REPORT
LOGIC DESIGN LAB
FINAL PROJECT

第 11 組

107000276 林家合

107062332 邱世綸

INTRODUCTION

PROJECT TITLE: ARCADE GAME - PONG (1972 ATARI)

Summary

Pong is a table tennis sports game featuring simple two-dimensional graphics, manufactured by Atari and originally released in 1972. The player controls an in-game paddle by moving it vertically across the left or right side of the screen. They can compete against another player controlling a second paddle on the opposing side. Players use the paddles to hit a ball back and forth. The goal is for each player to reach three points before the opponent; points are earned when one fails to return the ball to the other.

MOTIVATION

We decide to write a game is mainly because we think it will be both interesting and challenging to write a game using Verilog. We usually write a game by using a game framework (such as Allegro or Love2D) or game engine (such as Unity or Unreal Engine), but we haven't written any game using Verilog before, so we decide to give it a try.

Control and I/Os

Center Button : Reset

Keyboard

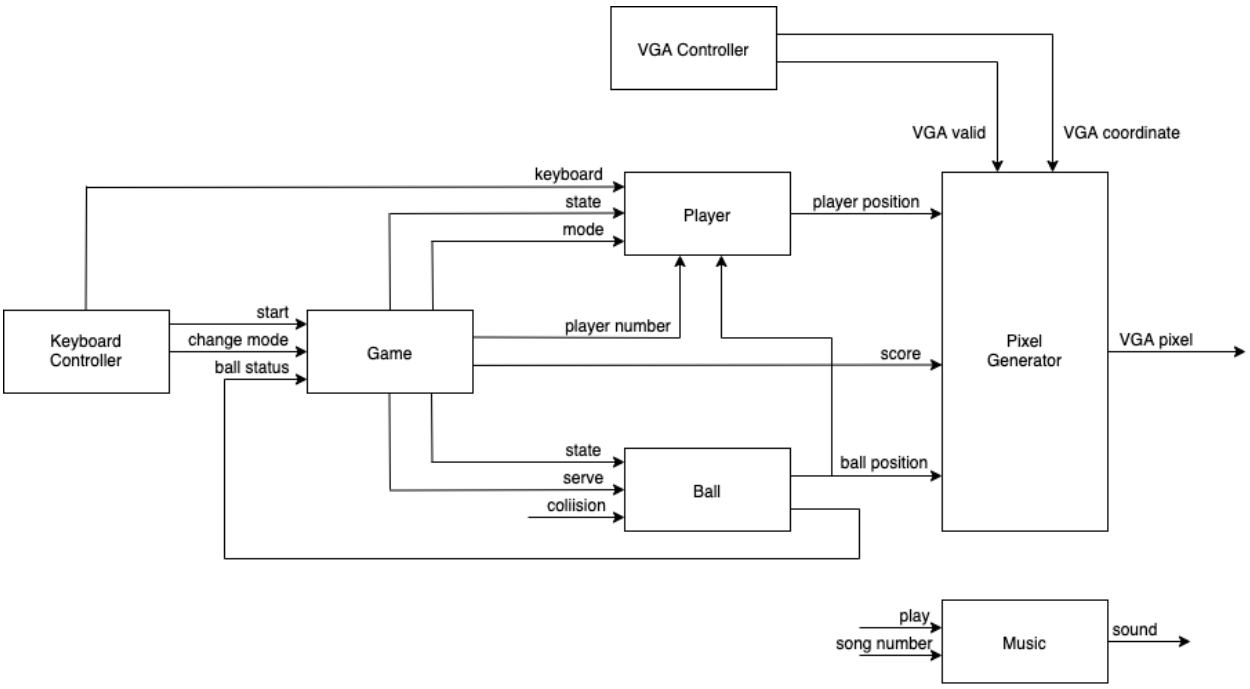
1, 2, 3, 4 : Change Mode

Player 1	Player 2
W : Move Up	Up : Move Up
S : Move Down	Down : Move Down

Enter : Start Game

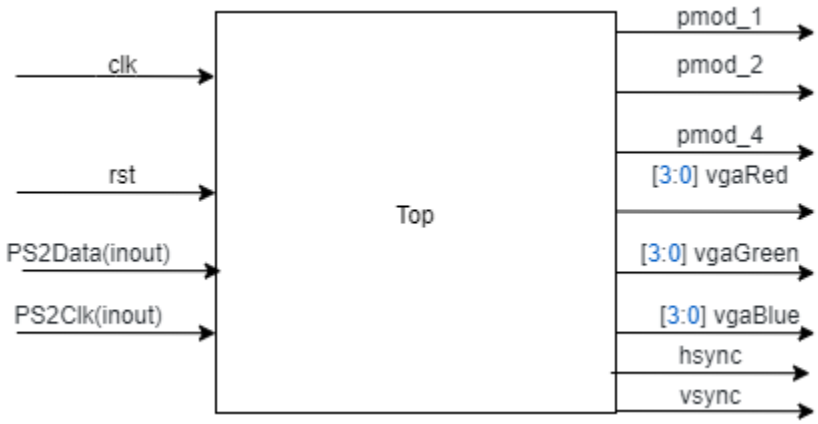
VGA and USB interface (for keyboard) are also used.

BLOCK DIAGRAM



SYSTEM SPECIFICATION

TOP



The Top module calls modules and passes values to each module.

We call Keyboard Decoder module to get the inputs which are W, S, up, down, enter (user input) and 1, 2, 3, 4(game mode). Debounce and onepulse are needed to remove the noises and synchronize keyboard signal with game module.

We call Pixel Generator module to output each RGB value of the pixels on the screen. This module is used to show the scores, paddles, borders and ball, the pixels that are needed to print on the screen.

We call VGA Controller module to output the game content to external monitor using VGA connector.

We call Player module to get the position X and position Y of each paddle, and then pass them to Pixel Generator module to show two paddles images.

There are two paddles in our games, so we use a variable in Player module to distinguish two different paddles. By doing so, we can instantiate two players by calling the same module twice but passing different values into the variable.

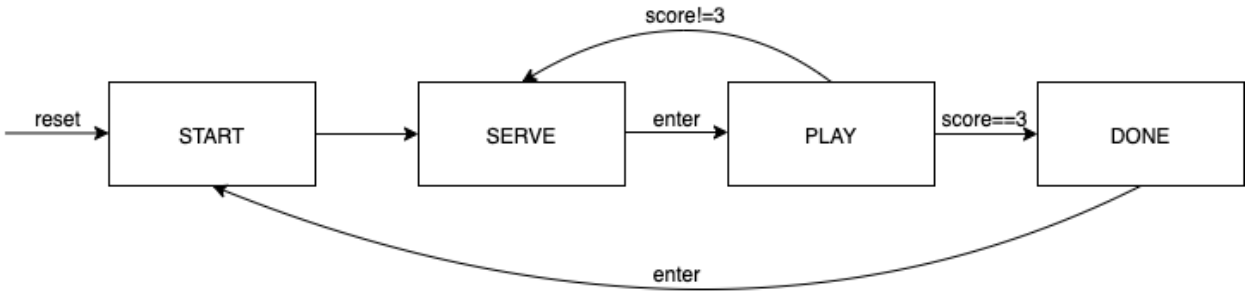
We design enable1, enable2, enable3, enable4, which are to check whether the four edges of ball have collisions against the border or the paddle. Enable5, enable6 are designed to check which player has won. All these enable signals are passed into music top module to control the timing of playing sound effects.

The sound effect is only played once when the ball collides with border or paddle and when the ball reaches the left side or right side of the screen. To synchronize the signal with the music top module, we use a counter to determine when to stop playing sound effect.

We call Ball module to get position X and position Y of the ball and then pass them to Pixel Generator module to show ball image.

GAME

Game module is an FSM that controls the game flow, based on the keyboard and in-game info.



// -- State of our game :

// -- 1. 'start' (the beginning of the game, before first serve)

// -- 2. 'serve' (waiting on a key press to serve the ball)

// -- 3. 'play' (the ball is in play, bouncing between paddles)

// -- 4. 'done' (the game is over, with a victor, ready for restart)

// -- Game mode of our game :

// -- PvP : player vs player

// -- PvA : player vs AI

// -- AvP : AI vs player

// -- AvA : AI vs AI

The initial state is START so that game is initialized by pressing reset button.

- START : Initialize the game. Two player paddles are initialized at the left and right side of the screen respectively. A ball is placed in the middle of the screen. Player 1 serve ball first. Set game mode to PvP and both players' scores to 0 and go to SERVE state.
- SERVE : Game mode can only be changed in SERVE state. Press enter to serve ball and go to PLAY state.
- PLAY : Change state based on the ball status and player's score. Update player's score when either player1 or player2 wins the round. Defeated player serves ball in the next round and go to SERVE state. First player to reach 3 points wins the game and go to DONE state.
- DONE : Press enter to restart the game and go to START state.

PLAYER

This module determines the position of the paddles based on the keyboard input, game mode and game state.

The position X of both paddles is fixed, player 1 is on the left-hand side, player 2 is on the right-hand side.

When reset button is pressed, the Y coordinate of each paddle will reset to the middle of screen.

We use a counter to control the movement of the paddle, the paddle can only move when the counter counts to a certain number. The counter will only count in SERVE and PLAY states, so that the paddle can only move in these states. We use this counter to slow down the paddles' speed to a degree that is suitable for the players to play with.

We design a rule-based AI that move in the same direction with the ball, the middle position of the paddle is always the same as the ball, so that the AI can always hit the ball back in any case.

Only player control is restricted by the counter, AI movement is independent of the counter.

State :

- START & DONE : Set paddles position to middle of each side.
- SERVE & PLAY :
  There are 4 game modes in player module, which are PvP, PvA, AvP and AvA. P stands for player and A stands for rule-based AI.
  When paddle is under the control of player, the position Y of paddle is change according to the keyboard input when the counter counts to a certain number.
  When paddle is under the control of AI, the position Y of paddle is changed according to the position Y of ball. When the paddle touches the upper or the lower border, the paddle cannot go beyond the upper and lower border.

Game Mode :

Player vs Player : Both paddles are controlled by players.

Player vs AI : Left paddle is controlled by player and right paddle is controlled by AI.

AI vs Player : Left paddle is controlled by AI and right paddle is controlled by player.

AI vs AI : Both paddles are controlled by players.

BALL

Ball module determines the position, moving direction and speed of the ball based on game state and collision.

// -- Ball status :

// -- Playing

// -- Player1 Win

// -- Player2 Win

When the reset button is pressed, ball position is set to the middle of the screen, speed is set to the original speed.

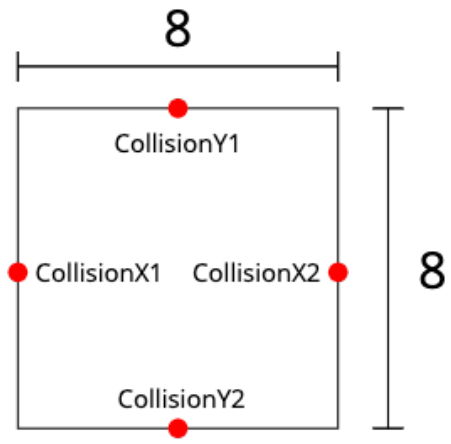
We use a counter to control the movement of the ball, the ball can only move when the counter counts to a certain number, by adjusting the value of this counter, we can change the speed of the moving ball. The counter will only count in PLAY state, so that the ball will only move in PLAY state. The speed of the ball will increase when the ball collides with the paddle.

The ball can move in four directions: up-left, up-right, down-left and down-right.

Collision detection is done in top module, top module checks if the edge of the ball is colliding with the top-bottom borders or the paddles, and send CollisionX1, CollisionX2, CollisionY1, CollisionY2 signals to the ball module. CollisionX determines the next horizontal moving direction, CollisionY determines the next vertical moving direction.

When both values of CollisionX1 and CollisionX2 are equal to 1, revert the current horizontal moving direction; If CollisionX1 is equal to 1, next horizontal moving direction is right; If CollisionX2 is equal to 1, next horizontal moving direction is left; If both values of CollisionX1 and CollisionX2 are equal to 0, the ball is moving in the same horizontal direction.

When both values of CollisionY1 and CollisionY2 are equal to 1, revert the current vertical moving direction; If CollisionY1 is equal to 1, next vertical moving direction is down; If CollisionY2 is equal to 1, next vertical moving direction is up; If both values of CollisionY1 and CollisionY2 are equal to 0, the ball is moving in the same vertical direction.



State :

- START : Set ball position to the middle of the screen and speed to the original speed.
- SERVE : Set ball position to the middle of the screen and speed to the original speed. Next moving direction is set based on the serving player. If the serving player is player1, next horizontal direction is right, if the serving player is player2, next horizontal direction is left. The vertical direction is random.
- PLAY : Change the moving direction based on the collision and move the ball according to the direction. If the ball reaches the left side or right side of the screen, change ball status to player win.
- DONE : Change ball status to playing and set ball position to the middle of the screen.

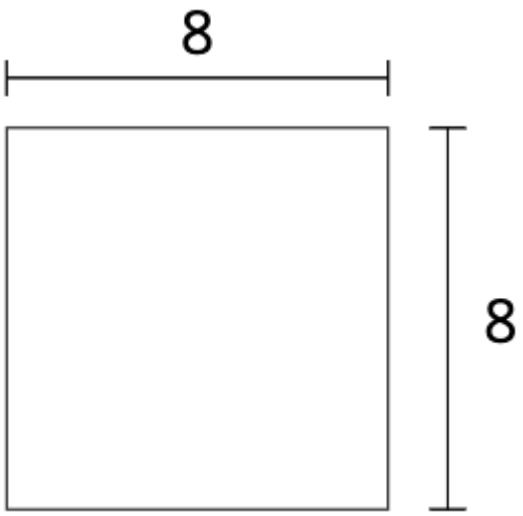
PIXEL GENERATOR:

In this module, we determine which area need to display as white pixel such as border, paddle, ball and scores, or black pixel such as background.

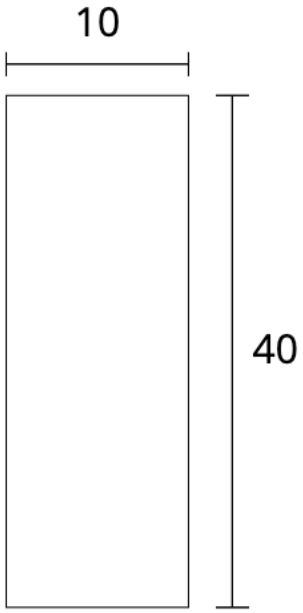
To display the white area (border, paddled, ball, scores), we need to assign the RGB value of the pixel to the corresponding value and vice versa.

The dimension of each object is listed below:

Ball



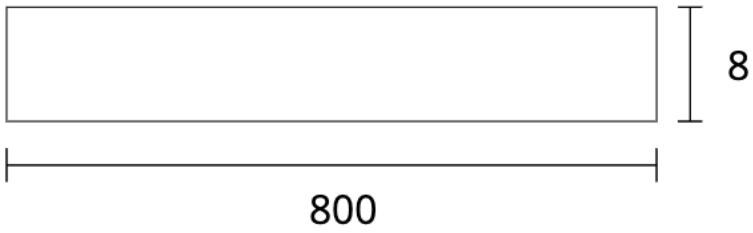
Paddle



Score

zero	one	two	three
28 x 32	10 x 32	24 x 36	24 x 36
			

Border



Our game screen dimension is 800 x 640. By having all the width, height and position of each object, we can assign RGB values of each pixel according to the current h_cnt and v_cnt. If h_cnt and the v_cnt is within the object, the pixel will be white, else will be black.

MUSIC

Music module play music when the ball collides with border or paddle and when the ball reaches the left side or right side of the screen.

There are two sound clips in the music module, which are collision sound effect and win sound effect. We use two PlayerCtrl and music modules to play different sound effect.

When the ball collides with border or paddle and when the ball reaches the left side or right side of the screen, top sends play signal to music module and sets which sound effect to play.

We use a counter to determine how long the sound clip should play, when the counter counts to a certain number, reset the play signal.

EXPERIMENTAL RESULTS

We create the same pong game that was originally released in 1972 by using Basys3 Artix-7 FPGA. We wish to create the original retro classic style pong game, so we decide to use pixel as our main design. We choose keyboard as our input as pong is a fast tempo game, playing with a keyboard can have a better gaming experience. We successfully implement the physics engine of this game, which is mainly the collision detection, player and ball movement. Furthermore, we add the sound effect to make the game looks more interesting. Other than that, we also implement rule-based AI to make our games more fun.

WORK ASSIGNMENTS

GAME MECHANICS : Mainly done by 林家合

DISPLAY & UI : Mainly done by 邱世綸

PROBLEMS ENCOUNTERED & SOLUTIONS

- The update timing is not synchronized. In a game framework, there is usually an update function that can update the game state synchronously. In this project, there are at least three clock cycles that work in different timing. For example, game display, game state and sound system are not working in the same clock cycle. This can be a disaster when passing signal from one module to another module, so in order to solve this problem, we use game state to synchronize the modules and counter to match other modules update timing.

CONCLUSION

Working on this project is fun and frustrating. By spending with only a few hours, we can easily implement this in lua, and achieve the same outcome. However, implementing this in Basys3 Artix-7 FPGA is really a challenge. It lacks a game framework that solves physics and does not have a synchronized update system. We encountered many problems and solved them respectively. After solving the problems, the game looks awesome and fun to play with.