Break The Bricks

Project Design

CSEE 4840 Embedded System Design

Prof: Stephen A. Edwards

Lianhao Qu(lq2140)
Chong Li(cl3064)
Bin Xie(bx2123)
Zimeng Chen(zc2240)
Yuanhan Yang (yy2437)

I. High-level overview of the project

This project is aimed to build a classic video game named **Break The Bricks**.

As shown in Figure 1, the game screen on the left part of the screen in the game zone, the player can control the paddle with the incremental rotary encoder to prevent the ball from hitting the bottom of the screen. Remove all the bricks to level up.

On the right part of the screen is the status zone with scoreboard, life bars and level mark. The scoreboard shows how many bricks the player already broken; life bar shows the number of chances the player remains; and the level mark shows the level of the game.

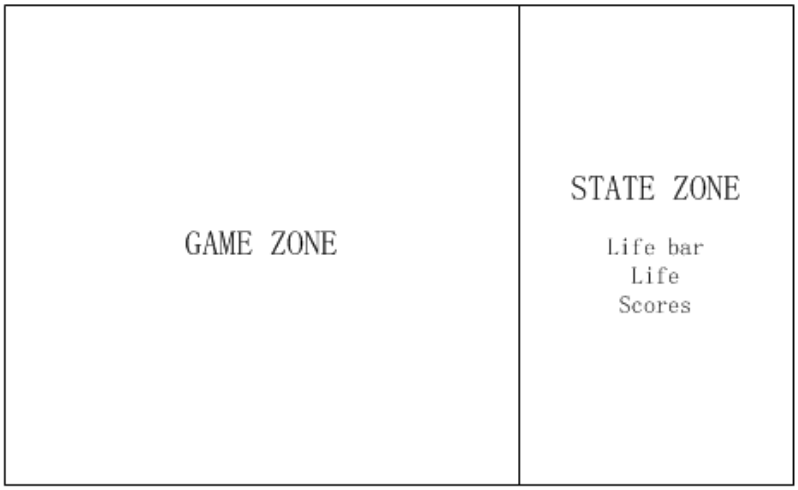

Figure 1. The game screen

Basic Realization

- Players control the paddle using rotary controller.
- Game interface is shown on the VGA.
- The player can start and pause during the game.
- The collision between the paddle and the ball is specular reflection.
- The initial start up speed is random.
- Theoretically, the ball can pass every site in the game zone; there is no bug point.

II. A high-level block diagram of all the hardware components and how they interact

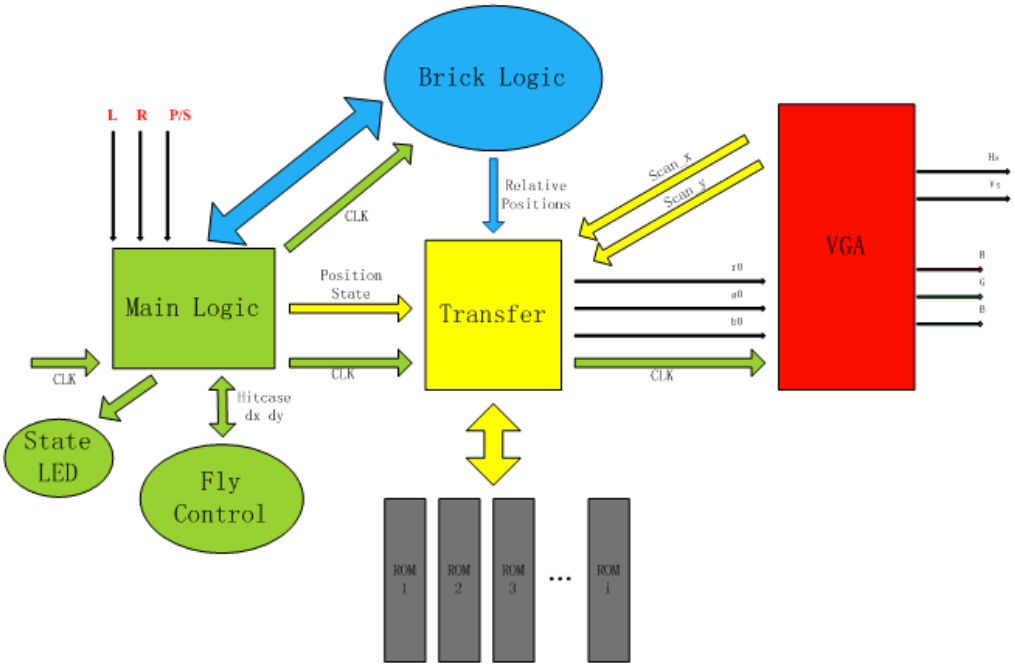

Figure 2. The block diagram of the system

III. An analysis of the memory requirements of your system

This project uses ROM to store all shapes in this game except for the screen frame. The ball, paddle, bricks, numbers and letters are all stored in ROM. We use MegaWizard Plug-In Manager tool to create ROM by transferring the .mif files.

IV. A detailed discussion of each unit

Main Logic Controller:

This main logic controller can handle the input signal from the incremental rotary encoder. They are left signal, right signal, start/pause signal and reset signal. It can also control how the status of the game switches between start status, pause status, reset status and stop status. Besides these, the main logic controller will calculate the situation of the paddle and bricks by interactive with Fly, Transfer and Brick controller.

The incremental rotary encoder has two channels and one push switch. When the rotary controller is rotated, A and B, as two input signals will produce a sequence listed in the figure 3. Based on the rotation direction the signal on encoder pin B will be either low for clockwise turns or high for counter clockwise turns. [1]One click of the push button generates the start/pause signal. Double click of the push button generates the reset signal. We define double click by receiving exactly two push signals within 0.1 ms after a software denounce process.

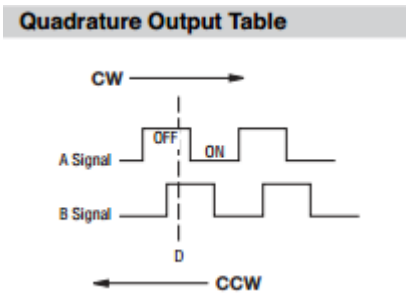

Figure 3

Fly Controller:

This Fly controller will receive the 2D location of the ball, and check whether the ball hits with the paddle or the bricks. And then calculate the moving direction and the speed of the ball after any hitting.

Transfer Controller:

This Transfer controller will receive the 2D location of paddle and ball from Main logic controller as well as 2D location of bricks from Brick controller. It will also read the graphic data from ROM, According to these locations, generate the corresponding RGB data with a 25Mhz's clock.

Brick Controller:

This Brick controller will receive the 2D position of the ball, and according to the position of the bricks, handle the collision effect between the ball and brick, and return the hitcase of collision.

VGA Controller:

The VGA controller will be based on what we learned in Lab 3. It will receive the RGB data from Transfer controller, draw the image and sync signals with a 25Mhz's clock.

V. I/O peripherals and data paths

1. Main logic unit

   This is the central part of the system. It handles all the signals coming from outside peripherals and controls each part of the system. It also deals with all the states such as pause, start and game over. There are 4 outputs and 4 inputs.

peripheral	data type	function
clk0	in std_logic	50MHz clock
pause	in std_logic	pause by hitting the button
left	in std_logic	left route
right	in std_logic	right route

R/G/B	out std_logic	color signal control
hs/vs	out std_logic	screen scan
state_led	out std_logic_vector(2 downto 0)	system state signal
hit_led	out std_logic_vector(3 downto 0)	ball hitting signal

2. Brick unit

peripheral	data type	function
clk	in std_logic	clock
life	in std_logic_vector(1 downto 0)	life bar
level	in std_logic_vector(1 downto 0)	level
state	in std_logic_vector(2 downto 0)	state
score	in std_logic_vector(7 downto 0)	score
bar_location	in std_logic_vector(9 downto 0)	the x location of the left terminal of the paddle
bar_length	in std_logic_vector(9 downto 0)	the length of the paddle
ball_x	in std_logic_vector(9 downto 0)	x location of the ball
ball_y	in std_logic_vector(9 downto 0)	y location of the ball
scan_x	in std_logic_vector(9 downto 0)	x location of the scan
scan_y	in std_logic_vector(9 downto 0)	y location of the scan
brick_dx	in std_logic_vector(9 downto 0)	relative x location of the brick
brick_dy	in std_logic_vector(9 downto 0)	relative y location of the brick
r0	out std_logic	the r of this point
g0	out std_logic	the g of this point
b0	out std_logic	the b of this point

3. Transfer unit

The function of transfer unit is to control the VGA screen, transferring the position of one pixel from (scan_x, scan_y) to the color of it by (R, G, B). There are 7 inputs and 3 outputs.

peripheral	data type	function
clk	in std_logic	clock
velocity_dx	in std_logic_vector(9 downto 0)	the horizontal component of the velocity
velocity_dy	in std_logic_vector(9 downto 0)	the vertical component of the velocity
ball_x	in std_logic_vector(9 downto 0)	x location of the ball
ball_y	in std_logic_vector(9 downto 0)	y location of the ball
scan_x	in std_logic_vector(9 downto 0)	x location of the scan
scan_y	in std_logic_vector(9 downto 0)	y location of the scan

brick_dx	out std_logic_vector(9 downto 0)	relative x location of the brick
brick_dy	out std_logic_vector(9 downto 0)	relative y location of the brick
hitcase	out std_logic_vector(3 downto 0)	the situation of the hit