for providing us with the opportunity to work on this project, which greatly enhanced our understanding of the Basys 3 and Verilog. We would also like to extend our thanks to Dr. Suzanne and Dr. Anany for their invaluable guidance in teaching us how to effectively utilize Verilog. This project would not have been possible without their support and expertise.

References

The Editors of Encyclopaedia Britannica. (2019). Pong | electronic game. In *Encyclopædia Britannica*. https://www.britannica.com/topic/Pong