2-Player Pong

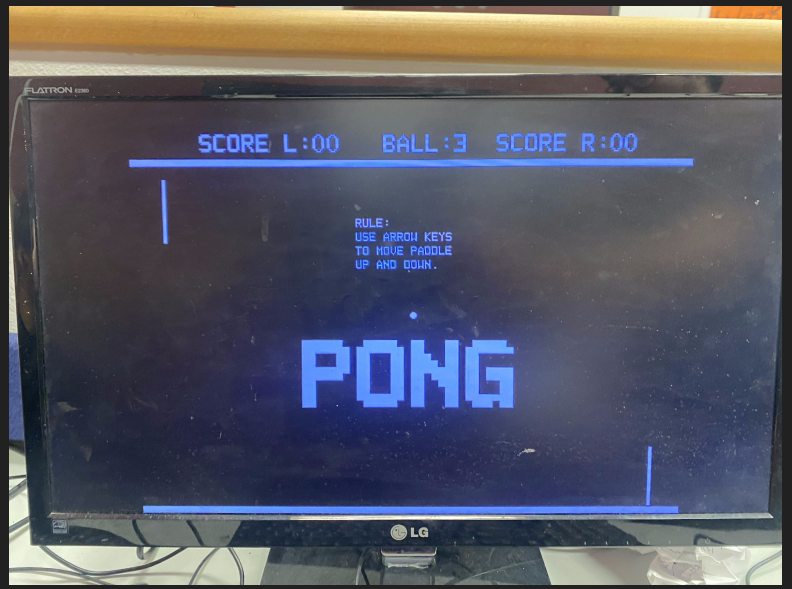
By: Cuauhtemoc Lona, Peter Tran, Rafael Baltazar, Nathaniel Case

ECE 3300.02, Fall 2023

To control Paddles:
Use Keyboard Up/down arrows for right paddle and for left paddle use W / S keys

Rules:
Each player defends their side with a paddle to deflect the ball away. Points are gained by hitting the ball with the paddle. It is a Best of 3 where each round is played until someone misses. Whoever has the highest points after 3 Rounds wins.

Demonstration:



Files for the Project

VGA Controller (vga_controller.v)

- Generates hsync, vsync, and pixel position for our 640x480 output signal

Inputs

- **clk_25MHz** (25 MHz input pixel clock)
- **reset**

Outputs

- **video_on** (ON while pixel counts for x and y and within visible display area)
- **hsync** (horizontal sync)
- **vsync** (vertical sync)
- **[9:0] x** (pixel count/position of pixel x, max 0-799)
- **[9:0] y** (pixel count/position of pixel y, max 0-524)

Score Counter (score_counter.v)

- Counts score and increments the ones and tens digit

Inputs

- **clk**
- **reset**
- **[1:0] d_inc** (digit increment)
- **d_clr** (digit clear)

Outputs

- **[3:0] dig0** (ones digit for left score)
- **[3:0] dig1** (tens digit for left score)
- **[3:0] dig2** (ones digit for right score)
- **[3:0] dig3** (tens digit for right score)

Debounce (debounce.v)

- Debouncing is removing unwanted input noise from buttons, switches or other user input.

Inputs

- **clk**
- **btn_in_1**(Input signal 1)
- **btn_in_2** (Inout signal 2)

Outputs

- **btn_out_1** (Debounced input 1)
- **btn_out_2** (Debounced input 2)

Keyboard (keyboard.v)

- Reads Keyboard inputs and maps their signals to control the paddles in game

Inputs

- **keyboard_clk** (top clk)
- **keyboard_kclk** (PS2_CLK from keyboard)
- **keyboard_kdata** (PS2_DATA from keyboard)

Outputs

- **keyboard_uart_rxd** (UART_RXD_OUT, unused)
- **[31:0] keyboard_out** (raw keyboard data input)
- **[3:0] keyboard_code** (interpreted key press output)
-

Seven Segment Driver (SEG_7.v)

- Takes raw keyboard data input and outputs hex key code on seven segments

Inputs

- **clk**
- **[31:0] x** (keyboard data input)

Outputs

- **[6:0] seg** (CC output)
- **[7:0] an** (AN output
- **dp** (decimal point output)

ASCII ROM (ascii_rom.v)

- ROM that stores 8x16 pixel ASCII characters, used to creates the letters and other symbols used in PONG, such as the alphabet and numbers

Inputs

- **clk**
- **[10:0] addr** (Input memory address)

Outputs

- **[7:0] data** (one 8 pixel row of a character)

Pong Text (pong_text.v)

- Utilizes ascii_rom module to help create characters to display on screen.
- Chooses what characters are displayed, when they are displayed and where
- Allows scaling of characters (16x32, 32x64)

Inputs

- **clk**
- **[1:0] ball** (number of balls left)
- **[3:0] dig0** (BCD input for left ones)
- **[3:0] dig1** (BCD input for left tens)
- **[3:0] dig2** (BCD input for right ones)
- **[3:0] dig3** (BCD input for right tens)
- **[9:0] x** (X position from VGA)
- **[9:0] y** (Y position from VGA)

Outputs

- **[3:0] text_on** (each bit is a flag telling when to render score, logo, rules, and game over text)
- **[11:0] text_rgb** (RGB output to VGA)

Pong Graph (pong_graph.v)

- Controls paddle movement, ball physics, paddle/wall/ball/background rendering

Inputs

- **clk**
- **reset**
- **[3:0] btn** (controls up/down of each paddle)
- **gra_still** (flag if graphics are still)
- **video_on**
- **[9:0] x**
- **[9:0] y**

Outputs

- **graph_on** (flag to render paddle, wall, ball, background)
- **[1:0] miss** (left or right miss)
- **[1:0] hit** (left or right hit)
- **[11:0] graph_rgb** (RGB output to VGA)

Timer (timer.v)

- 2 second down counter, uses 60 Hz refresh rate as reference
- Used to wait before a new game starts and when a game finishes

Inputs

- **clk**
- **reset**
- **timer_start** (flag to start timer)
- **timer_tick** (external clock tick)

Outputs

- **timer_up** (flag representing if timer is finished)

Pong Top (pong_top.v)

- Top file, also controls the states of the game (newgame, play, newball, over) and RGB multiplexing from other modules

Inputs

- **top_clk** (100 MHz system clock)
- **pix_clk** (25 Mhz pixel clock)
- **reset**
- **key_clk** (PS2_CLK from keyboard)
- **key_data** (PS2_DATA from keyboard)
- **inputsw** (Switch between onboard buttons or keyboard input)
- **[3:0] btn** (Onboard buttons)

Outputs

- **hsync**
- **vsync**
- **[11:0] rgb**
- **[7:0] ssd_port_cc** (outputs A-G to SSDs)
- **[6:0] ssd_port_an_out** (outputs an)
- **ssd_port_odp** (output decimal point)

Resources:

- FPGA Dude (Single-Player Pong Repo)
- VGA Controller Verilog Tutorial
- FPGA Labkit by MIT - VGA
- https://www.synopi.com/interlaced-and-progressive-video
- Nexys A7-100T Keyboard Repo
- ZipCPU/vgasim: A Video display simulator (github.com)
- Veripool - Verilator