Final Project: Snakes

109060001 梁靜如 109060028 蔡佩諭

1. Design Specification:

For snake

Input

clk (100M Hz), rst(active high reset)

in (press the button to show the highest score in history on ssd)

start (switch for start the game)

level1, level2, level3 (three switch for speed level)

mute (decide display the music or not)

Inout

PS2_DATA (input/output signal from the keyboard port)

PS2_CLK (input/output signal from the keyboard port)

Output

vgaRed, vgaGreen, vgaBlue (decide the red, blue, green on the vga)

hsync, vsync (for vga horizontally and vertically)

[3:0] display_c (for ssd control)

[7:0] display (display on seven segmen)

[3:0] led (four led to show the current speed level)

audio_sdin (1 bit serial audio data output)

audio_sck (25M Hz/4, serial clock)

audio_mclk (25M Hz, divided by 4 from 100MHz)

audio_lrck (25M Hz/128, sample rate clock of parallel input audio)

For frequency divider (1Hz, 5Hz, 10Hz, 25MHz, update)

Input

clk (100M Hz), rst(active high reset)

Output

[1:0] ssd_ctl (for scan control)

clk_10 (10 Hz)

clk_vga (25M Hz, for change the pixel on vga)

clk_update

For speed

Input

clk (100M Hz), rst(active high reset), clk_10 (10 Hz)

level1, level2, level3 (three switch for speed level)

Output

[9:0] speed

[3:0] led

For ssd

Input

[3:0] dig (the score)

Output

[7:0] display (display on seven segmen)

For ssd_ctl

Input

[1:0] ssd_ctl (clk control for ssd)

[7:0] display0, display1, display2, display3 (four digit for score)

Output

[7:0] display (display on seven segmen)

[3:0] display_c (for ssd display control)

For vga

Input

pclk (scanning), reset (reset the pixel conting)

Output

hsync (video synchronization in horizontal)

vsync (video synchronization in vertical)

valid (determine the valid scanning area)

[9:0] h_cnt (scan the image in horizontal direction)

[9:0] v_cnt (scan the image in vertical direction)

For blk_mem_gen_0, blk_mem_gen_1

Input

clka (clk)

wea (write enable, 0 for read out the image)

addra (read address from h_cnt and v_cnt)

dina (data input)

Output

douta (data output)

For KeyboardDecoder

Input

clk (100M Hz), rst(active high reset)

Inout

PS2_DATA (input/output signal from the keyboard port)

PS2_CLK (input/output signal from the keyboard port)

Output

[511:0] key_down (control the buttons of the keyboard)

[8:0] last_change (composed of extend code and make code)

key_valid (used to indicated the press and the release action to the buttons)

For direction

Input

clk (100M Hz), rst(active high reset)

start (switch for start the game)

[511:0] key_down (control the buttons of the keyboard)

[8:0] last_change (composed of extend code and make code)

key_valid (used to indicated the press and the release action to the buttons)

Output

[3:0] dir (4 bits to show up/down/left/right)

For random_point

Input

clk_vga, rst (active high reset)

Output

[9:0] rand_x (indicate the position of apple in x-axis)

[9:0] rand_y (indicate the position of apple in y-axis)

For note_gen

Input

clk (100M Hz), rst(active high reset)

start (switch for start the game)

mute (decide display the music or not)

music (indicate whether we should play the music or not)

[21:0] pitch (the value which is responsible to control the frequency of output voice)

Output

[15:0] audio_left (the output voice signal for left channel)

[15:0] audio_right (the output voice signal for right channel)

For speaker_control

Input

clk (100M Hz), rst(active high reset)

[15:0] audio_in_left (the output voice signal for left channel)

[15:0] audio_in_right (the output voice signal for right channel)

Output

audio_sdin (1 bit serial audio data output)

audio_sck (25M Hz/4, serial clock)

audio_mclk (25M Hz, divided by 4 from 100MHz)

audio_lrck (25M Hz/128, sample rate clock of parallel input audio)

For sound

Input

clk_5 (5 Hz), rst (active reset)

start (switch for start the game)

gameover (indicate the game is over or not)

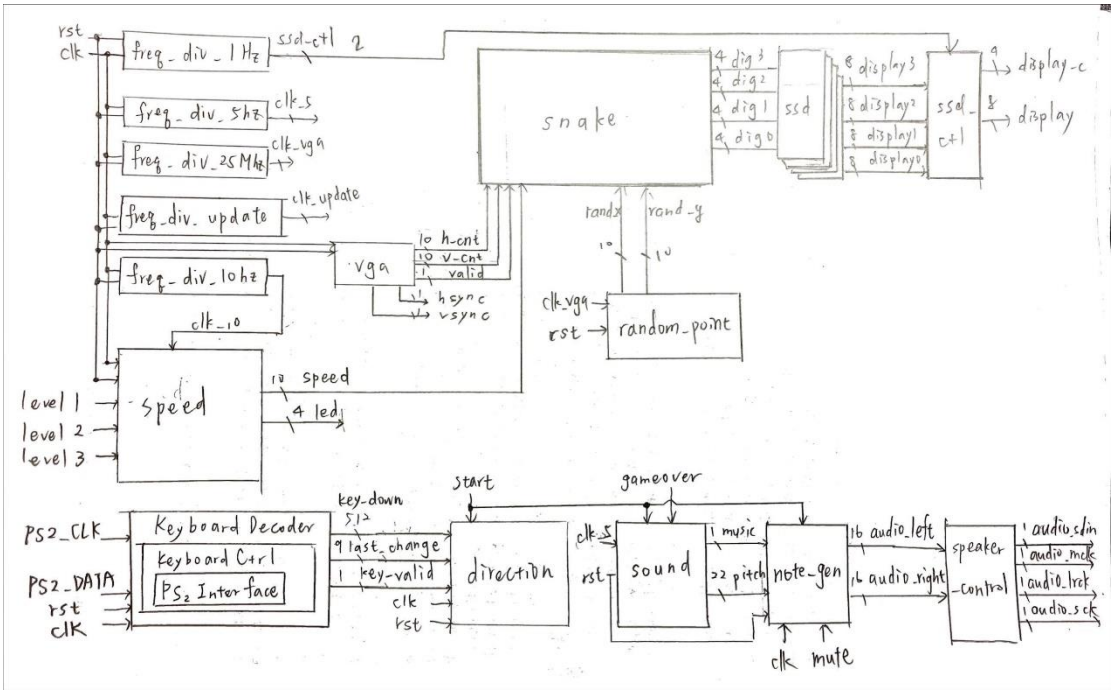
Output

Music (indicate whether we should play the music or not)

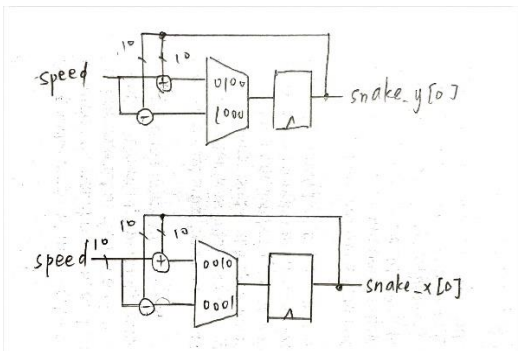
[21:0] pitch (the value which is responsible to control the frequency of output voice)

2. Block diagram:

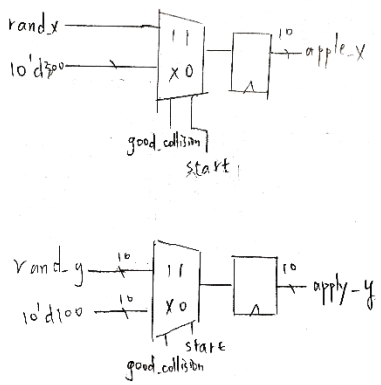
For snake.v



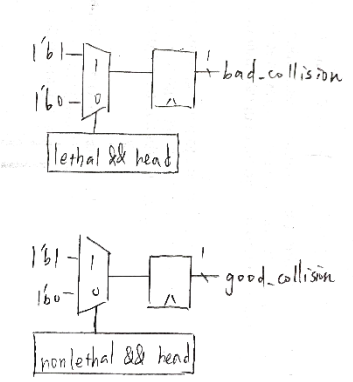
For snake moving



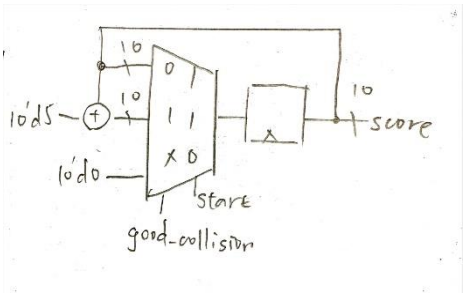
For apple's position



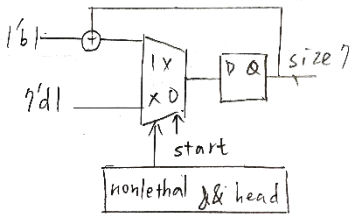
For judge good collision or bad collision



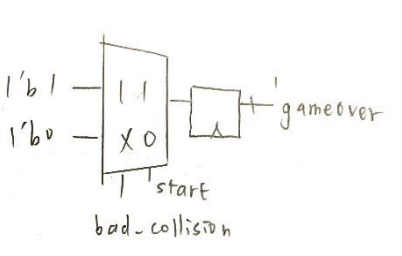
For scoring



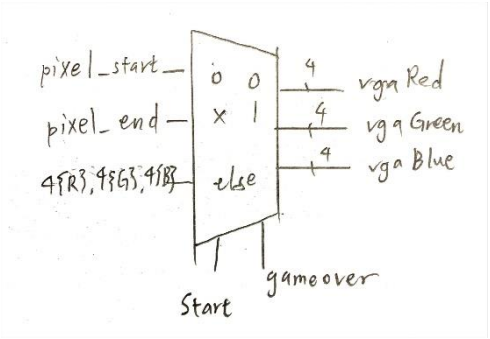
For snake's size



For definition of gameover



For vga setting

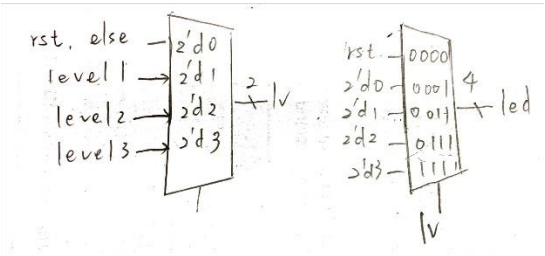


(i). start

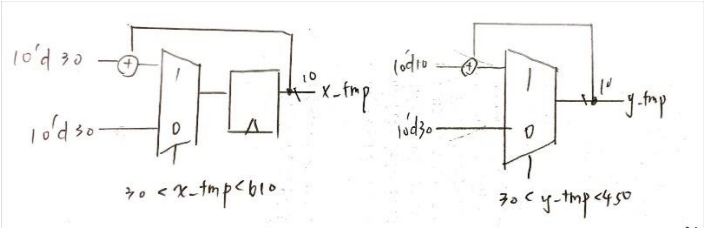
(ii). end

The pixel of start is 320*180, so the write depth in setting ip is 57600.
The pixel of end is 320*240, so the write depth in setting ip is 76800.

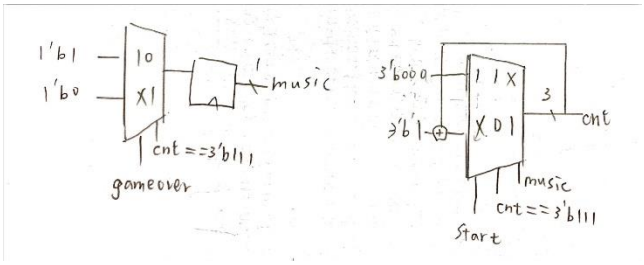
For speed.v

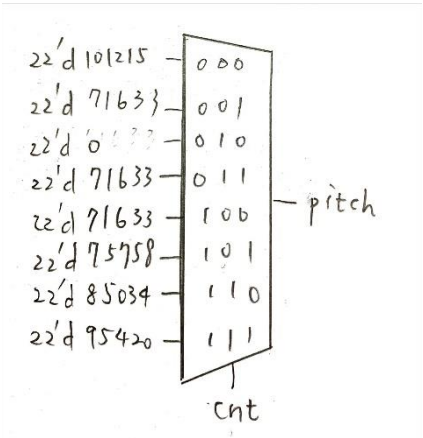


For random point



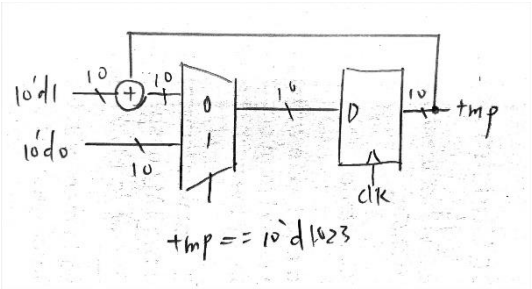
For sound.v

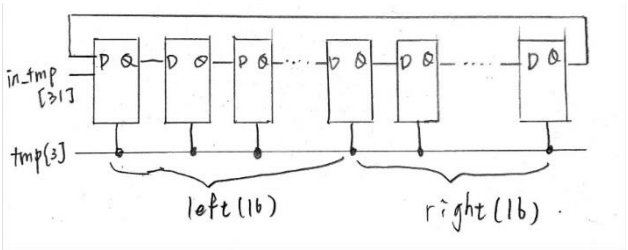




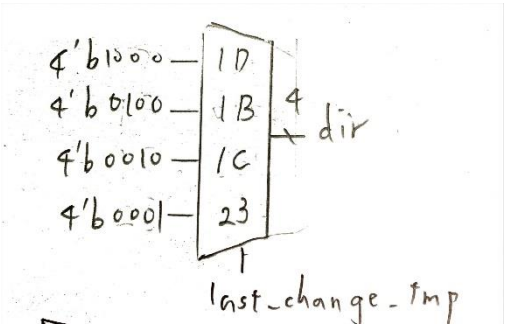
The sound plays here refered to “Mario”.
(game in Nintendo)
The sound would play non-stop until the play switch(T1) turns down again.

For speaker_control.v





For direction.v



This design comes from the usual game, which plays 1 usually use “ASDW” to control the character to move.

3. Implementation:

(i). This is a classical game called “Snake”. The player use “A/S/D/W” on the keyboard to control the snake to move left/down/right/up and to eat the apple. Once you eat an apple, you can get 5 points and make the snake grow longer. To make the growth more easily to discover, we let the snake grow 4 units per time, which means you get 20 points an apple. All the score will be shown on the seven segments display. Your goal is not to bump into yourself or the border of the screen. Our game can show you your history highest score if you press the button T17.

(ii). There are 4 speed level you can choose, which is determined by the DIP switch W17/W16/V16. Level0 is default value. After deciding the speed of snake, you can enjoy your game by turn up DIP switch T1. Before you turn on the start game, the screen would show you a start picture which has been shown in page 5.

(iii). When the game starts, the screen will show you the game. Red indicates apple, Green indicates snake, and Blue is the border. When you die, the speaker would play a circular soundtrack. Then the screen will show you a gameover picture which has been shown in page 5.

(iv). Corrections:

- In our proposal, the start is controlled by a button. But after we start writing our project, we found it easily to start a game by switch. So we change the start from “button” to “switch”.
- There is no back home button.
- We add a new button “mute”. If you don’t like the soundtrack, you can turn up this DIP switch V17. Your world would be quiet immediately.

I/O specification:

I/O	vgaBlue[3]	vgaBlue[2]	vgaBlue[1]	vgaBlue[0]	clk	rst
LOC	J18	K18	L18	N18	W5	R2
I/O	vgaGreen[3]	vgaGreen[2]	vgaGreen[1]	vgaGreen[0]	PS2_CLK	PS2_DATA
LOC	D17	G17	H17	J17	C17	B17
I/O	vgaRed[3]	vgaRed[2]	vgaRed[1]	vgaRed[0]	hsync	vsync
LOC	N19	J19	H19	G19	P19	R19
I/O	display[7]	display[6]	display[5]	display[4]	display[3]	display[2]
LOC	W7	W6	U8	V8	U5	V5
I/O	display[1]	display[0]	display_c[3]	display_c[2]	display_c[1]	display_c[0]
LOC	U7	V7	W4	V4	U4	U2
I/O	lv	led[3]	led[2]	led[1]	led[0]	start
LOC	U18	V19	U19	E19	U16	T1
I/O	mute	audio_lrck	audio_mclk	audio_sck	audio_sdin	in
LOC	V17	A16	A14	B15	B16	T17
I/O	level1	level2	level3			
LOC	V16	W16	W17			

4. Discussion:

Here are some problems when we are writing this project:

(i). Problems: We are not familiar with the procedure of vga.

- Watch the class video again, and try to do the lab10.1 though it does not need to be submitted. We also discuss with other classmate, which helps us a lot.
- After successfully put on the first picture, we want to have another picture. But the vivado show us an error for we don't have enough RAM. I found the problem is that the picture I want to put in is bigger than the write width(76500). I think that the number didn't need to change at first, but I'm wrong. I use 小畫家 to modify the pixels and also modify the write width in ip setting. Then it works successfully.
- Another problem about vga is that I didn't create different lace to store different data. So the computer can't decide which data to read.

- Wrong

```
106 blk_mem_gen_0 blk_mem_gen_0_inst
107 (
108     .clka(clk_vga),
109     .wea(0),
110     .addra(pixel_addr),
111     .dina(data),
112     .douta(pixel_start)
113 );
114 blk_mem_gen_1 blk_mem_gen_1_inst
115 (
116     .clka(clk_vga),
117     .wea(0),
118     .addra(pixel_addr),
119     .dina(data),
120     .douta(pixel_end)
121 );
```

Correct

```
106 blk_mem_gen_0 blk_mem_gen_0_inst
107 (
108     .clka(clk_vga),
109     .wea(0),
110     .addra(pixel_addr),
111     .dina(data_start[11:0]),
112     .douta(pixel_start)
113 );
114 blk_mem_gen_1 blk_mem_gen_1_inst
115 (
116     .clka(clk_vga),
117     .wea(0),
118     .addra(pixel_addr),
119     .dina(data_end[11:0]),
120     .douta(pixel_end)
121 );
```

(ii). Problems: When debugging, the vga show “out of range”

- We had ask this question on eeclass. The answer from the teacher is that we might be given the wrong frenquncy. So we separate the freq_div module into small specific frequency divider module. Then it works.

(iii). Problems: We found that the soundtrack would only play once initially or even no sound.

- We check all the module about speaker and found that there is a parameter doesn’t return to zero. After modifying, it can play soundtrack circularly. We also add a switch in case being too noisy. When you turn on the DIP switch v17, it will be mute.
- Wrong

```
49 always @(posedge clk_5 or posedge rst)
50 if(rst || (~start))
51     cnt <=3'b000;
52 else if (music && (cnt < 3'b111))
53     cnt <= cnt +3'b1;
```

Correct

```
49 always @(posedge clk_5 or posedge rst)
50 if(rst || (~start))
51     cnt <=3'b000;
52 else if (start && (cnt == 3'b111))
53         cnt <= 3'b0;
54 else if (music && (cnt < 3'b111))
55     cnt <= cnt +3'b1;
```

- In our previous project, we could make our left and right ear sound different tunes. We also try to make the sound in this project can be more delicate. When we tried to change our sound different from left and right ear, it turns out that left and right ear truly hear different sound, but it comes with the noise that we can’t get rid of it. It’s too bad that we can’t upgrade our soundtrack.

(iv). Problem: Optimize the user's feel of use.

- Firstly, we use 10 Hz for playing the soundtrack. After replay the game, we found it a little irritable with the circular, quick note. So we change the frequency into 5 Hz. It sounds better!

5. Work contribution

梁靜如: 整合和設計 snake.v(含 ssd, ssd_ctl, freq_div, random_point, border 定義等)， debug，製作報告影片

蔡佩諭: 設計有關聲音(sound, speaker_control, note_gen), 插入遊戲起始和結束畫面，設計 speed.v，製作書面報告