Digital Logic Project

Ping Pong Report

Monica Luna Estrada
Wen Xuan Xie
Karmen Yung

January 3, 2017

Section 1:   Introduction

In this logic project, our group designed a VGA-based two player traditional Ping Pong game using Verilog HDL. The game is designed for Kintex 7 SWORD Board (with Xilinx Spartan-6 FPGA), visualized with a VGA port connected to a computer screen, and controlled with a set of buttons on the board. The game is created for two players to play, so that the two players each control their own paddle to beat against each other for achieving high score.

Section 2:   Design specification

2.1 Design Environment

Experimental Device: Kintex 7 (Compatible with programs running on Spartan 6)
Developing Software: Xilinx ISE
Developing Language: Verilog ISE

2.2 Inputs and Outputs

Inputs: The buttons on Kintex 7 Board
Outputs: The game generated from the program is visualized on the computer screen connecting the VGA-port to the processing board.

2.3 The core design – Ping Pong Game

Like the traditional visual style that we usually see at an arcade game center, our ping pong game has a black background and comprises two main features, a ping pong ball and a pair of rectangular paddle appears vertically located on the side of the screen opposite to each other. The game is played by two players, one player controls the left paddle, and the other player controls the right one. Here are the following mechanisms which build up the game:

- The game starts immediately once the software successfully running the program on the device.
- The ping pong ball appears right at the center of the computer screen, along with the left paddle and right paddle.
- Players may start the game as the ping pong ball begins to move. The players move the paddles using the buttons on the device.
- When either player misses the ball, the game pauses and a new ball is provided. The ball always moves toward the right paddle for each time it is created.

To become a skillful pong player, players not only have quick reaction over the incoming ball's movement, but also have strong prediction to the directions the ball will bounce from the opponent's paddles. Those skills collaborate with the control of the switches which move the paddles. The pong game has an additional feature where players can enlarge and shrink the sizes of their paddles, which can greatly influence the atmosphere of the game.

Section 3: Model Design

3.1 The Main structure of Ping Pong Game

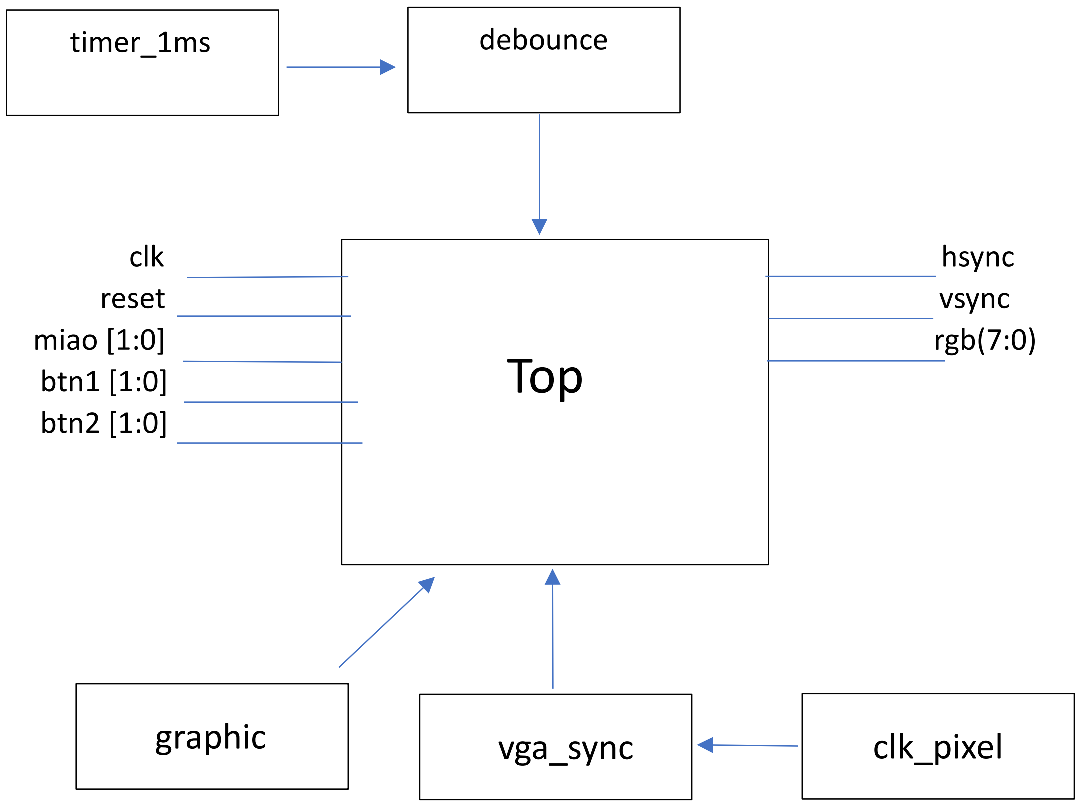


Figure 3-1: Image Representation of the Ping Pong Project

This design has a big top module, which contains all the important components that makes up the game. There is a set of four debounce modules, each of them includes a timing module timer_1ms. There is a graphic module, which is responsible for making up the shape, size, and colours, coordinates, and controls of every object that will appear on the computer screen. The notables are ping pong ball and paddles. Finally, vga_sync is what allows the elements of the design to be visualized.

3.2 Debounce module

Descriptions

Usually, when a user presses a button on an electronic device, there is an easy chance that bouncing happens where the tendency of any two metal contacts in an electronic device to generate multiple signals. To ensure the device generates only one signal for each button pushed, debounce is created. Debounce enforces that only a single signal will be acted upon for a single opening or closing of a contact. Debouncing enforces that a function not be called again until a certain amount of time has passed without it being called. The time components of this module associates with timer_1ms.

Input

- clk: the clock impulse generated by the software
- button: this variable determines whether the button is hit by the user.

Output

- click: Determines whether the information sent by the pushed button is delivered to the device.

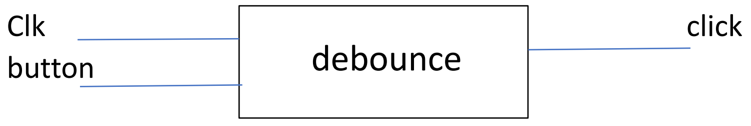


Figure 3-2: Image Representation of “debounce” module

3.3 Time Module “timer_1ms”

Descriptions

This time module makes the program to generate a period for each 1 milliseconds passed.

Input

- clk: the clock impulse generated by the software

Output

- Generate a time period after each 1 milliseconds has passed

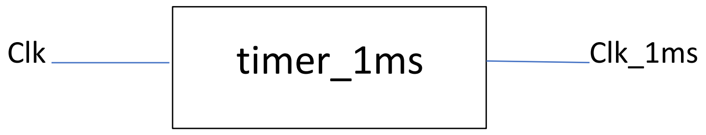


Figure 3-3: Image Representation of timer_1ms” module

3.4 “graphic” Function

Descriptions

This module creates the game physics and graphics display which makes up the game. The screen, as well as the size and color of the ball and paddles are all constructed here. Besides their looks, we also implemented game mechanisms in this portion, such as the velocities of the ball, control movements of the paddles, collisions characteristics and so on. In addition to all this, the black background colour is also declared within this module. Last but not least, all this information will be packed up and prepared to show up on the computer screen under the name rgb(7:0).

Inputs

- Btn1(1:0): The up and down buttons for the left paddle
- Btn2(1:0); The up and down buttons for the right paddle
- Miao(1:0): Those two buttons change the size of the paddles
- X(10:0): Horizontal pixels for reference
- Y(10:0): Vertical pixels for reference
- Clk: the clock impulse generated by the software
- Reset: This button shuts down the running program

Outputs

- Rgb(7:0): The components are packaged into this output variable ready to be displayed on the screen

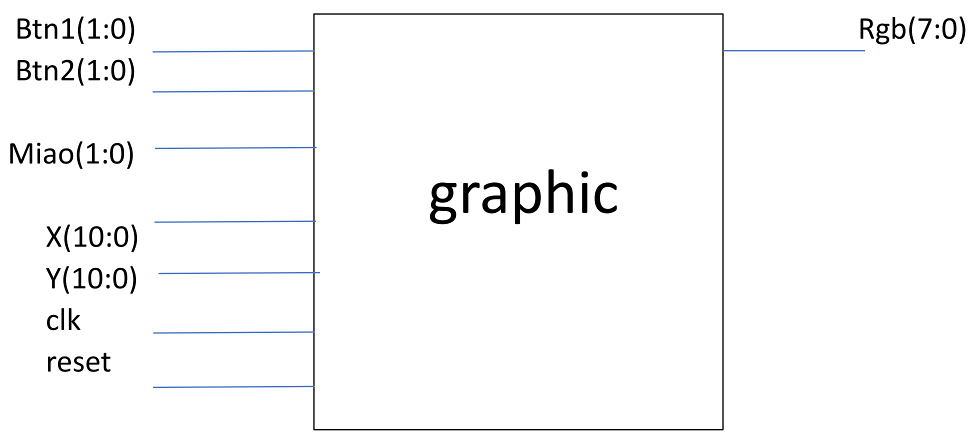


Figure 3-4: Image Representation of “graphic” function

3.5 “vga_sync” Function

Descriptions

This module synchronizes the program onto the computer display through VGA port. The game begins as a result.

Input

- Clk: the clock impulse generated by the software

Output

- Hsync is "Horizontal Sync", synchronizes the start and finish of the horizontal picture scan line in the monitor with the picture source that created it, via a negative pulse in each case.
- Vsync is the equivalent vertical synchronization, it ensures the monitor scan starts at the top of the picture at the right time.
- X: output the maximum horizontal pixels a computer screen will display
- Y: output the maximum vertical pixels a computer screen will display

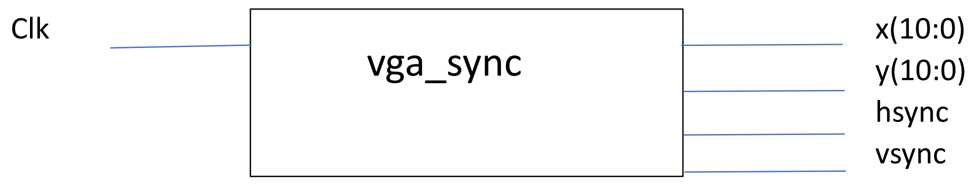


Figure 3-5: Image Representation of “vga_sync” function

3.6 clk_pixel

Descriptions

This module serves as a sub-component of vga_sync. Its purpose is to generate a 25MHz pixel clock that suits the best for the computer resolution.

Input

- Clk: the clock impulse generated by the software
- Clk-pixel: the finial generated time settings

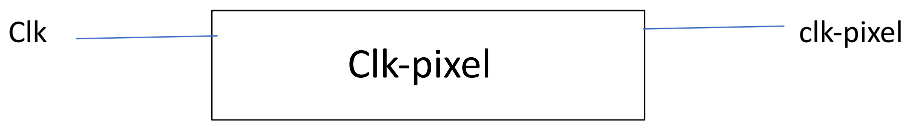


Figure 3-6: Image Representation of “clk-pixel” function

3.7 The Top Module

Descriptions

The top module behaves like a main function in computer language. It contains statements which declares all the essential modules implemented in this project. Each declaration contains various parameters initialized in this module, which are also served as the inputs and outputs for the project in general.

Input

- Clk: clock generated by the software
- Btn1(1:0): Control buttons of the left paddle
- Btn2(1:0): Control buttons of the right paddle
- Miao(1:0): Size adjustment switches for both paddles
- Reset: An option to shut off the game

Output

- Hsync: Horizontal Sync
- Vsync: Vertical Sync
- Rgb(7:0): Colour Visualization

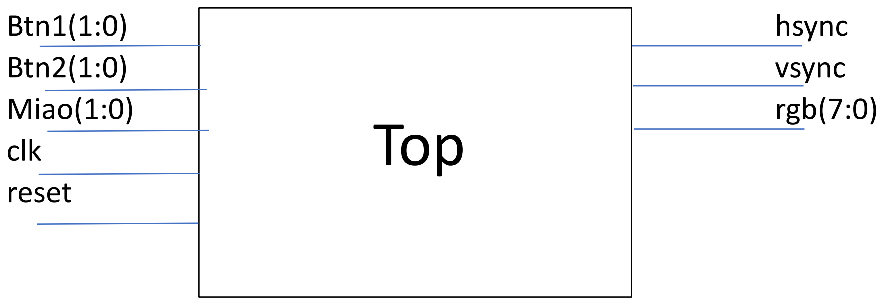
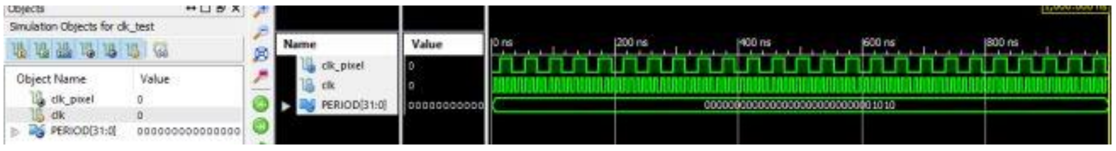


Figure 3-7: Image Representation of the Top module

Section 4: Simulation

First is the debounce waveform simulation.

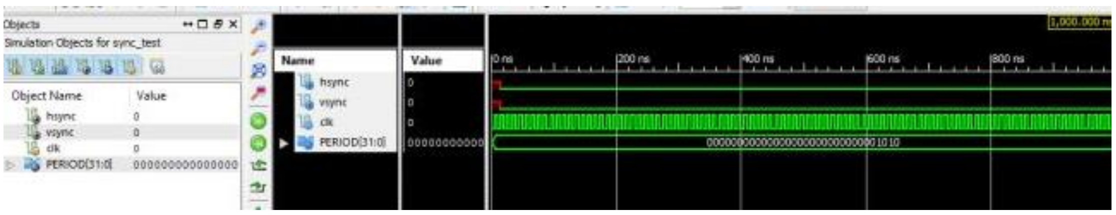


Object Name	Value
clk_pixel	0
clk	0
PERIOD[31:0]	000000000000000

Name	Value
clk_pixel	0
clk	0
PERIOD[31:0]	00000000000

```
00000000000000000000000000001010
```

Second is the vga_sync waveform simulation: The negative pulses at the beginning of hsync and vsync start the drawing process which allows for the game to display.



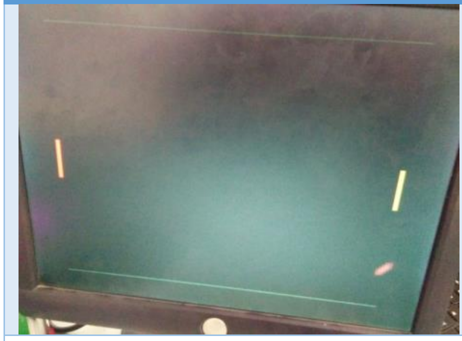
Section 5: Debug and analysis

The pong game consists of a ball bouncing on a screen. Two paddles on opposite sides of the screen (controlled using buttons on the SWORD board) enables the users to make the ball bounce back and forth across the screen.

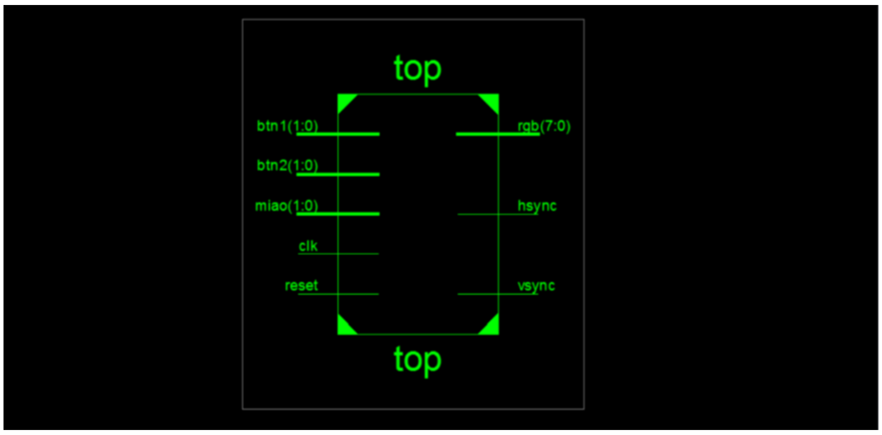
The first problem encountered was that no picture would display when the program was run, this was due to the incorrect pinning of signals. To create a VGA video signal from FPGA pins, a VGA monitor requires five signals to display a picture. These are R (red), G (green), B (blue), HS (horizontal synchronization), and VS (vertical synchronization); where R, G, and B are analog signals and HS and VS are digital signals. To debug we rearranged the pins following a schematic we found. The final arrangement resulted in pins 1, 2, and 3 used as (R, G, and B); using these pins with different combinations of 1’s and 0’s results in different colors. Pins 13 and 14 are used for the digital HS and VS signals. Finally pins 5 through to 10 are used as pins to ground.

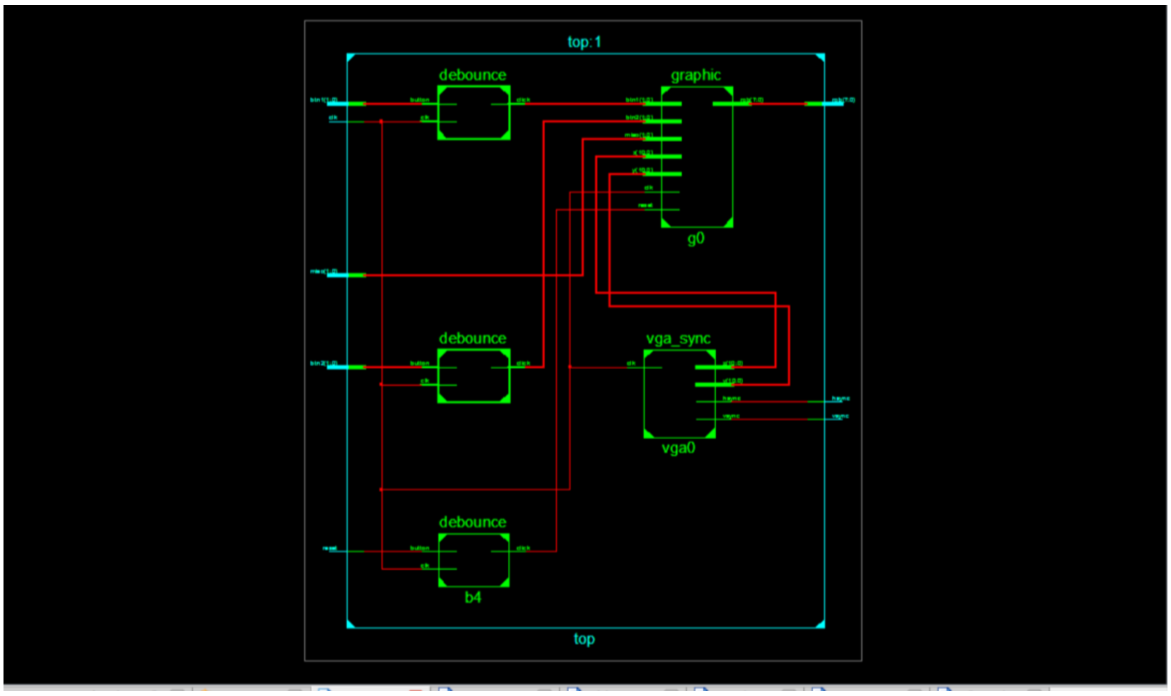
The second problem was encountered when testing the game. Pushing the respective set of control buttons for each paddle should result in a paddle moving up or down, however this would only work sometimes. The buttons needed to be pushed multiple times before a paddle would move, a paddle would move when its controls were not being used, there would be large jumps when paddles moved, or there would simple be no movement at all. It was concluded that this was due to interference; since the buttons are mechanical devices, when they are pushed there is a possibility that the contacts will not simply go from open to closed creating “bounce”, where for a short time after a button is pushed the reading from the IO pin will toggle between 1 and 0 before settling on a value. The problem was fixed by adding the Debounce module in conjunction with timer_1ms. This requires that for a button to register as pressed it must give the appearance of being pushed for a set amount of time and that only one final signal per button push is registered. In addition to this, we then mapped the buttons (btn1[0], btn1[1], btn2[0], and btn2[1]) to switches instead so that instead of having to push a button multiple times to get it to move up or down continuously, the switch simply had to be flipped to the correct direction.

Test (use of controls):

Game start	
	
Move left paddle up	**Move left paddle down**
Increase thickness of paddle	**Increased height of paddle**

5.1 Debug Processes with Schematic Visuals





5.2 Device's Features Representations and Fucntionalities

I/O Type	Feature	Control Value	Meaning
Input	Btn1[0]	Press the button	Left paddle moves downward
	Btn1[1]		Left Paddle moves upward
	Btn2[0]		Right Paddle moves downward
	Btn2[1]		Right Paddle moves upward
	Miao[0:1]	00	No changes with the size of paddles
		01	Enlarge width of paddles; height stays the same
		10	Enlarge height or paddles; width stays the same
		11	Enlarge both height and width of paddles.
Output	RGB[7:0]	None	The objects (ball and paddles) behaves as the button commands as requested by the users
	Hsync	None	Syncronization and Visualization Procedures
	Vsync		

5.3 Controlling Instructions

Once the program is successfully implemented to the electronic device, the ping pong game appears on the screen and the users may start playing right away. The users may use the buttons on the device which corresponds to the variables in the UCF to move their paddles toward the upcoming ball. The users are free to change the size of their paddles using switches that represent the Miao[1:0] if they wished to do so.

Section 6: Conclusion

The overall design is fairly simple and straightforward, with the most difficult parts being the graphical display. An object-mapped pixel generation circuit is used for implementation; to keep track of objects positions on the screen, for each of the objects represented (left paddle, right paddle and ball), two variables are used: current position and next position. When the clock is set to one, the next position data is transferred to the current position variable which then causes the graphics generation module output the correct VGA signals. Additional features added that changed the paddle size were more for visual effect and do not really affect the overall difficulty of the game (although perhaps a longer paddle makes it easier). Problems encountered were wiring, difficulties displaying the game on screen, and game controls. One large improvement which could have been made is mapping the paddle controls to switches that are easier for two players to access at once. Further improvements that could be added are a scoreboard that terminates (instead of the current version which simply increments until manually cleared), sound, and different levels of difficulty for the user(s) to choose from.

Section 7: Team member and contribution

Monica Luna Estrada: 33% - Programming, debugging, simulation and proofreading
Wen Xuan Xie: 33% - Programming, debugging, and report writing
Karmen Yung: 33% - Resource finding, report writing, and proofreading

Section 8: References

[1] Nicolle, Jean. P. (2005 May 10). *Pong Game*. Retrieved from http://fpga4fun.com/PongGame.html

[2] No Name. (2007 June 19). *Pong Game-VGA*. Retrieved from http://www.fpgacenter.com/examples/vga/index.php

[3] Beall, J. Katzin, A. *Pong Final Project Report.* 2006. Harvey Mudd College. PDF file. 26 December 2016

[4] Shi, Qingsong. "逻辑与计算机基础." May 2008. PowerPoint presentation.

[5] Eastwood, Eric. (2014). Pong on a FPGA [Blog Post]. http://ericeastwood.com/blog/9/pong-on-a-fpga

[6] Chu, Pong. P. (2008). FPGA Prototyping by Verilog Examples. John Wiley & Sons, Inc.