Creating *Pong* Game in Verilog

Jordan Smith, Andy Ho, Jared Alanis, and Benjamin Black

Abstract—This project aims to recreate the classic *Pong* game on the Nexys A7-100T which contains a Xilinx Artix-7 field programable gate array (FPGA) using Verilog, introducing innovative features for enhanced gameplay. The implementation leverages a video graphics array (VGA) controller for rendering, with the game's point and lives displayed on a seven-segment display (SSD). User interaction is facilitated through two buttons for vertical movement of the paddle and a reset button for initiating new games. A 4-bit switch customizes the number of lives before a game-over condition. In addition to the traditional *Pong* setup, the project incorporates a pulse width modulation (PWM) module. Unlike the traditional use, this PWM functionality dynamically adjusts the intensity of a light emitting diode (LED), providing a visual cue that changes based on the selected difficulty level. Three individual switches enable users to choose between easy, medium, and hard difficulty levels. In addition to the LED serving as a visual indicator for difficulty, there are now three additional LEDs. These LEDs indicate when the ball has been hit, when the ball was missed, and when the game is over. This comprehensive approach combines hardware components, Verilog programming, and dynamic features like LED intensity adjustments and status LEDs to offer an engaging and adaptable *Pong* gaming experience on the FPGA platform.

Index Terms—FPGA, LED, PWM, seven-segment display, video graphics array, Verilog

I. INTRODUCTION

After learning much about what the Nexys A7-100T FPGA board can be utilized for, the next step was to put everything that was learned and use it in a single project, the *Pong* game. The *Pong* game was created in Verilog using the Vivado Design Suite by AMD. Using this software suited, hardware design language (HDL) designs can be synthesized and analyzed. This paper will show the process of creating and implementing *Pong* in Verilog utilizing the Nexys A7-100T FPGA.

II. MODULES

Creation of the *Pong* game was done through the combining of multiple different Verilog modules all under a top-file. Different modules were created in order to drive the VGA and the seven-segment displays, control how many lives a player will spawn with, the movement of the ball and paddles, the point system, a pulse width modulation driver in order to control LED brightness, and to render and display the game and its elements on a screen.

A. VGA Driver

A VGA driver was created in Verilog in order to display the game and its contents on a screen at a resolution of 640x480p. The driver counts each pixel from row to row in order to output all pixel information that is needed in order to generate the game. This module runs at a clock speed of 25 MHz in order to output properly at a refresh rate of 60 Hz.

B. Seven-Segment Display Driver

The seven-segment display driver uses a clock input in order to cycle between one of the eight seven-segments that are present on the FPGA board. The values that are input into one of the seven-segments are converted from hexadecimal to a 7-bit binary sequence that controls which LED in the seven-segment is enabled or disabled.

C. Lives Module

Lives module three inputs in order to reset the module, 4 switches to set the number of lives from a minimum of 1 to a maximum of 15, load the input values from the switches and 1-bit input for if the player has missed a ball or not. Utilizing the 100 MHz of the FGPA, the module will check repeatedly to see if a ball has been missed, and if this is true the lives of the player will decrease. If the player’s lives reaches 0, the module will output a 1-bit game over signal.

D. Difficulty Module

The difficulty module is used in order to increase the speed of the ball based on which difficulty level the player desires. The speed is stored as 2-bits that can range from 0 to 3. This module also outputs a 12-bit duty value that will increase or decrease the brightness of an LED based on the ball speed using the PWM module.

E. PWM Module

Pulse width modulation is used in order to control the brightness of an LED by decreasing or increasing the amount of time the LED is on. This module is used to represent increased difficulty as a brighter LED. An 8-bit input is used to change the duty cycle of the PWM output, with an input of 255 representing the maximum duty cycle and 0 the minimum duty cycle of the module.

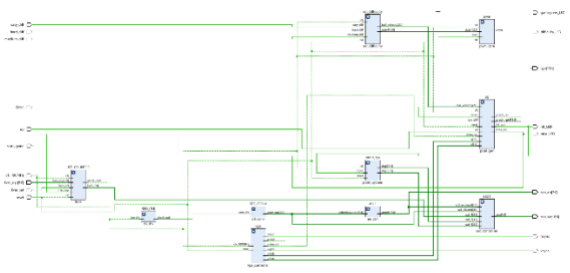

Fig 1. Schematic of *Pong* game

III. CONCLUSION

This project resulted in the successful implementation of a single-player Pong game. The game features a paddle on the right side of the screen and a wall on the left, with a circular bouncing ball between them. The play area encompasses the entire display, except for the leftmost portion hidden behind the wall. The player's score is prominently displayed on the four left-most seven-segment displays of the Nexys A7-100T FPGA board, increasing by 1 each time a point is scored.

The four leftmost switches on the controller allow the player to set the number of lives for each gameplay session. Pressing the right button locks in the selected value, showcasing it on the rightmost seven-segment displays. The Pong game offers three difficulty levels: easy, medium, and hard. The first three switches on the right side determine the difficulty level, with the first representing easy, the second medium, and the third hard mode. A red LED indicates the selected mode, becoming brighter as the difficulty level progresses from easy to hard.

To maneuver the paddle up and down, the player utilizes the up and down buttons on the controller. Once the player exhausts all their lives, the game concludes and can be reset by pressing the middle button on the controller.

The most challenging aspect of the implementation was devising distinct difficulty levels. As difficulty increases, the ball's velocity rises by 1 pixel, starting at 2 pixels and reaching a maximum of 4 pixels. However, this adjustment posed an issue as the ball sometimes phased through the wall and paddle instead of bouncing against them. This issue stemmed from the interplay between the frame rate, clock speed, and the ball's velocity. The ball’s position updated rapidly, and the ball would skip over the position where a collision should have been detected and would instead phase through the objects. To address this, a collision margin parameter of 4 pixels was introduced, creating a 'safety zone' around the boundaries of the paddle and wall. This additional area ensures that collisions are detected even when the ball's rapid movement might otherwise skip over the precise collision point.

After the inclusion of the collision margin, other issues surfaced, such as double registering of misses, resulting in a decrease of two lives instead of one. Hits were also registering twice, leading to double the points. To ensure that only one miss or hit is recorded per event, a miss flag was integrated into the lives module, and a count flag was incorporated into the point system module. These additions resolved the unintended doubling of misses and hits, ensuring accurate scoring and maintaining the integrity of the gameplay experience.

The development of this Pong game project not only provided hands-on experience in implementing classic arcade gameplay but also presented valuable challenges in game design and real-time logic. Overcoming obstacles such as frame rate limitations and collision detection difficulties demonstrated the importance of accurate system design and the interplay of hardware components. The process of identifying and resolving issues showcased the significance of debugging techniques and effective collaboration. Overall, this project has been a rewarding journey in applying digital logic concepts to create an engaging and functional gaming experience, highlighting the intersection of hardware and software in the realm of FPGA-based game development.