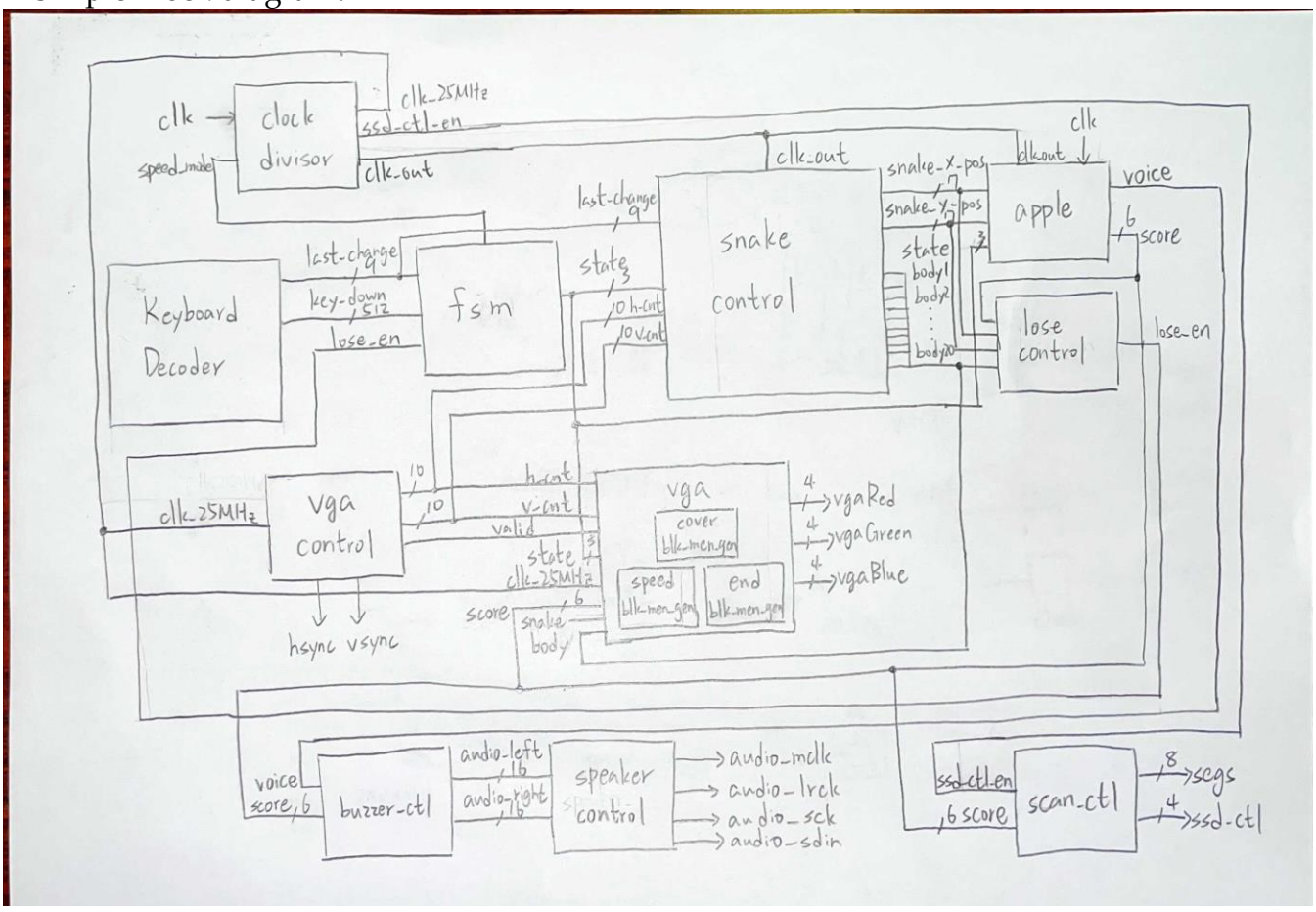
Final Project Report

Design Specification

IO:
Input: clk, rst.
Inout: PS2_DATA, PS2_CLK.
Output: [7:0] segs, [3:0] ssd_ctl,
audio_mclk, audio_lrck, audio_sck, audio_sdin
[3:0] vgaRed, [3:0] vgaGreen, [3:0] vgaBlue, hsync, vsync.

Design Implementation

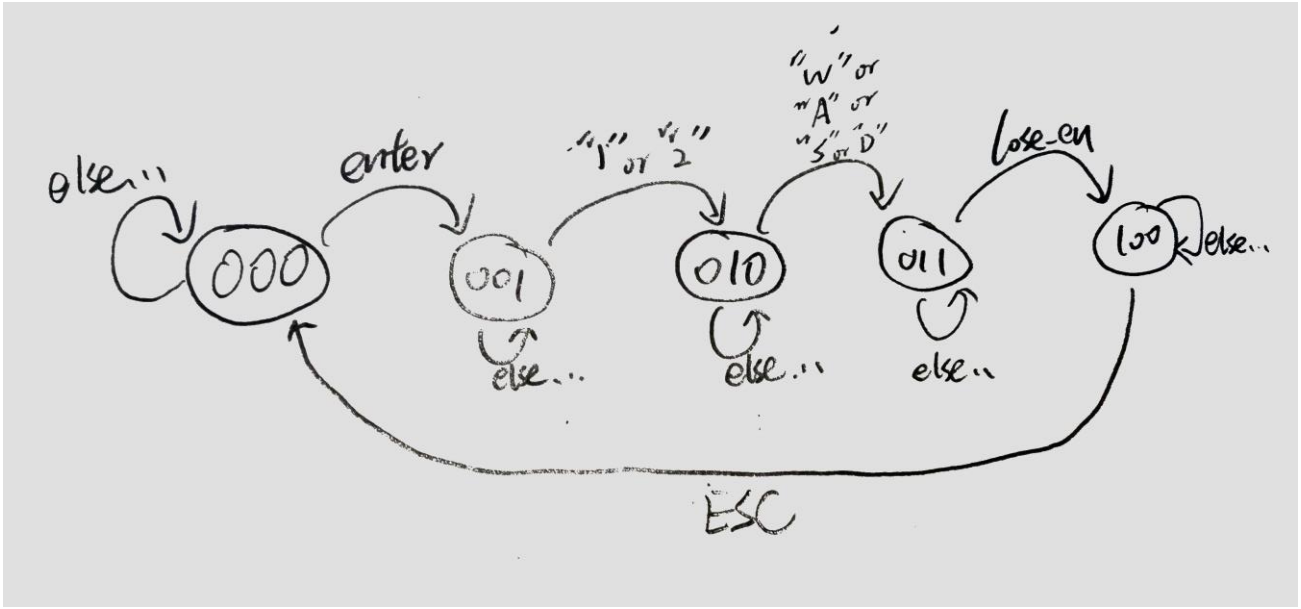
為了可讀性，我把整個 project 分成了十二個部分：KeyboardDecoder, fsm, clock divisor, snake control, apple, lose control, vga control, vga, buzzer control, speaker control, scan_control 和 top，Block diagram 如下圖所示。
A simple Block diagram:



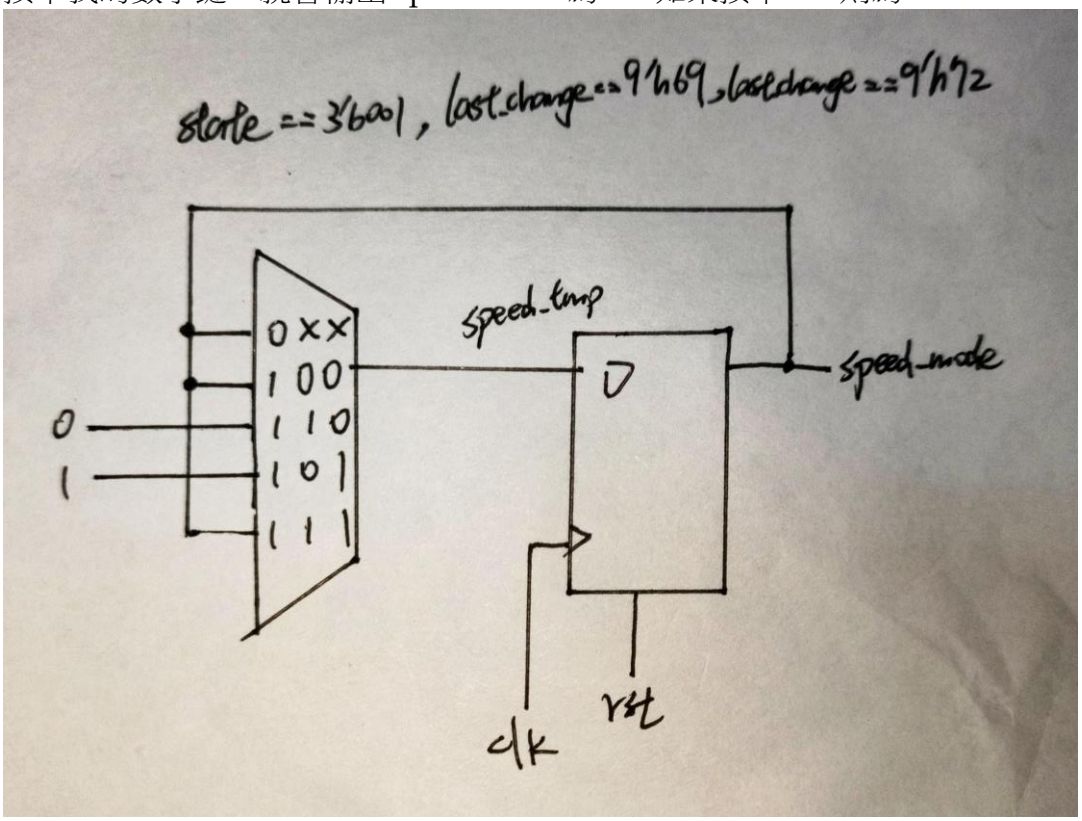
KeyboardDecoder：
將鍵盤訊號轉換成可以成我們 verilog 可以作用的訊號 keydown, last_change 和 key_valid。我們可以從 keydown 訊號裡面得知此時有哪　些按鍵是被按著的。從 last_change 得知最近一個被操作的按鍵組合為 何，而 key_valid 則是在按下或放開時第一時間通知的訊號。

fsm (finite state machine)：

這裡我有五個 state，000 是起始的畫面，按下 enter 後會進入下一個畫面，001 是選擇速度的畫面，選擇的方法是按鍵盤右側的數字鍵的 1 或 2，之後就會進到 010 就是控制前的靜止畫面，這是我為了讓玩家多一段時間準備而做的，之後當玩家按下 WASD 任意一個按鍵時，就會跳轉到 011。011 就是玩家的控制頁面，只有到輸了(lose_en)才會跳到 100，也就是結束頁面。最後在 100 時，只要按下 ESC 就會回到我們的起始頁面。



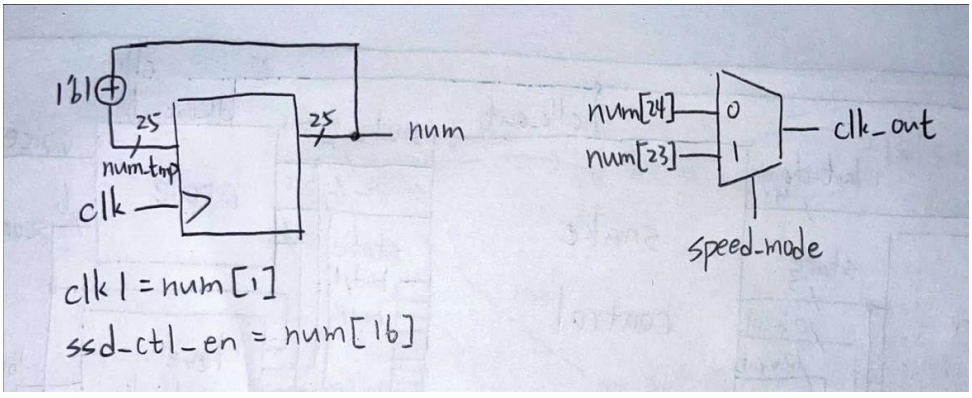
在這個 module 裡面，我順便設定了 speedmode 的控制，做法也很簡單，就是一個 MUX 和一個 DFF。我用 state 是否為 001，數字鍵是否為 1 或 2 來控制 MUX。如果在 001 的 state 裡面，按下我的數字鍵 1 就會輸出 speedmode 為 0，如果按下 2，則為 1。



snake control：
這個 module 是我們的蛇身控制。原本的畫面是 640*480 個像素，我將他分成 40*30 格，其中每一個格子為 16*16 個像素。因為我不想讓蛇在全螢幕上面跑，所以我特別規劃了一個新的活動區域。我上下各取 5 格，左右取 10 格。因此我蛇活動範圍橫向只有新定義的 10~30，直的只有 5~25。我先設定方向，之後再來設定蛇身的位置。我設定了一個二位 direction_state，當我的蛇向上面移動時為 00，向下面為 01，向左為 10，右為 11。這裡比較巧思的設計是，我讓它一開始的 state 為 11，因為當遊戲進入到控制頁面(011)時，蛇的位置在最左上角，這樣遊戲開始的時候，就不用擔心我的蛇會到處跑。

有了方向，之後就是移動我的蛇頭。因為我限定了蛇的移動範圍，所以他的起始點為(10,5)。之後我按照前面定義 direction_state 去對我的 x 或 y 座標進行加或減。這樣一來，就能很簡單的控制我的蛇頭了。另外，控制蛇頭的情形只會在 011 的 state 發生，一旦跳到其他的 state，就會把蛇頭的位置設為初始位置，同樣的蛇身也是當跳到別的 state 就會回歸到初始位置。
最後就是蛇身的控制，我設定蛇身的上限為 20 格。然後並不是當蛇吃一個蘋果時，會生一節身體，而是這些身體打從一開始都是存在的，他們只是隱形的，且不會受到碰撞。當蘋果吃到對應的數量，對應蛇身才會塗上顏色，碰撞效果也才實現。因為我讓他們隱形，所以他們的起始位置會在(10,6~25)。之後我就會讓第一節以 clk_out 的速度去覆蓋蛇頭的位置，第二節覆蓋第一節先前的位置，以此類推。

clock divisor：
簡單來說就是除頻器。這裡我需要三個 clk。分別為掃描七段顯示器的 ssd_ctl_en，行進速度需要的 clk_out，VGA 螢幕掃描及圖片產生需要的 clk1(clk_25MHz)。
這裡的做法很簡單，就是讓 25 位的 num 按 clk 的速度跑 DFF，之後再取他的第二位做 clk1(clk_25MHz)，就能得到 1/4 的原先 clk 頻率，$\frac{100MHz}{4}$ = 25MHz。因為七段顯示器的掃描不需要太準確，只要能造成視覺暫留即可，所以我取 num 的第 17 位作為 ssd_ctl_en。最後行進速度的 clk，因為我的速度有兩種，所以我這裡輸入之前做的 speedmode，若為困難模式則取 num 的第 24 位，反之則為第 25 位。這樣的設計就能讓我的蛇在不同模式以不同的速度前行。

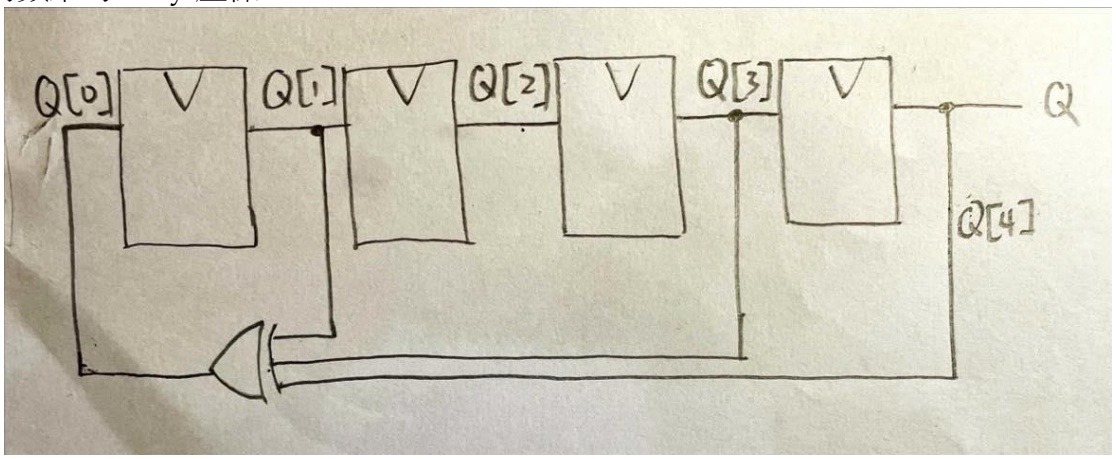


apple：
這個 module 是去產生要改變蘋果位置的訊號、計算蘋果隨機的位置、累積分數和產生蛇吃到蘋果要發出聲音的訊號。

1. 產生要改變蘋果位置的訊號：
當蛇頭的位置與蘋果的位置相同時，產生一個 clock(100MHz)的訊號(change_en)

2. 蘋果隨機的位置：

因為蘋果只能出現在我們新設定好的區域內，也就是中心 20*20 的範圍內，所以我把蘋果的 x、y 軸座標分開產生，利用教授提供產生隨機數字的方法，也就是一個 5-bit 的 shifter，輸入是從中間取出幾個 bit 做 XOR，但因為五個 bit 的 shifter 會產生 32 個隨機數字，但我只需要 20 個隨機數字，所以我判斷如果 Q >= 5'd20，就把 Q 的值減 20，這樣我就能得到 0~19 總共 20 個隨機數字，當改變蘋果位置的訊號出現，就產生下一個隨機數字。另外，因為我們遊戲的範圍是在螢幕中心 20*20 的空間，所以要把生成出來 x 軸的隨機數字加 10，y 軸的隨機數字加 5，這兩個數字就分別為蘋果的 x、y 座標。

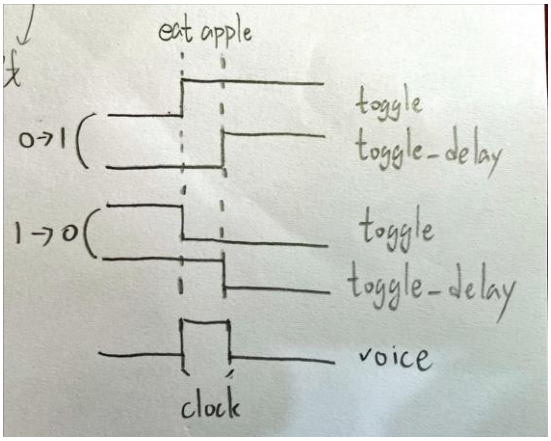


3. 累積分數：

當改變蘋果位置的訊號(change_en)出現時，代表蛇吃到了蘋果，所以把分數(score)加 1，如果到遊戲的起始畫面(state == 3'b000)就把分數重置。

4. 蛇吃到蘋果要發出聲音的訊號：

因為蛇吃到蘋果是一瞬間的狀況，如果要發出聲音要一段比較長區間的訊號，所以我另外設一個訊號(toggle)，當蛇吃到蘋果時，就 toggle 一次(0→1;1→0)，再把這個訊號 delay 一個 clock(蛇移動的 clock)，最後把這個訊號與 delay 一個 clock 的訊號做 XOR，這樣就得到蛇吃到蘋果要發出聲音且夠長的訊號。



vga control：

這個 module 是參考教授掃描螢幕的方式，產生 h_cnt, v_cnt, valid, hsync, vsync，其中 h_cnt 和 v_cnt 是代表現在螢幕掃描到的位置，h_cnt 表示橫向的座標，v_cnt 表示縱向的座標，而 valid 則是表示現在掃描到的位置是否為輸出所會利用到的位置。

lose control：

這個 module 的目的是檢測是否有碰撞，若有，遊戲會跳到結束畫面（100）。這裡跟 VGA_ctl 和 snake_ctl 的一樣，會根據我的分數(score)，來確認我的蛇身此時是否應該存在，進而判斷碰撞與否。

這裡的碰撞的判斷，我是用一系列的蛇身的位置判斷是否與蛇頭重疊，如果位置一致且碰到的蛇身存在，那就會判定遊戲結束(lose_en 為 1)，跳轉到結束畫面。

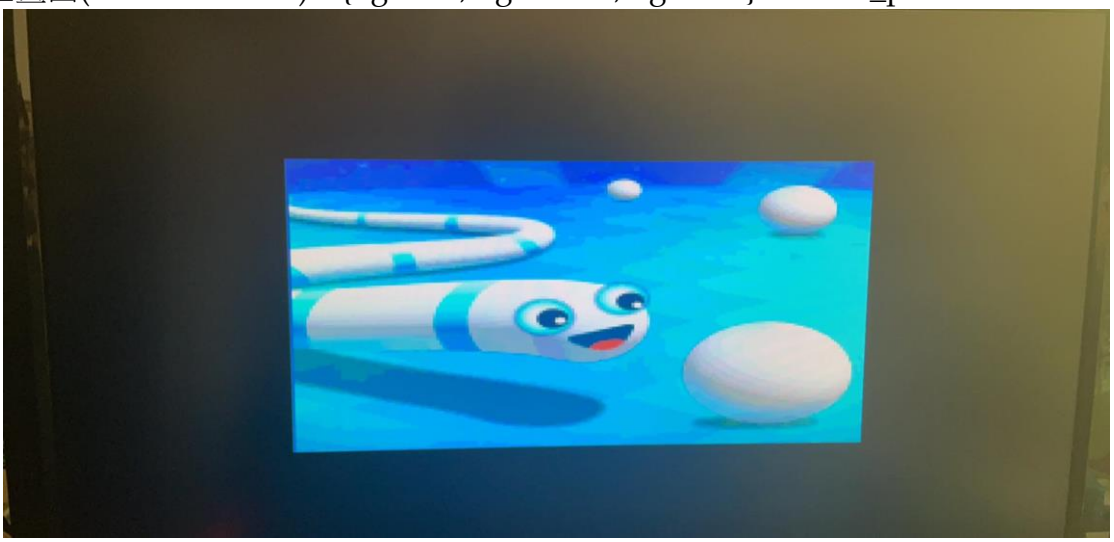
上面那段是跟蛇身的碰撞，跟牆壁的碰撞同樣會判定遊戲結束，跳轉畫面。這裡的條件很簡單，就是當我蛇頭的 x、y 座標到達我的邊界以外時（x=9 或 30，y=4 或 25），lose_en 就為 1，也就是遊戲結束。

vga：

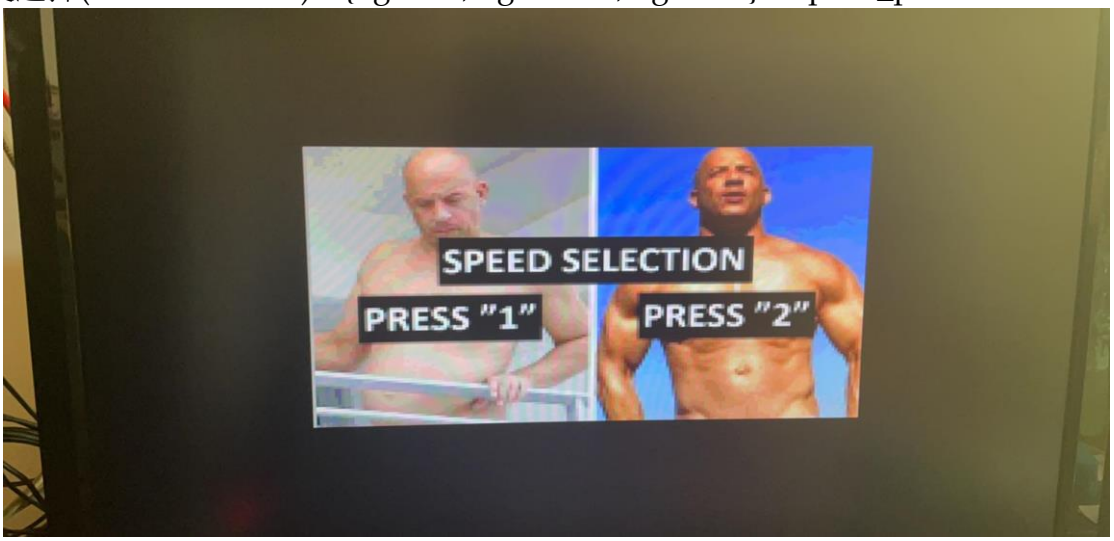
我利用這個部份去控制所有螢幕相關的輸出。

首先，因為考慮到我要放三張圖片，記憶體可能會不足，所以我每張圖片都只用 180*120 的 RAM，並且把一格的 RAM 貼到四個 pixel，把 180*120 的圖片放大成 360*240 的圖片，但圖片的解析度不變。接著判斷現在的 state 決定要輸出什麼顏色在螢幕上：

- 起始畫面(state == 3'b000)：{vgaRed, vgaGreen, vgaBlue} = cover_pixel

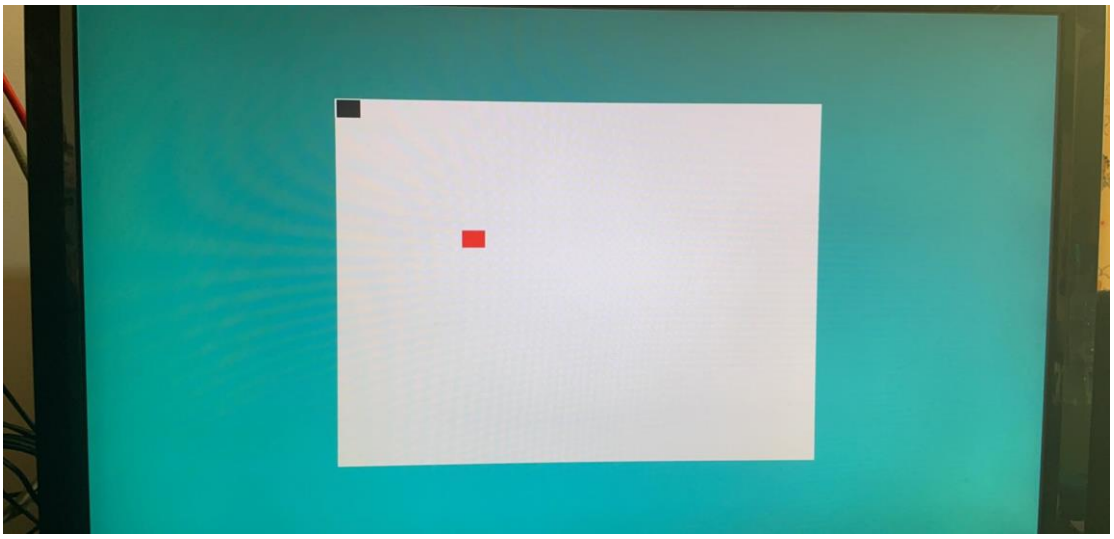


- 速度選擇(state == 3’b001)：{vgaRed, vgaGreen, vgaBlue} = speed_pixel

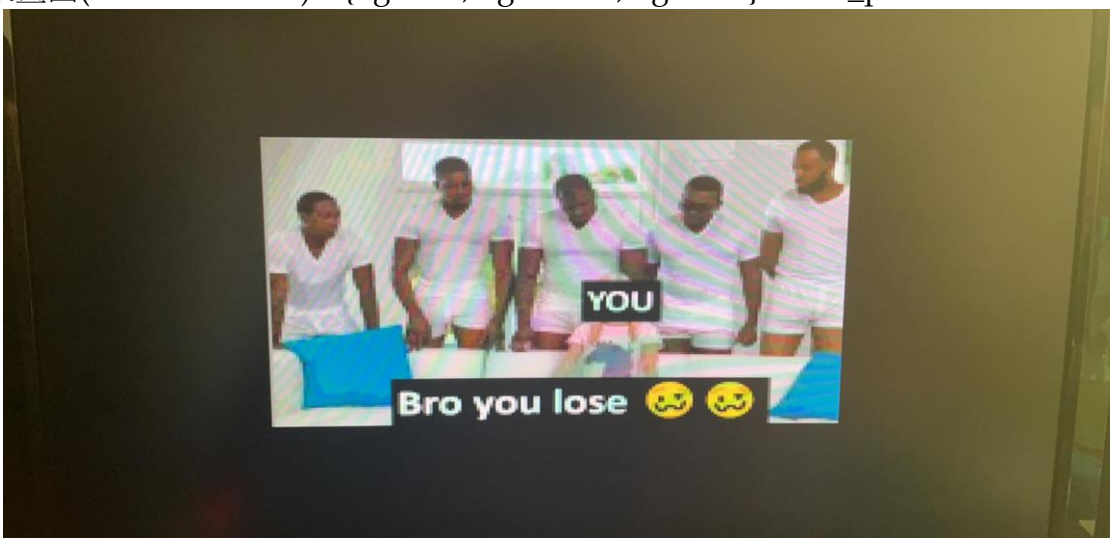


- 遊戲等待畫面與遊戲進行中(state == 3'b010 || state == 3'011)：

先把邊界外部塗上相同的顏色 {vgaRed, vgaGreen, vgaBlue} = 12'h099；再利用 snake_control 所輸出的蛇頭(黑色)與蛇身的位置，並且加入 score 進行判斷，例如:如果 score > 6'd0，代表蛇身 1 應該要顯示在螢幕上，所以就把指定的顏色賦予到蛇身 1 所代表的 pixel 上，蘋果也是利用 apple 這個 module 所輸出蘋果的位置再把那格的顏色指定為紅色({vgaRed, vgaGreen, vgaBlue} = 12'hf00)；最後把蛇頭、蛇身、蘋果的之外的位置都指定為白色({vgaRed, vgaGreen, vgaBlue} = 12'hfff)



- 結束畫面(state == 3'b100)：{vgaRed, vgaGreen, vgaBlue} = end_pixel



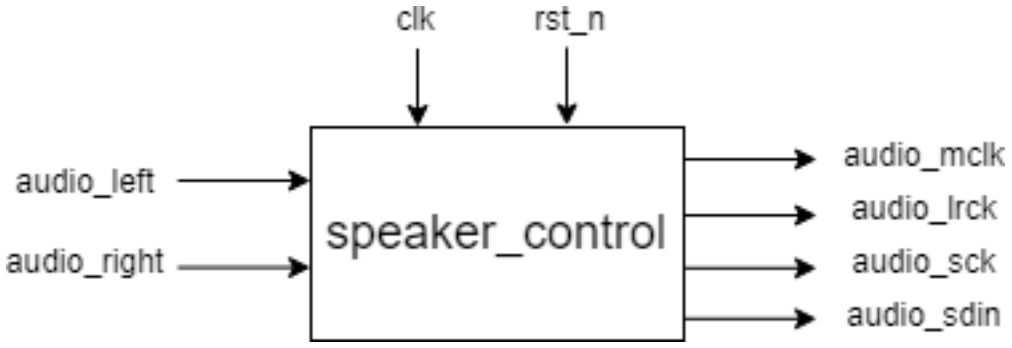
buzzer control：
為了增加一些趣味性，蛇在吃了第一顆蘋果揚聲器會發出中音 Do，第二顆會發出中音 Re，第三顆會發出中音 Mi，接著就按照這個順序循環撥放，所以我把分數(score)除三的餘數作為現在要發出什麼聲音的依據，如果餘數等於 1，就把中音 Do 的臨界值賦給 freq_gen。

計算臨界值的方法：
中音 Do、Re、Mi　3 種音頻的數位訊號(參考下面附表)。例如：中音 Do 是 262Hz 的頻率，所以臨界值是 191571 ($\frac{100MHz}{2\times191571} \approx 262Hz$)；中音 Re 則是 294Hz 的頻率，所以臨界值是 170648 ($\frac{100MHz}{2\times170648} \approx 294Hz$)。

Sound	(Hz)						
B	247	494	988	1976	3951	7902	15804
A#	233	466	932	1865	3729	7458	14917
A	220	440	880	1760	3520	7040	14080
G#	208	415	831	1661	3322	6644	13288
G	196	392	784	1568	3136	6272	12544
F#	185	370	740	1480	2960	5920	11840
F	175	349	698	1397	2794	5588	11176
E	165	330	659	1319	2637	5274	10548
D#	156	311	622	1245	2489	4978	9956
D	147	294	587	1175	2349	4699	9398
C#	139	277	554	1109	2217	4435	8870
C	131	262	523	1047	2093	4186	8372

接著，我宣告一個變數 toggle，當 counter 數到臨界值時，把 counter 重置且 toggle 的值轉變一次(toggle = ~toggle)。最後，使用一個 MUX，以 toggle 作為條件去設定震幅的大小，當 toggle = 1 時，我把左聲道和右聲道震幅的高值設為 16'h5000；當 toggle = 0 時，我把左聲道和右聲道震幅的低值設為 16'h3111，藉此創造出各聲音所需要的頻率和震幅。

speaker control：
我把 speaker control 分成了兩個部分：frequency divider、conversion of data from parallel to serials。Frequency divider 去產生 Pmod I2S2 所需要的 clock。



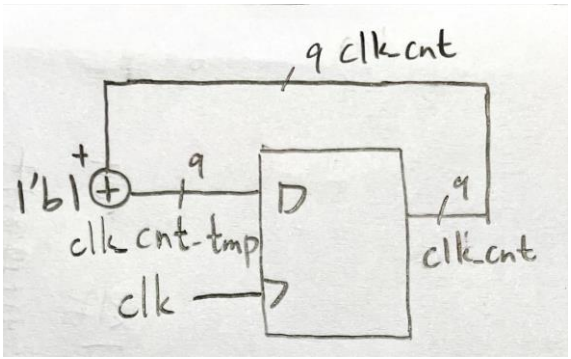
freqency divider:

According to the picture below, we can produce 25MHz for MCLK, $\frac{25MHz}{128}$=192kHz for LRCK. Furthermore, because we need to convert 32bits to 1bit (parallel to serial), we also produce 32 times LRCK ($\frac{25MHz}{128} \times 32 = \frac{25MHz}{4}$ =6.25Hz).

LRCK (kHz)	MCLK (MHz)									
	64x	96x	128x	192x	256x	384x	512x	768x	1024x	1152x
32	-	-	-	-	8.1920	12.2880	-	-	32.7680	36.8640
44.1	-	-	-	-	11.2896	16.9344	22.5792	33.8680	45.1580	-
48	-	-	-	-	12.2880	18.4320	24.5760	36.8640	49.1520	-
64	-	-	8.1920	12.2880	-	-	32.7680	49.1520	-	-
88.2	-	-	11.2896	16.9344	22.5792	33.8680	-	-	-	-
96	-	-	12.2880	18.4320	24.5760	36.8640	-	-	-	-
128	8.1920	12.2880	-	-	32.7680	49.1520	-	-	-	-
176.4	11.2896	16.9344	22.5792	33.8680	-	-	-	-	-	-
192	12.2880	18.4320	24.5760	36.8640	-	-	-	-	-	-
Mode	QSM				DSM		SSM			

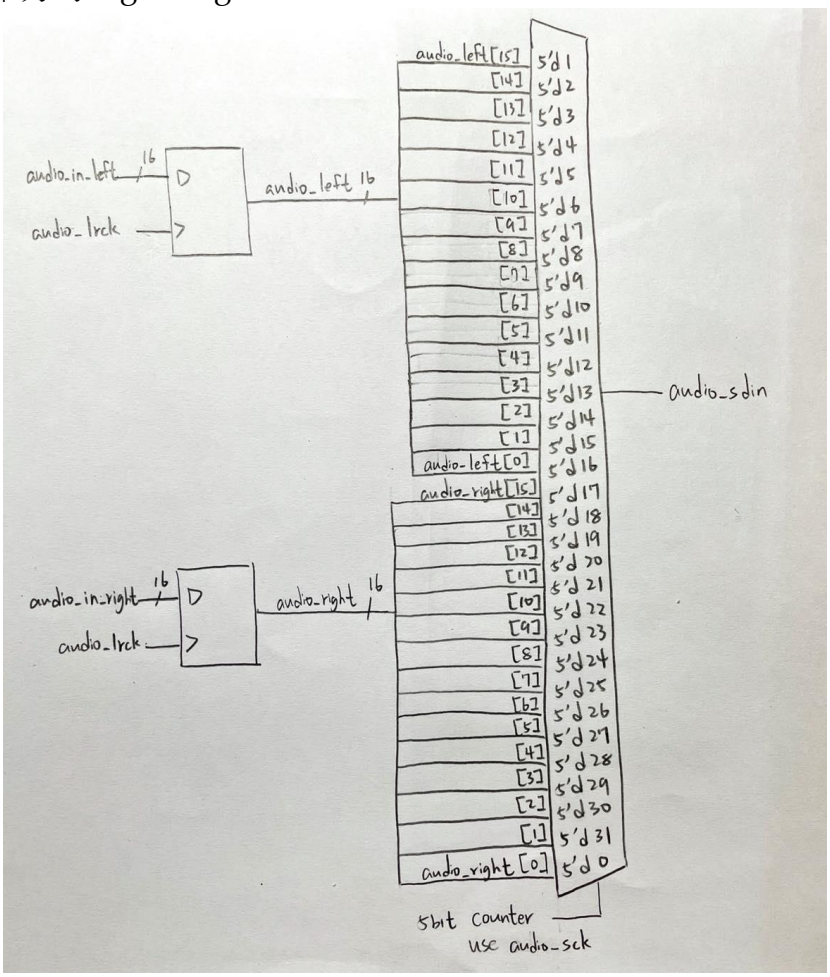
因為我要產生的 clock 都是$\frac{100MHz}{2^n}$，所以我建構一個 9-bit 的 counter，並使用 counter 中不同的 bit 作為我所需要的 clock。

logic diagram for counter:



audio_mclk = clk_cnt[1]
audio_lrck = clk_cnt[8]
audio_sck = clk_cnt[3]

Conversion of data from parallel to serial:
為了把 32bit 轉為 1bit，我把 5-bit up counter 的值作為 MUX 的條件。而且 5-bit up counter 使用 audio_sck 作為它的 clock，因為 audio_sck 比 audio_lrck 快 32 倍，所以在訊號在傳輸時，時間才不會出錯。根據教授的講義，我們從 audio_left 的 MSB 輸出到 audio_right 的 LSB，所以我建構 code 藉由下方的 logic diagram：



scan_control：
我設定這個 module 的目的是為了在七段顯示器，顯示我當前吃了多少個蘋果，雖然蛇身最長只會到 21 格，但是分數(score)還是會增長。所以我在 scan_ctl 轉換 score，並顯示在七段顯示器的一二位。這裡轉換的方式很簡單，就是單純的除以十和除以十求餘數而已。接著，使用 clock divisor 的輸出 ssd_ctl_en 作為 MUX 的條件去切換陽極與陰極，在陽極和陰極快速切換的情況下造成了視覺暫留，使我們看到不同的數字顯示在七段顯示器上。

Top：使用此 module 去連接各 modules, inputs, outputs 和 signals，當遊戲結束(state == 3'b100)，LED 會全亮。

****
綜合上述模組功能，我們完成了一個可以自由調整難度，且可隨心所欲控制的貪食蛇，並且會在七段顯示器紀錄得分數，在吃到蘋果時會發出聲音，撞到牆或身體時遊戲會結束，結束時 LED 燈會全亮，結合各種不同功能的貪食蛇小遊戲的期末專題。
****

I/O pin assignment:

seg[7]	seg[6]	seg[5]	seg[4]	seg[3]	seg[2]	seg[1]	seg[0]
W7	W6	U8	V8	U5	V5	U7	V7

ssd_ctl[3]	ssd_ctl [2]	ssd_ctl [1]	ssd_ctl [0]	clk	rst	PS2_CLK	PS2_DATA
W4	V4	U4	U2	W5	R2	C17	B17

audio_mclk	audio_lrck	audio_sck	audio_sdin	LED[15]	LED[14]	LED[13]	LED[12]
A14	A16	B15	B16	L1	P1	N3	P3

LED[11]	LED[10]	LED[9]	LED[8]	LED[7]	LED[6]	LED[5]	LED[4]	LED[3]
U3	W3	V3	V13	V14	U14	U15	W18	V19

LED[2]	LED[1]	LED[0]	vgaRed[0]	vgaRed[1]	vgaRed[2]	vgaRed[3]	vgaGreen[0]
U19	E19	U16	G19	H19	J19	N19	J17

vgaGreen[1]	vgaGreen[2]	vgaGreen[3]	vgaBlue[0]	vgaBlue[1]	vgaBlue[2]	vgaBlue[3]
H17	G17	D17	N18	L18	K18	J18

hsync	vsync
P19	R19

遇到的問題&解決方案：

因為要符合 proposal 我們所提出的功能(在吃到蘋果時揚聲器要發出聲音)，我們首先遇到的問題就是因為蛇頭接觸到蘋果是一瞬間的狀況，不能作為揚聲器發出聲音的訊號，所以我們絞盡腦汁想出一個辦法去解決這個問題，最後的解決方案就是利用 toggle 和 toggle_delay 去做 XOR，這樣只要控制 clock 就可以創造出任何長度的訊號了。

另外，製作期間，我們其中一人的電腦無預警壞掉（無限重啓），檔案面臨消失的危險。在萬念俱灰的時候，一次重啓的間隔，我們即時插入外接硬碟，把我們的 project 搶救回來。我們也學到了要有定時保存文件和檢視電腦狀況的習慣。

Conclusion：

我們兩個這次都是第一次自己親自做出一款遊戲，雖然過程中可能會不斷地有挫折，但是每一次成功了某項功能，進度往前多走了一步，就會覺得很有成就感，這次的期末專題把我們之前學過的、甚至是沒學過的螢幕顯示都結合在一起，因此透過之前的 lab 來熟悉邏輯觀念、verilog 的語法功能真的非常重要，透過這次跟隊友的合作經驗，我們都學習到了許多以前不會注意的細節，並完成了一個自己設計出來的遊戲，雖然過程很累，但我們的付出一定會得到更多的收穫。

◎分工：

施珮燁：四個 module、proposal、report、圖片後製

簡健翔：六個 module、proposal、report