Digital Logic Project

Flappy Bird Design Report

陈博文
舒义然
朱曼青

Date: 2017-01-08

Section 1: Introduction

In this project design, our group managed to produce a game named Flappy Bird. Flappy Bird is a 2013 mobile game developed by Vietnamese programmer. Using Verilog HDL, our group not only implement the most function of the game, but also modify it to increase some attractive elements.

The game is a side-scroller where the player controls a bird, attempting to fly between rows of pipes without hitting them. The title of the game is displayed on the right half of the screen and the scene for game is on the left half. The bird briefly flaps upwards each time that the player taps the jump button. If the player doesn't tap the jump button, the bird falls because of gravity. The bird is dynamic while flying, you can see the action of its wing. Every time when it passes through a pair of pipes, the player earns a single point. The mark of the player is displayed immediately by the seven-segment LEDs. And after passing through a certain number of pipes, the graph of background will make a change. There are two different pieces of scenery--one for daytime and the other for night. The player can choose the model of the game for three different game modes. The moving velocity of the pipes varies under different game models. Besides, the player can make a pause and reset the whole game whenever he or she likes. If the bird touches the pipe, the game loses and a "game over" logo will be sent to the screen. The player can decide whether to play the game again.

Figure 1 the screen of the game

Section 2: Design specification

A. development environment

System Boards: Nexys 4 DDR Artix-7 FPGA
Development environment: Xilinx ISE 14.7
Hardware Description Language: Verilog HDL

B. input/output interface

input: Keyboard (through USB interface)
output: VGA screen, the seven-segment LEDs on the system board

C. the design of the core module

As it's shown in the Figure 1 below, the core module is connected to three input/output modules, which is the VGA, keyboard and LEDs display. The interaction between the game and the player is mainly through the VGA screen and keyboard. The scenes of the game are sent to the VGA screen and the instructions from players are received by the keyboard. The mark of the player get in the game is displayed by the LEDs on the board.

There are four kinds of instructions can be sent to the game. The Reset, for both start and restart the game. The Pause, to stop the game in the process of playing. The Jump, to make the bird flappy upwards. And the selection buttons for three game models of different degree of difficulty.

The major thought of our design is to change the behavior of the game the state of the game. For example, we stop the clock signal of game when the bird hits the pipe to make the game loses. And the pause instruction also is implemented by stopping the clock signal.

The biggest challenge in our digital design is to display and change the movement of the bird and pipes appropriately. To solve this problem, we have written a lot of modules in our project file to deal with the change of positions of the bird and pipes.

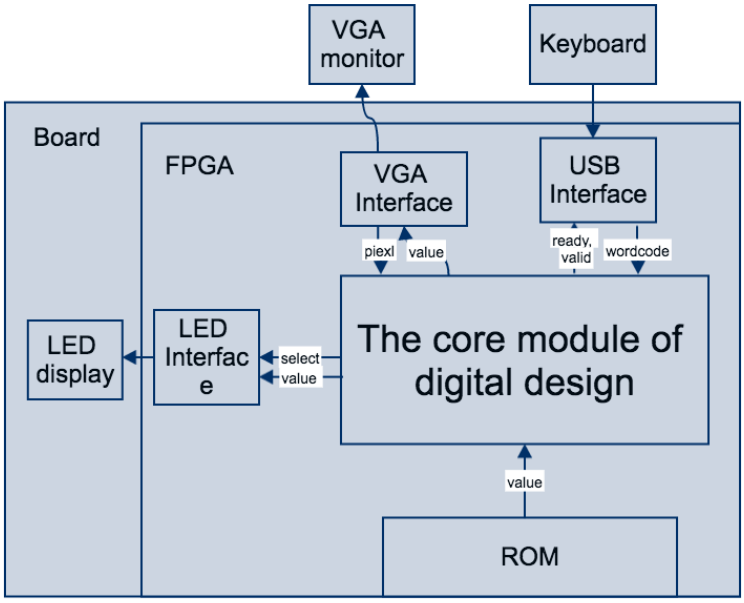


Figure 2 the Overall structure of our design

To be friendly with players, we design the help function. By pressing “H” on the keyboard, the player can see the description of operations in the game.

Figure 3 the description of operations in the game

We use LED segments on the board to display the mark number of the player. The background of the game will change after the player get a certain mark.

Figure 4 the mark displayed by LED and the change of the background

The player can make a pause while playing the game.

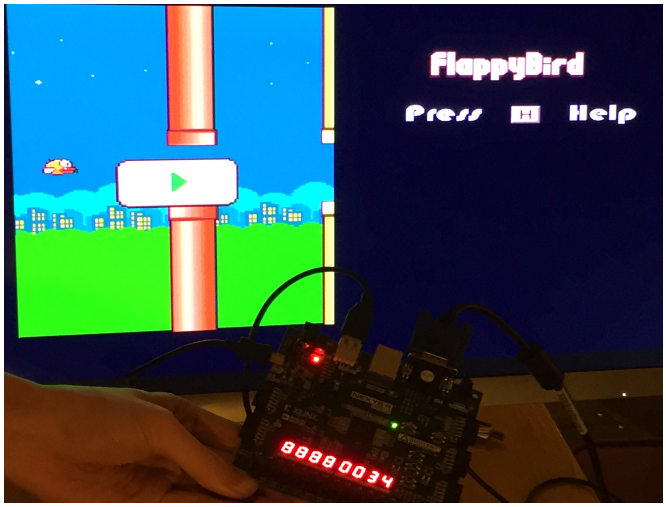


Figure 5 the pause function

When the bird hits the pipe, the game loses.

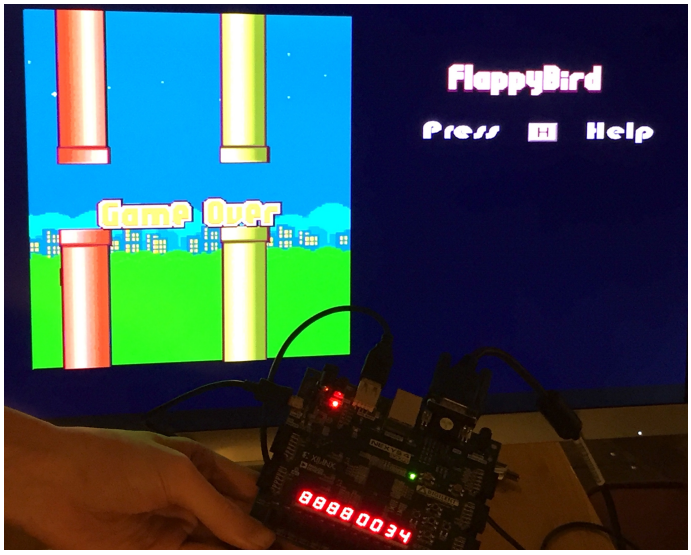


Figure 6 the game loses

Section 3: Model Design

A. the relation graph of modules

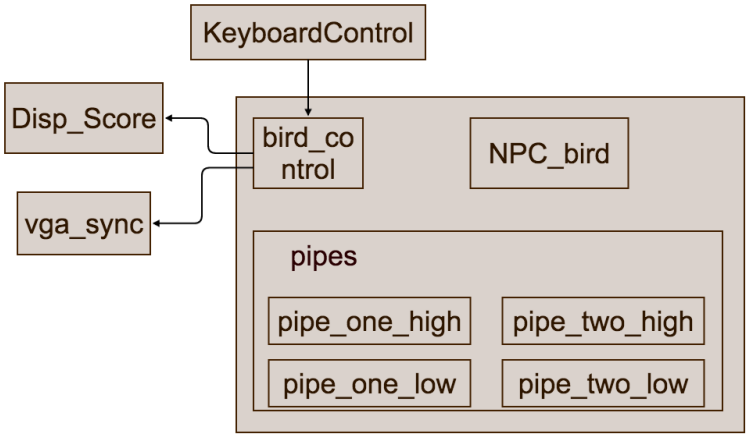


Figure 7 the relation graph of modules in the project file

B. Bird_control ---- the top module

input:

system_clk: the clock signal
reset: the signal for whether to start or restart the game
btn: the signal for the jump operation of the bird
ps2Dat: the serial input of the keyboard
ps2Clk: the clock signal of the keyboard
sw: the signal for pause

ouput:

AN[3:0]: the selection of the 4 bit numbers for LEDs display
Segment[7:0]: the output value of the seven-segment LEDs
rgb[11:0]: the color information for the scene of the game
hsync: the horizontial pixel for printing the scene on the VGA screen
vsync: the vertical pixel for printing the scene on the VGA screen

function:

As the top module of the project file, call other modules to work together. Control the state of the whole game.

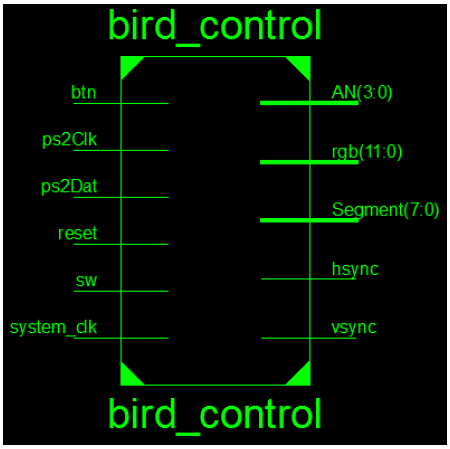


Figure 8 the bird_control module

C. NPC_bird ---- the module to generate the position of the bird

input:

pixel_x[9:0]: the current horizontal write position for the VGA screen
pixel_y[9:0]: the current vertical write position for the VGA screen
btnup: the signal for the jump operation of the bird
game_clk: the clock signal to control the state of the game
status: the signal for whether the bird hits pipes
system_clk: the clock signal
reset: the signal for whether to start or restart the game

ouput:

bird_pic_b[9:0]: the current position of the bottom of the bird
bird_pic_t[9:0]: the current position of the top of the bird
bird_pic_r[9:0]: the current position of the right side of the bird
bird_pic_l[9:0]: the current position of the left side of the bird
rgb_bird[11:0]: the color information for the bird

function:

As the control module for the bird, output the position of the bird under different state.

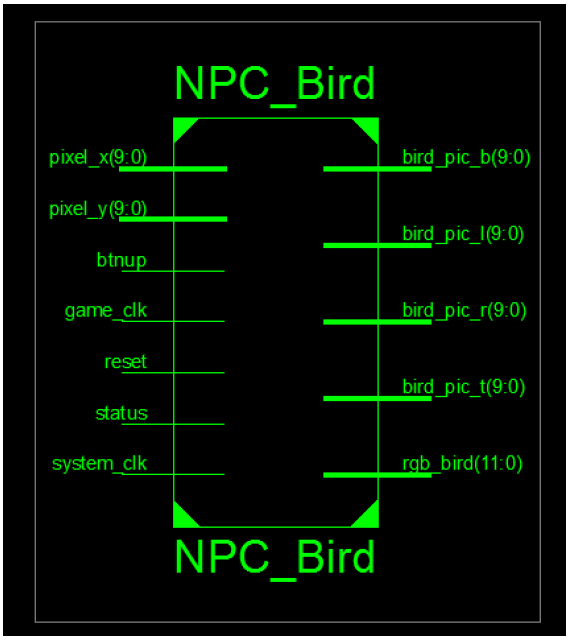


Figure 9 the NPC_Bird module

D. pipe_one_high pipe_one_low
pipe_two_high pipe_two_low ---- modules to generate the position of pipes

input:

For, higher pipe:

pixel_x[9:0]: the current horizontal write position for the VGA screen
pixel_y[9:0]: the current vertical write position for the VGA screen
game_clk: the clock signal to control the state of the game

system_clk: the clock signal
reset: the signal for whether to start or restart the game

For lower pipe:

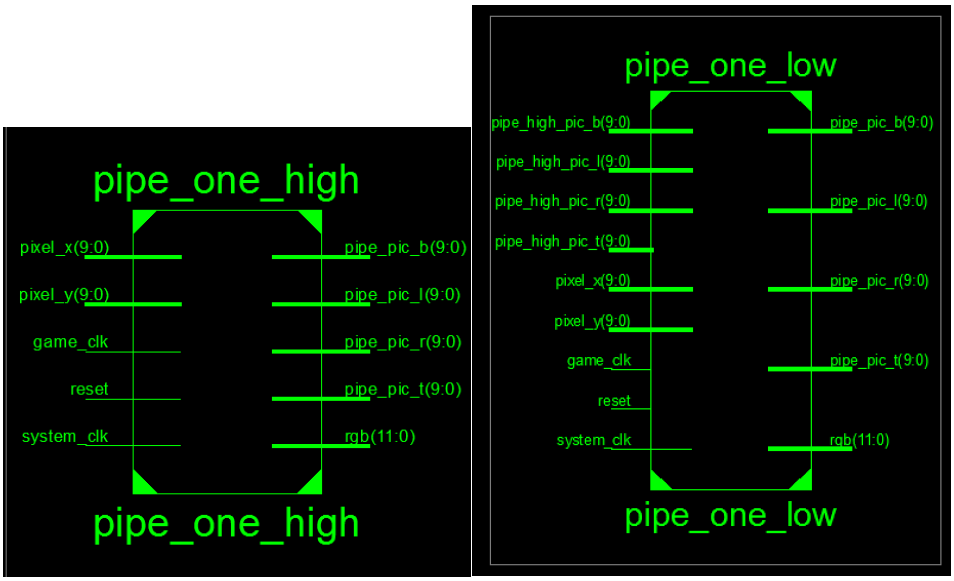
pipe_high_pic_b[9:0]: the position of the bottom of the higher pipe
pipe_high_pic_t[9:0]: the position of the top of the higher pipe
pipe_high_pic_l[9:0]: the position of the left side of the higher pipe
pipe_high_pic_r[9:0]: the position of the right side of the higher pipe
pixel_x[9:0]: the current horizontal write position for the VGA screen
pixel_y[9:0]: the current vertical write position for the VGA screen
game_clk: the clock signal to control the state of the game
system_clk: the clock signal
reset: the signal for whether to start or restart the game

ouput:

pipe_pic_b[9:0]: the current position of the bottom of the pipe
pipe_pic_t[9:0]: the current position of the top of the pipe
pipe_pic_l[9:0]: the current position of the left side of the pipe
pipe_pic_r[9:0]: the current position of the right side of the pipe
rgb[11:0]: the color information for the bird

function:

As the control module for the bird, generate the position of the pipe.



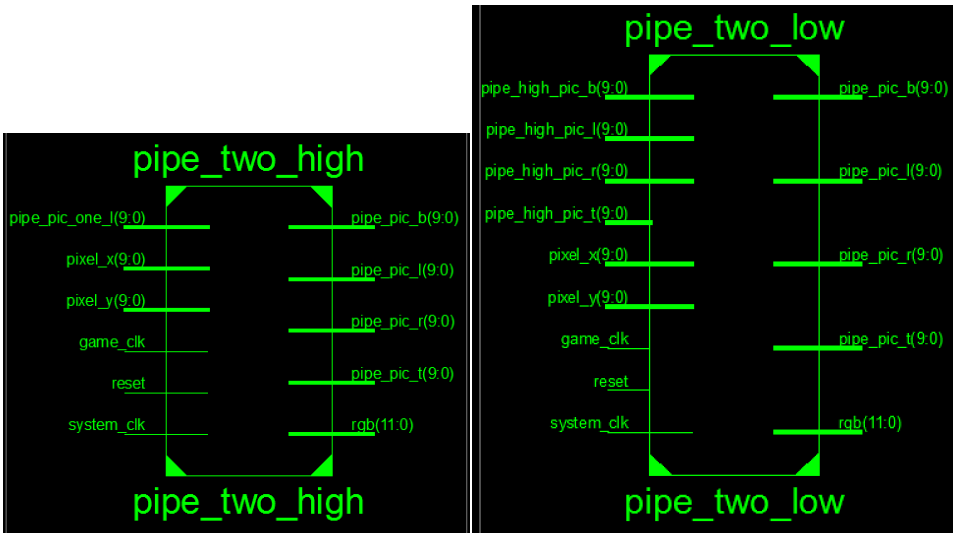


Figure 10 modules for pipes

E. vga_sync ---- the module to print the output the scene of the game

input:

clk: the clock signal

reset: the signal for whether to start or restart the game

ouput:

pixel_x[9:0]: the current horizontal write position for the VGA screen

pixel_y[9:0]: the current vertical write position for the VGA screen

hsync: the horizontial pixel for printing the scene on the VGA screen

vsync: the vertical pixel for printing the scene on the VGA screen

p_tick: the clock signal for the VGA display

video_on: the signal for whether it is in the display area

function:

As the VGA scan module, to output on the VGA screen.

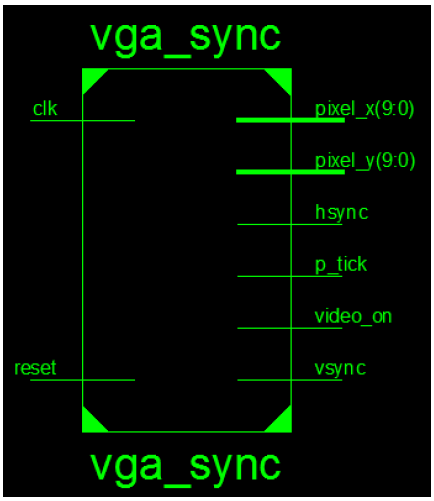


Figure 11 the vga_sync module

F. KeyboardControl ----the module to accept instruction

input:

btn: the input signal from the keyboard
clk: the system clock signal
reset: the signal for whether to start or restart the game
clk_25: the 25th signal derived from clkdiv
ps2Dat: the serial input of the keyboard
ps2Clk: the clock signal of the keyboard

ouput:

btnup: output the instruction of the player

function:

The module is used to receive instruction from the key board.

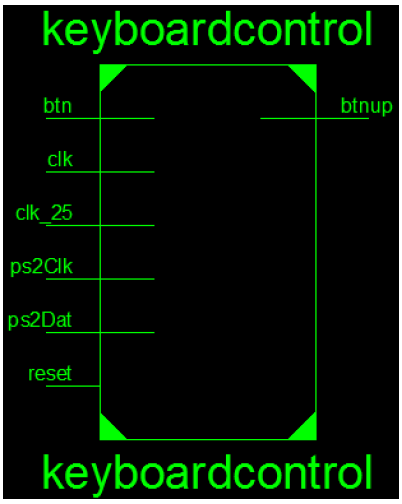


Figure 12 the keyboardcontrol module

G. Disp_Score ---- the module to display mark

input:

bird_pic_b[9:0]: the current position of the bottom of the bird
bird_pic_t[9:0]: the current position of the top of the bird
bird_pic_r[9:0]: the current position of the right side of the bird
bird_pic_l[9:0]: the current position of the left side of the bird
pipe_high_pic_b[9:0]: the position of the bottom of the higher red pipe
pipe_low_pic_t[9:0]: the position of the top of the lower red pipe
pipe_high_pic_l[9:0]: the position of the left side of the higher red pipe
pipe_high_pic_r[9:0]: the position of the right side of the higher red pipe
ypipe_high_pic_b[9:0]: the position of the bottom of the higher green pipe
ypipe_low_pic_t[9:0]: the position of the top of the lower green pipe
ypipe_high_pic_l[9:0]: the position of the left side of the higher green pipe
ypipe_high_pic_r[9:0]: the position of the right side of the higher green pipe
game_clk: the clock signal to control the state of the game
system_clk: the clock signal
reset: the signal for whether to start or restart the game

ouput:

AN[3:0]: the selection of the 4 bit numbers for LEDs display
Segment[7:0]: the output value of the seven-segment LEDs
score[15:0]: the mark of the player(output of the BCD code)

function:

The module to generate the mark of the player and display the mark number.

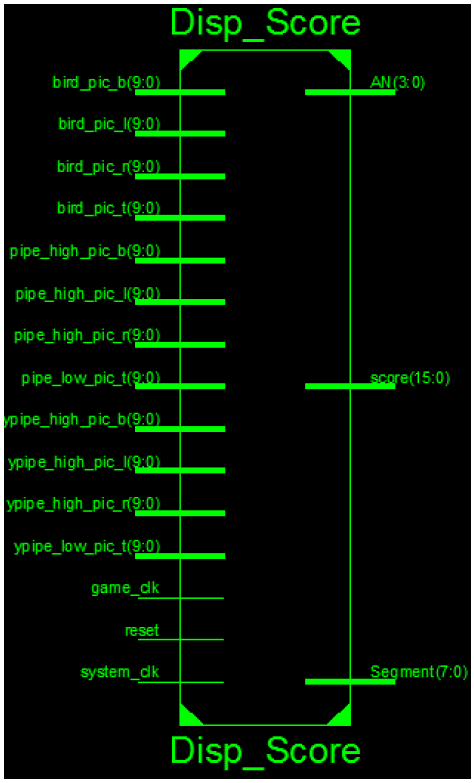


Figure 13 the Disp_Score module

Section 4: Simulation

A. the simulation for the Random module

The Random module is an important part of the module to generate the position of the pipe. Because there is no built-in mechanism to produce a random number in Verilog HDL. In this module, we use Verilog HDL to simulate the behavior of a shift left register to make the output numbers approximately in random.

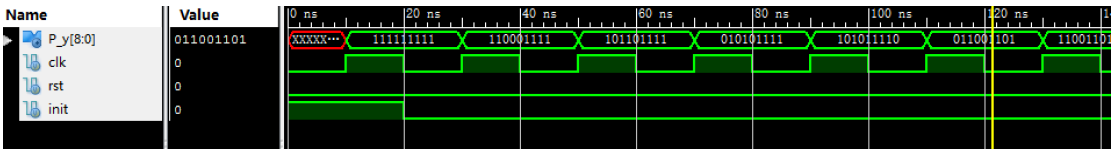


Figure 14 the wave graph of the Random module

The P_y[8:0] is the output of the Random module. The clk is the clock signal for

simulation. The rst is the reset signal, and the init is used to initialize the value of P_y[8:0].

The Random module will produce a new output at every positive edge of the clk signal. From 0 to 50 ns, the value of P_y[8:0] remains unsure because the clk signal is 0 and the module hasn't produced an output. At 50 ns, the clk changes into 1 from 0 and the init is 1, so the output P_y[8:0] is 9'b1_1111_1111. And after that, we can see the output of the module from the wave graph above. The value of the P_y[8:0] varies like that: 9'b1_1000_1111 => 9'b1_0110_1111 => 9'b 0_1010_1111 =>1_0101_1110 = > 0_1100_1101 => ……

Actually, what the Random module has done is to produce all the number in a scale from 9'b0 to 9'b1_1111_1111. Because the output changes every time the clock signal changes, we can get a approximately random number from this module.

B. the simulation for the NPC_Bird module

The NPC_Bird module is used to generate the position of the bird. The position of right and left side of the bird remains unchangeable. The bird's vertical position is designed to change as the clock signal changes. It's required to make a reaction to the jump button. The bird is designed to flappy upwards when the jump button is pressed. Otherwise, the bird will fall.

pixel_x[9:0] and pixel_y[9:0] are the current write position for the VGA screen, rgb_bird[11:0] is the color information for the bird. The status signal is used to control the clock signal for the game in other module. For the convenience of test, these signals are set as constant 0 in the simulation.

The system_clk is the clock signal. The game_clk is the clock signal to control the state of the game. The frequency of these two clock signals is same.

The other two input signals reset and btnup are used to control the state of the bird. The reset signal is used to make the bird return to its initial position, where the top position of the bird is 'd235. The btnup signal is designed to implement the jump operation of the bird. The coordinate number of the bird will decrement when the btnup signal is 1.

bird_pic_b[9:0] and bird_pic_t[9:0] are the output signals for vertical coordinate of the bird.

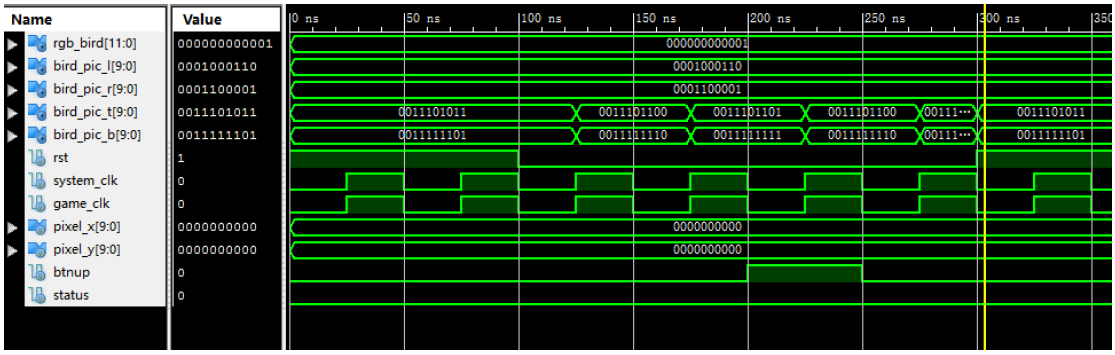


Figure 15 the wave graph of the NPC_Bird module

The vertical coordinate of the bird is produced at the positive edge of the clock signal. From 0 to 100 ns, the bird_pic_t and the bird_piuc_b remain unchangeable because the rst is 1. After then, the rst is set as 0 and the value of bird_pic_t and bird_piuc_b can be changed when game_clk and system_clk are 1. We can see the graph that from 125 ns to 200 ns, the btnup is 0 and the vertical coordinate of the bird is increasing. So the bird in the VGA screen will fall. From 200 ns to 250 ns, the btnup is 1. So the coordinate decrement after the clock signal changes into 1 at 225 ns.

C. the simulation for the Disp_Score module

This module is to calculate and display the score. In this module, there are 15 input ports. game_clk is the clock for the whole game, while system_clock is the clock for the whole operation system. While the game is stopped, game_clock will be stopped as well. The system_clock will provide the clock pulse all the time, no matter whether the game is stopped or not.

In this module, reset is of no use. bird_pic_b, bird_pic_t, bird_pic_l, bird_pic_r are variables to describe the state of the bird. They are bottom coordinate, top coordinate, left coordinate and right coordinate respectively. Other inputs, pipe_high_pic_l, pipe_high_pic_r, pipe_high_pic_b, pipe_high_pic_t, ypipe_high_pic_l, ypipe_high_pic_r, ypipe_high_pic_b, ypipe_high_pic_t are variables to describe the state of the pipes. They are also left coordinate, right coordinate, bottom coordinate and top coordinate respectively.

Our method to judge whether the bird passes the pipes is to test if

```
   ((bird_pic_l >= pipe_high_pic_l) & (bird_pic_r <=
   pipe_high_pic_r)& (bird_pic_t >= pipe_high_pic_b) &
   (bird_pic_b <= pipe_low_pic_t)) |  ((bird_pic_l >=
   ypipe_high_pic_l) & (bird_pic_r <=ypipe_high_pic_r)
   & (bird_pic_t >= ypipe_high_pic_b) & (bird_pic_b <=
ypipe_low_pic_t))
```

If so, a variable named pass is set to 1. If not, the variable is 0. If pass is 1, the

output score adds 1. If pass is 0, it is likely that the bird flies normally or the bird crashes into a pipe.

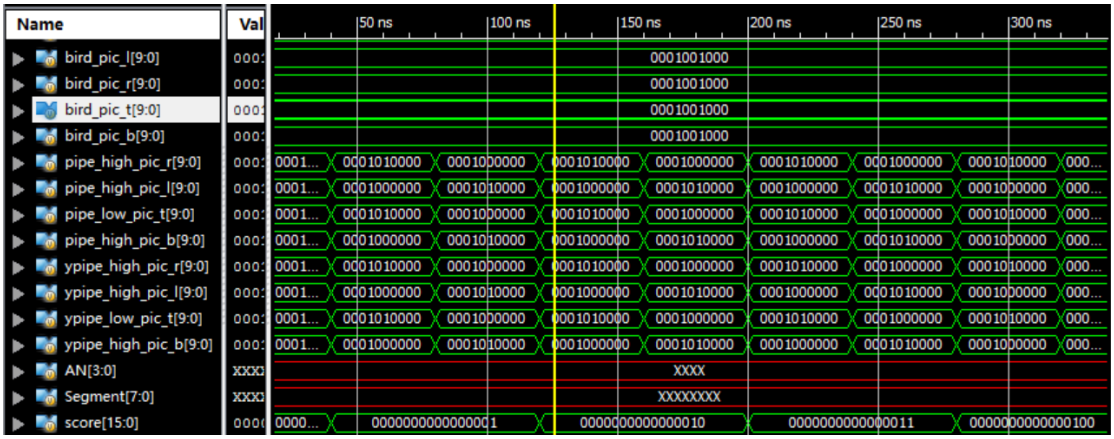


Figure 16 the wave graph for the Disp_Num module

As for the wave diagram, it is evident that AN and Segment are of no meaning. For the state (location) of the bird, the variables are set to a constant value (0001001000). As the state (location) of the pipes changes, sometimes the bird passes but sometimes the bird fails to pass. The state of the pipes changes each 40ns.

The value of the coordinate of the two pipes fluctuate between 0001010000 and 0001000000. In other word, each 80ns, the bird passes once and the score adds 1. According to the wave diagram, the score indeed adds 1 each 80ns, which reflects that the bird passes the pipes once each 80ns. In fact, the increase of the score takes place in

$$(2n + 1) * 40 \text{ ns} \quad (n = 0, 1, 2, \ldots).$$

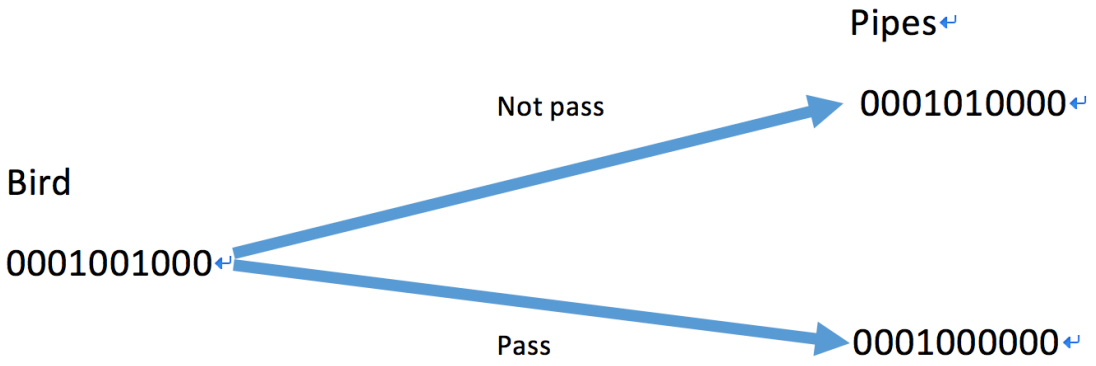


Figure 17 the relation between the coordinate of the bird and the pipe

D. the simulation for the pipe_one_high module

This module is to generate the higher part of the first pipe. There are 5 input ports in this module. They are pixel_x, pixel_y, game_clk, system_clk and reset.

Each function:

1. game_clk: The clock is for the whole game. While the game is stopped, game_clock will be stopped as well.
2. system_clock: The clock is for the whole operation system. The system_clock will provide the clock pulse all the time, no matter whether the game is stopped or not.
3. reset: To generate the pipes at the very beginning of the game.
4. pixel_x: The horizontal coordinate of the pipe.
5. pixel_y: The vertical coordinate of the pipe.

The output ports of this module are pipe_pic_b, pipe_pic_l, pipe_pic_r, pipe_pic_t and rgb.

Each function:

1. pipe_pic_b: The bottom coordinate of the pipe.
2. pipe_pic_l: The left coordinate of the pipe.
3. pipe_pic_r: The right coordinate of the pipe.
4. pipe_pic_t: The top coordinate of the pipe.
5. rgb: The color of the pipe (red).

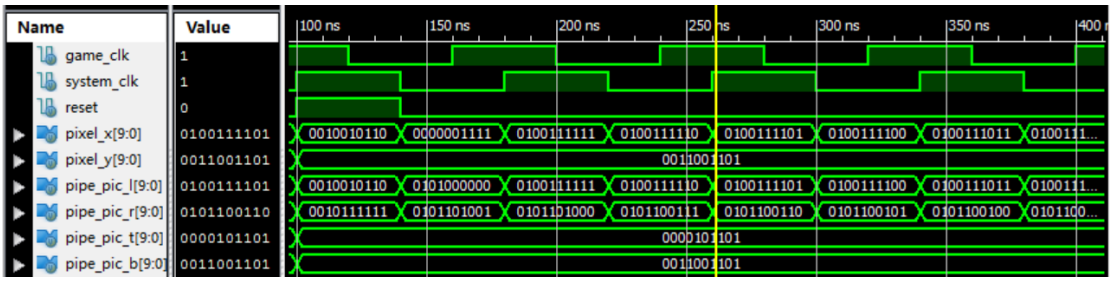


Figure 18 the wave graph of the pipe_one_high module

During 100~140ns, reset is 1, meaning the game just starts. The bottom coordinate depends on the random value. In this case shown as the wave diagram, the bottom coordinate is 205 (0011001101). The top coordinate is 45 (0000101101) all the time. The left coordinate is initialized as 150 (0010010110), while the right coordinate subtracts the left coordinate is 41 all the time. Thus the present value of the right coordinate is 191 (0010111111). Later reset is 0.

During 140~180ns, the left coordinate adds 41, smaller than the minimum (57), meaning that the pipe has moved to the very left side of the screen. So a new pipe appear on the right side later. The new pipe's left coordinate is 320 (0101000000) and its right coordinate is 361 (0101101001).

Later after each 40ns, the pipe move left by 1 pixel. Thus we can see during 180~220ns, the pipe's left coordinate is 319 (0100111111) and its right coordinate is 360 (0101101000). And during 220~260ns, the pipe's left coordinate is 318 (0100111110) and its right coordinate is 359 (0101100111). The rest can be done in the same manner.

Section 5: Debug and analysis

(1) Comprehensive analysis
After designing the circuit module, we enter the debug stage. At first we check the syntax. This part is similar with that of C Language. Syntax debug can tell the bug in our code very accurately. After syntax debug, we ran the RLT of the top module and analyze the top module and test whether the function is just as we expected. Later we analyze and skim the comprehensive system report, figuring out the usage of the system resource. Then we checked the errors and warnings of every module and eliminate the errors and warnings one by one.

(2) Debug
The sequence of debugging is NPC_bird module, keyboardcontrol module, pipe_one_high, pipe_one_low, pipe_two_high, pipe_two_low, pause module, deplay_score module, Disp_Score module, and vga_sync module. According to the error prompt from ISE, we eliminated variable definition errors, sentence errors. After the code was compiled successfully, write the UCF file, providing definitions of each pin. After that, we download the bit file onto Spartan III experiment board to test the project.

(3) Operation process
First start the game. The bird on the screen is subject to the gravity, gradually falling if the user doesn't press any button.
A. If the user presses space button, the bird rises.
B. Press A to accelerate the bird.
C. Press D to decelerate the bird.
D. If the user presses P, the game is stopped. Press P again to resume the game.
E. H is a special button. If the user presses it, the button direction is shown to him.

We find the game processes smoothly. The background graph is also performing as we have expected. The day is transferred to the night or the night is transferred to the day every time the bird passes 10 pipes. If the bird passes one pipe, the score on the segment adds 1. If the user operates improperly, leading the bird to crash into the pipe, the bird will disappear and "game over" will be printed on the interface.

(4) Result
We test the game for 10 times. Each time the game perform normally as we have expected.

Section 6: Conclusion

We think our design is of high level of difficulty. The usage of all kinds of interface, how to generate pipes and how to judge whether the bird passes the pipe or not is the core of this game. We have put a lot of efforts into bettering user experiment of the game. As for the fluency of the game, we have processed quantities of attempts. We were confronted with the following **problems** at this process.

1. The score counter didn't work as we expected.
2. The space between two pipes became smaller and smaller. Finally, they made superposition.
3. The display was not stable enough. There were some unexpected dots and unexpected pause icon appearing in the sight.
4. When the user accelerated the bird, the display would tremble.
5. The switch of day background and night background was out of order.

There are still some room for us to **improve** the game.

1. For example, it would be better if we put the score display on the screen rather than drawing on the segments.
2. Next, we lack background music.
3. We can better the game by adding some coins on the screen. If the bird touches the coin, the score will increase faster.
4. The game lacks a file to record the history. A leaderboard would be better.
5. If the game can support two users to play at the same time, adding some competitive factors, it would be perfect.

All in all, although the game is not perfect, leaving us a lot to do to perfect it, yet it can satisfy the need from users that they need to relax after bustling. After all, based on the fundamental function, we have added many new ideas into the game. As we later add more powerful functions, the game will be more and more welcome.

Section 7: Team member and contribution

Assignment:

Project Design: 陈博文 舒义然 朱曼青

Program Code: 陈博文 朱曼青

Project Report: 舒义然 朱曼青

Video: 陈博文 舒义然 朱曼青

Contribution:

陈博文 3150102397 34%

舒义然 3150105765 33%

朱曼青 3150102186 33%

References

[1] ***FPGA Prototyping by VerilogExamples.*** **By Pong P. Chu**

[2] 张延伟，杨金岩，葛爱学等，“Verilog HDL 程序设计实例详解”

[3] 施青松，“软硬件课程贯通教学实验系统接口说明”