ECE385

Fall 205

Final project report

Final Project report

Xiang Li

Jinying Li

TA: Xinying Wang

AB8 Wed 9:00m am

1 introduction

This the final report for our eve 385 final project.This game is using an Altera DE2 FPGA , a keyboard and a monitor for vga output. Our game is a basic two players fighting game. All the sprites from this game is supported by game “LIttle Fighter” which is an open-source game. The first player can use w,a,d to move and k,l to attack while the second player can use left arrow, up arrow, right arrow to move and 2,3 to attack. Each player have a hp bar, and when he/she is attacked, he/she could lose his/her hp. The time one player’s hp is zero, the game is over. The reason we choose this project is because this is an interesting game idea and we think we are able to implement it.

2 game setup and instruction

This game is a simple 2-players fighting game. After starting the game, you will see two characters. P1 can use ‘w’ to jump, ‘a’ to move left, ‘d’ to move right, ‘j’ to normal attack and ‘k’ to ranged attack. Same function can be done with ‘↑’, ‘→’, ‘←’, ‘2’, ‘3’ by player2. The goal for this game is to defeat another player. When player get hit by another, he/she will lose hp. Either one player’s hp is zero, the game is over. Then you can use reset key to restart the game.

3 description of circuit and module

The whole project is made by several modules. “lab8.sv” is our top level entity, and we have different module to handle data inputting, processing and via drawing.

module : lab8.sv (top level entity)

The lab8.sv is our top level entry which basically has all other module communicate with others. Like it will get signals from sprite_table and send it to color_map to do via drawing. It also connect nios_system to other module to get all 6 keycode inputs.

module : sprite_table

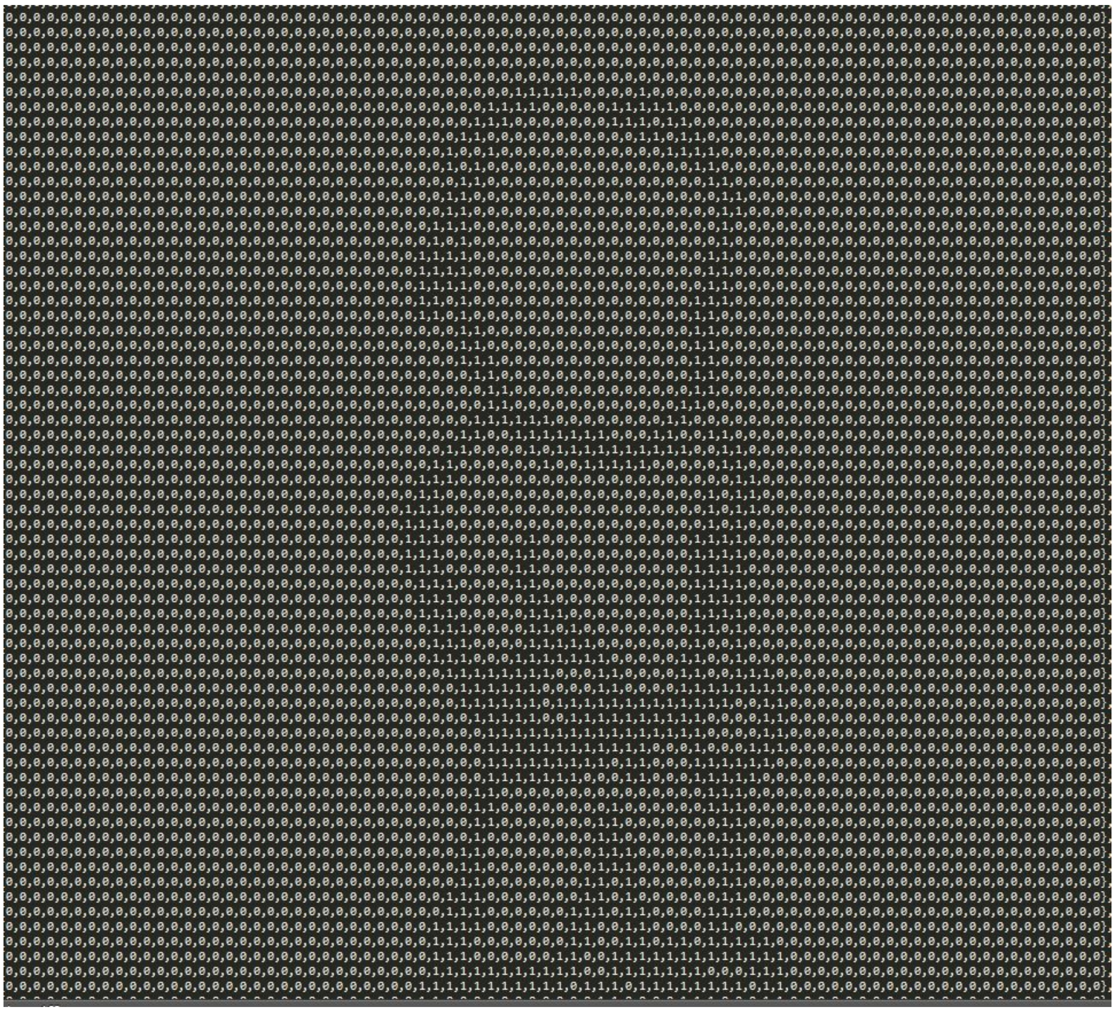
Important signals:

output logic [0:79][0:79][0:3] player_w0,player_w1,player_w2,player_w3, player_j0, player_j1, player_j2,player_a0, player_a1, player_a2,player_a3,player_a4,player_a5,

Those are all drawing sprites we need for animation.

Description :

The way we store sprites is very simple. We write a c++ program to convert normal picture to sprite table with 0, 1, 2, 3... to stands for different color. Like 0 stands for blue, 1 stands for red. Then module sprite_table output those sprites as 2d array to color_map module to do drawing steps. This method use on chip memory and it takes a very long time to pre-loading those sprites-2d arrays.



module : player1/player2

Important signals:

[15:0] keycode0, keycode1, keycode2 : keycode input

[9:0] damage : damage control signal

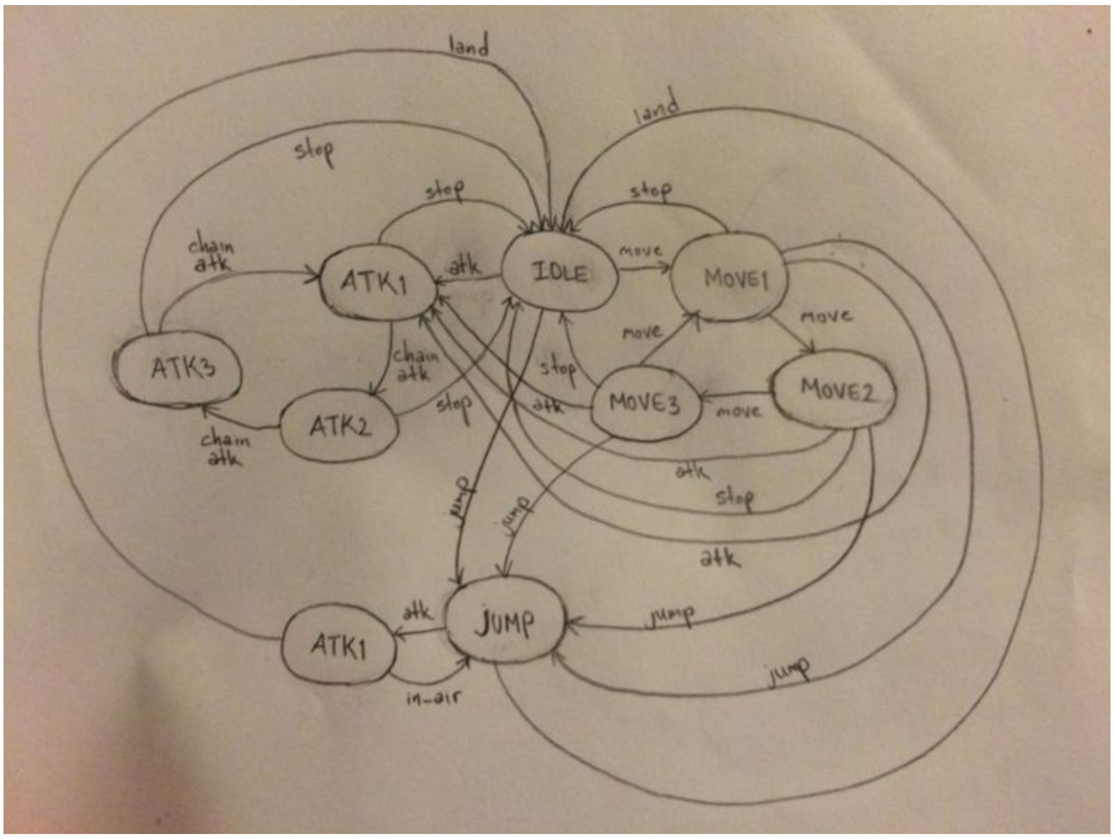
Summon_ball : ball generated signal

[8:0] action : player animation signal

Description :

This module is used to control all position, animation signals towards players. It get inputs as key codes and has logic including moving control, jumping as well as a finite state machine which is used to control all animation. To deal with 6 keycodes input, we use "or" logic to detect whether these 6 keycodes contain one certain key. If it does, it can affect state machine which we used to control player's behavior.

It have a action signal to control all animation part. We have a finite state machine to control walk, jump, attack etc states. Every state has its own action signal, color_map module will draw different sprites according to different action signals. And summon_ball signal is to generate a ball for ranged attack.



In this diagram, different states stands for different animation. Like move1, move2, move3 will give different signals to color_map to draw different sprites.

module : ball1/ball2

Important signals:

Summon_ball : drawing ball signal

Summoned_ball : ball reset signal

[8:0] ball_action : ball animation signal to tell color_map draw different ball sprites

Description :

The ball module handle “range attack”. When player press certain key, ball module will generate a ball whose position is same as player’s current position and moving forward. Player could lost hp by touching the ball. And it also has a counter to process ball animation. Like the counter increase by itself, and color_map will draw different ball according to different counter number. Each player can only have one ball on screen due to summon_ball signal. After the ball disappear at edge, summoned_ball signal can reset summon_ball signal to generate a new ball.

module : hp

Important signals:

Ending : game end signal

[9:0] player_hp1, player_hp2 : player’s hp signal

Description :

This module draw two player’s hp bar. When player gets attacked, it will decrease its hp_w signal to show that player loses hp. When hp become 0, the game is over.

module : hexdriver

Description :

The hex driver module is used to show keycodes input for all 6 keycodes. This is used for easy debugging.

module : nios_system

Description :

The nios2 cpu is used to get usb-keyboard input by OTG chip. We can use nois2 cpu to get most 6 keycode from usb keyboard input signals which are used to control movement of two players. If there are more than 6 keyboard inputs, all keyed signals would become 01.

module : color_map

Important signals:

[9:0] DrawX, DrawY,cloudX1, cloudY1,cloudX2, cloudY2, cloudW, cloudH,hp1_x, hp1_y, hp2_x, hp2_y,player_x1, player_y1,player_x2, player_y2, ball_x1, ball_y1 : Drawing sprites position signals

Description :

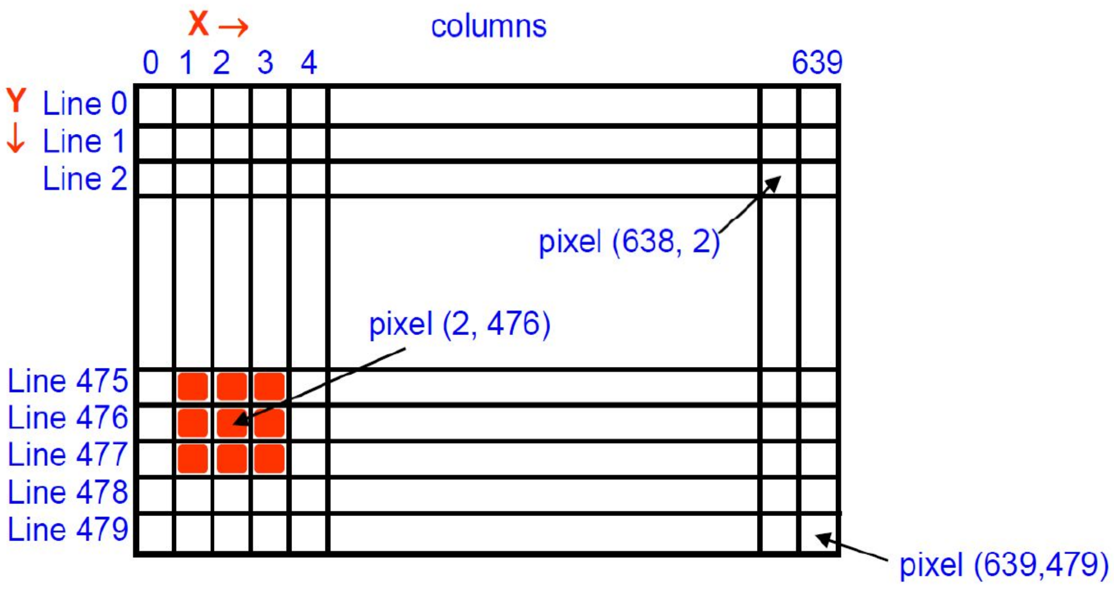
The color_map module is handle all drawing sprites stuff in our project. It will take all positions signals towards different sprites and signals from sprite_table. First we have logic to detect whether to draw one sprite in a right position like:

```
if((DrawX>=player_x)&&(DrawX<=player_x+player_w)&&(DrawY>=player_y)&&(DrawY<=p
layer_y+player_h))
        player_on = player[DrawY-player_y][DrawX-player_x];
else
        player_on = 0;
```

The second function of color_map is to determined all color we need to draw sprites. Like for different numbers in player_on, we set different RGB value for them. That is basically what color_map module does in our project.

module : hpi_io_intf
Description :
This module is used to communicate data between nios cpu and otg chip. It let CPU receive all keycodes from use-keyboard. It is given from lab8. It pass 16 bit data from usb register to CPU,

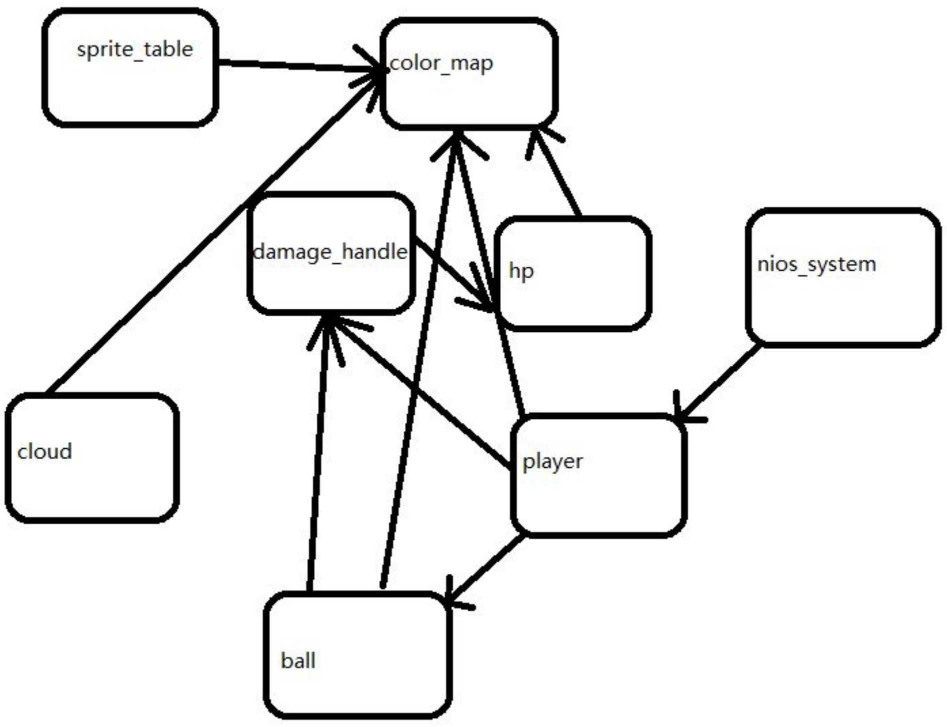
module : vga_controller
Description :
This vga_controller module was used to synchronize the VGA monitor. It will update current pixel and synchronize it with vga_hs, vga_vs signals. It also output Draw_X and Draw_Y signal which is current coordinate of pixel in monitor.



module : damage_handler
Important signals:
[3:0] hit1, hit2 : hit state signals, which is used to have hit delay(player will not hit others according to clock cycle, it will be a delay there)
Description :
The damage_handler do all collision detect work. When player is in attack behavior and he/she hits collision area towards another player. For collision area, we test distance between two players, when the distance is below than a certain limit, then damage_handler will output "player_dmg" signal. The other player will lose hp for that. This is one most important feature about the game. Damage_handler will output "player_dmg" signal to tell player module whether they will lose hp.

<u>4 top level entity diagram</u>

This is top level entity diagram for this final project.



We can't provide schematic diagram for each module because it takes so long(more than 30 min) to get RTL view with sprite_table module, which basically is a giant LUT. And even we get the RTL views, it will be too complex to see clear. That is why we decide to provide with top level entity diagram.

5 Software Description

For this project, we just use c-code for communication between cpu and usb-keyboard. We modify the main.function from lab8 to make it can read 6 keycode instead of two. To do this we add two more IOWR function in main.c.

5 statics

LUT	8765
DSP	No dsp used
Memory(BRAM)	681592
Flip-Flop	2460
Frequency	140.45MHz
Static Power	203.44wW
Dynamic Power	97.46mW
Total Power	353.54mW

6 conclusion

Challenge:
During working for project, we solve lots of problems like
-reading whole 6 keys from usb keyboard
-use vga draw sprites
-creating finite state machine to handle animation
-how to make 2-players game
-how to deal with hit-area(collision)

This is a very interesting project and we all put a lot of time, effort on it. We use all what we learned from this course to accomplish this project. We still have plenty idea like more special attack, player block, audio etc. to add but due to time limitation, we can not. But I believe this is a great project and a very fun game.