

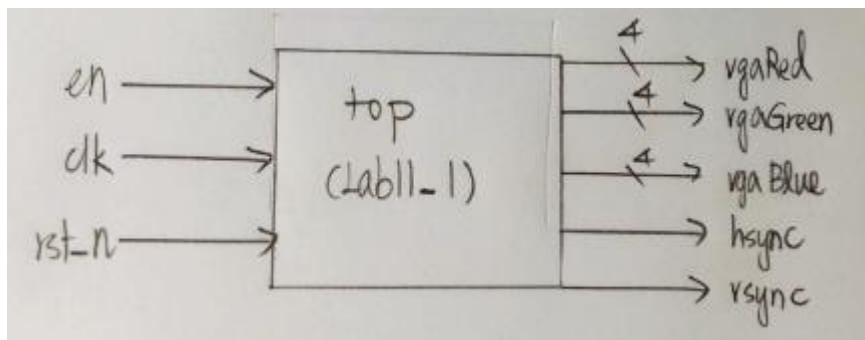
1.

(1) Design specification :

A. Inputs and outputs(表一) :

Inputs	rst_n, clk, en
Outputs	vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync
↑ 表一 : Inputs and outputs of 1	

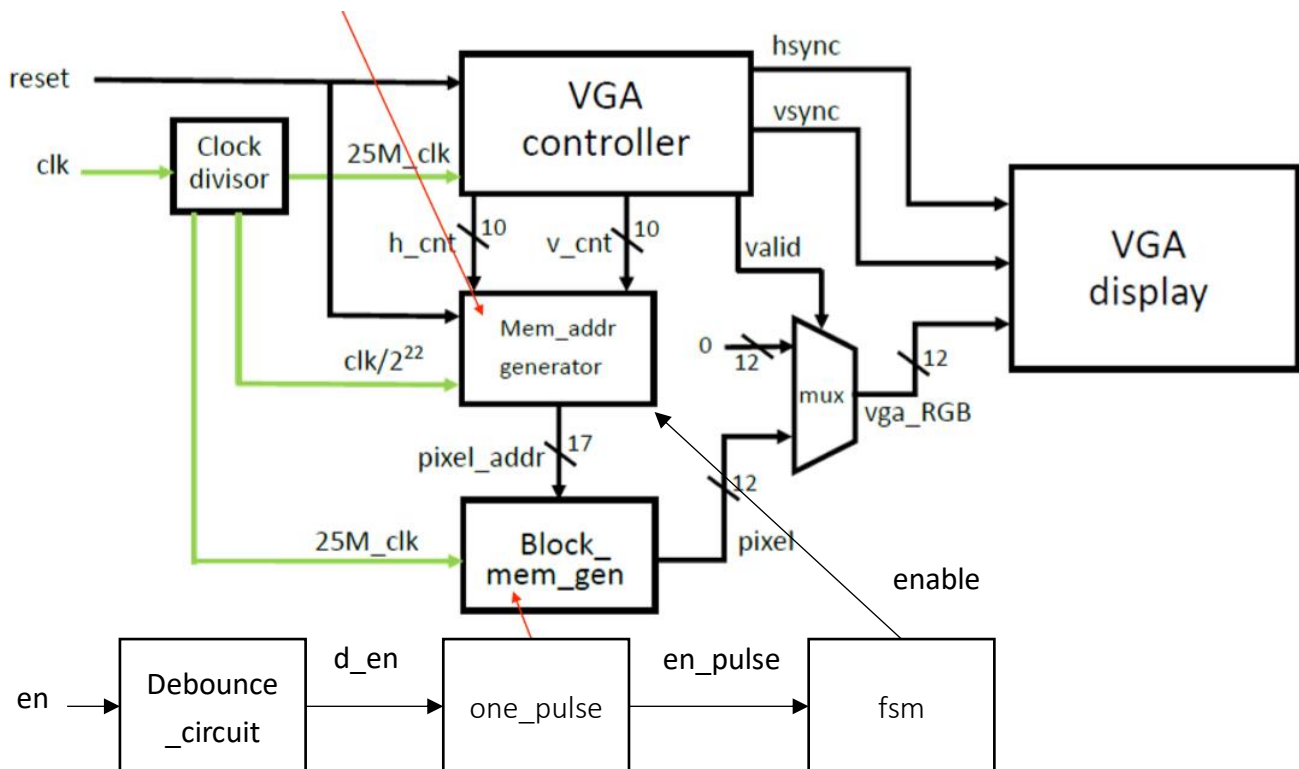
B. Block diagram(function table)(圖一) :



↑ 圖一 : The block diagram of 1

(2) Design implementation :

A. Logic diagram(function table)(圖二) :



↑ 圖二 : logic diagram of 1

B. I/O pin assignment(表二)：

I/O	vgaRed[0]	vgaRed[1]	vgaRed[2]	vgaRed[3]	vgaBlue[0]	vgaBlue[1]	vgaBlue[2]
LOC	G19	H19	J19	N19	N18	L18	R18
I/O	vgaBlue[3]	vgaGreen[0]	vgaGreen[1]	vgaGreen[2]	vgaGreen[3]	hsync	vsync
LOC	J18	J17	H17	G17	D17	P19	R19
I/O	en	clk	rst_n				
LOC	T17	W5	U17				

↑ 表二：I/O pin assignment of 1

C.功能與做法說明：

本題為幫 Demo2 加上開始滾動與停止滾動的功能。

首先 import Demo2，然後用程式將.jpg 圖片轉為.coe 檔，並利用 verilog 內建的 Block Memory Generator 將圖片的.coe 檔轉成 Block_mem_gen 模組。

接著再加入一個 input 按鈕，按奇數次圖片會滾動，按偶數次圖片會暫停滾動。所以將按鈕輸入接到 debounce 和 one_pulse，最後接到 fsm，fsm 共有兩個 state，分別代表暫停和開始，這兩個 state 的 state output(enable)會接到 Mem_addr_generator 來控制圖片的滾動。

2

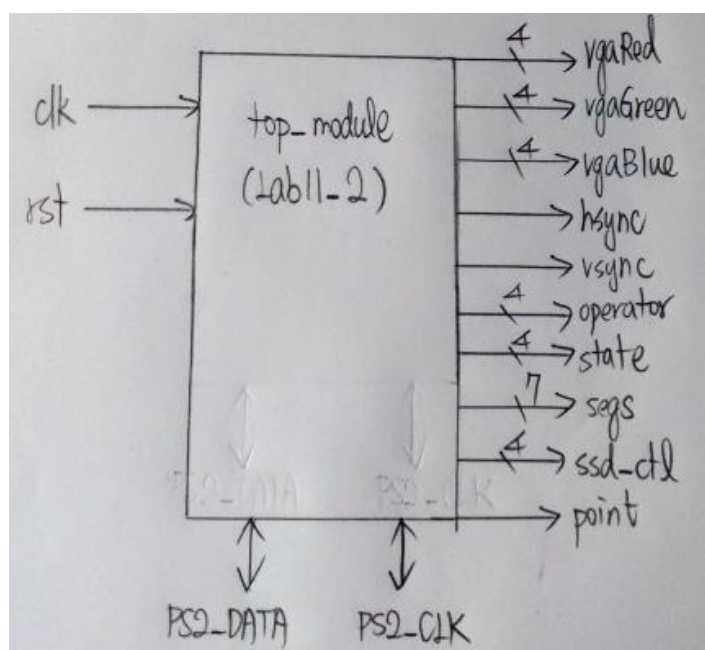
(1) Design specification：

A. Inputs and outputs(表三)：

Inputs	rst, clk
Outputs	[6:0]segs, [3:0]ssd_ctl, point, capital, audio_mclk, audio_lrck, audio_sck, audio_sdin
Inouts	PS2_DATA, PS2_CLK

↑ 表三：Inputs and outputs of 2

B. Block diagram(function table)(圖三)：

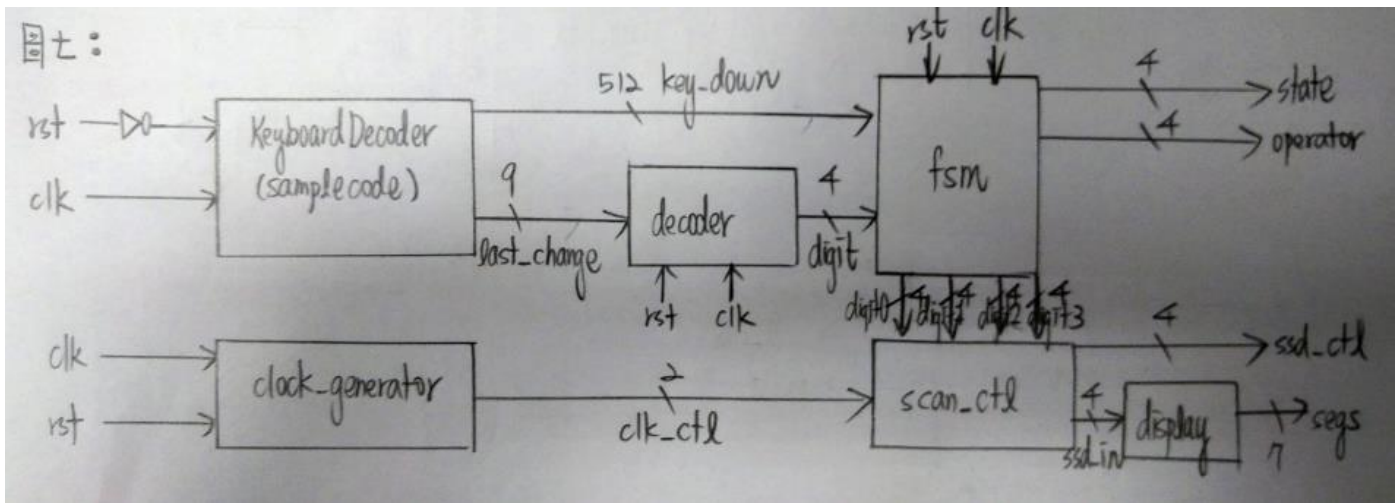


圖三：The block diagram of 2

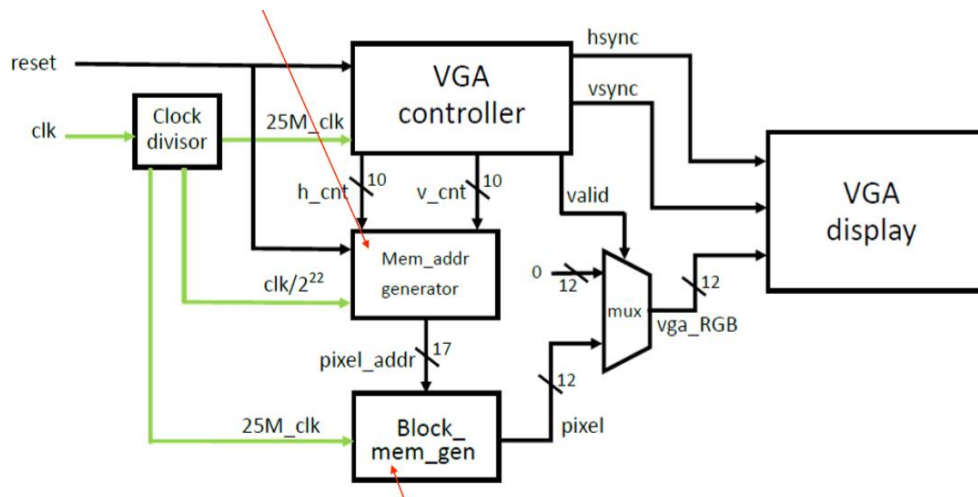
(2) Design implementation :

A. Logic diagram(function table)(圖四、五)：

本題分為兩部分，第一部分是 Lab9_3 題的計算機(圖四)，另一部分為 Demo2：



↑ 圖四：logic diagram of 2(計算機部分)



↑ 圖五：logic diagram of 2(螢幕顯示部分)

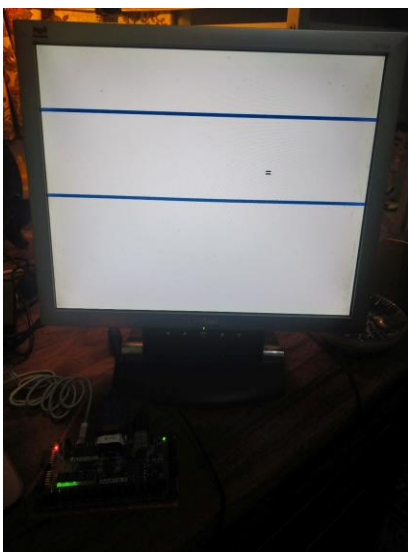
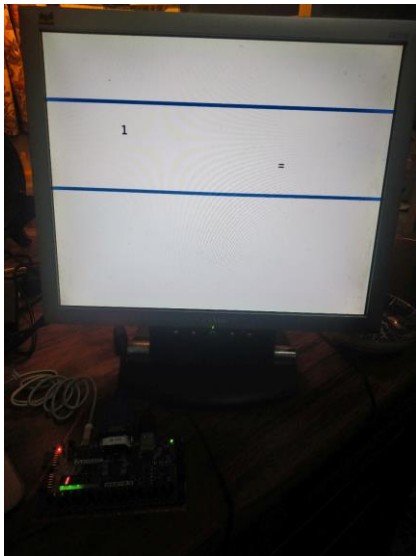
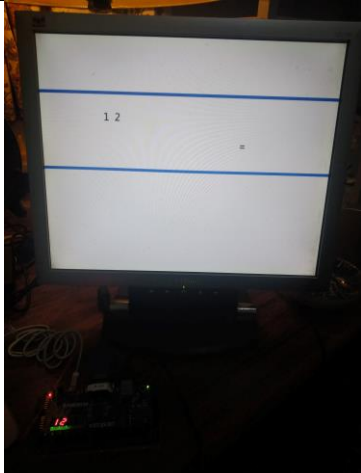
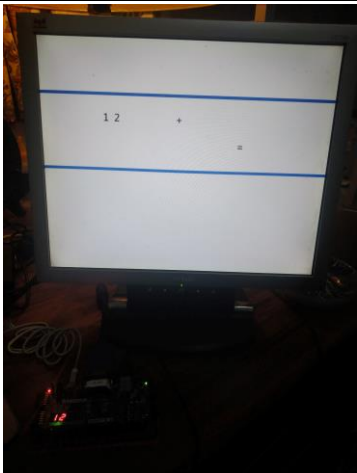
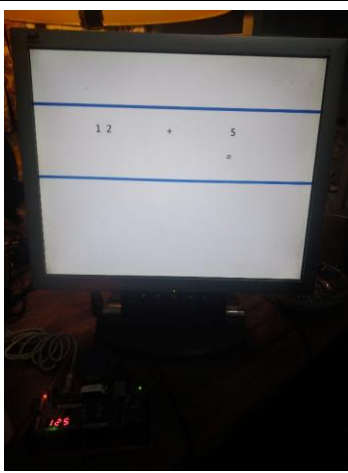
補充說明：對於 Lab9_3 在 fsm 的部分有做一點修改，為了要能夠在螢幕上顯示例如：22 + 22 = 44，所以 fsm 要輸出 8 個 digit(digit0~digit7[3:0])(圖四只有四個，應該為八個) 分別為第一個數字的十位、個位，第二個數字的十位、個位，和最後結果的千位、百位、十位和個位，和一個運算子(+或-或*)。這些 fsm output 接到圖五中的 Mem_addr_generator。

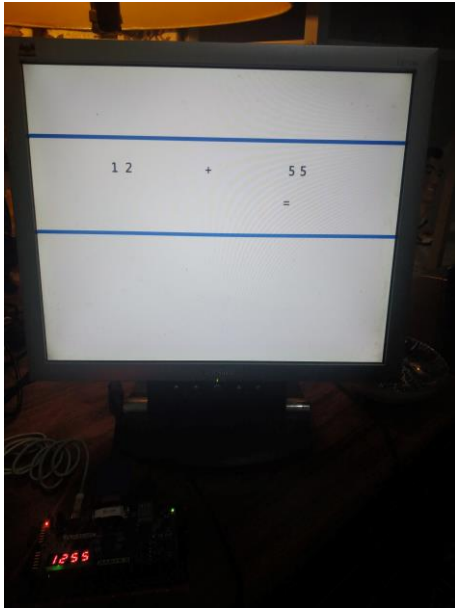
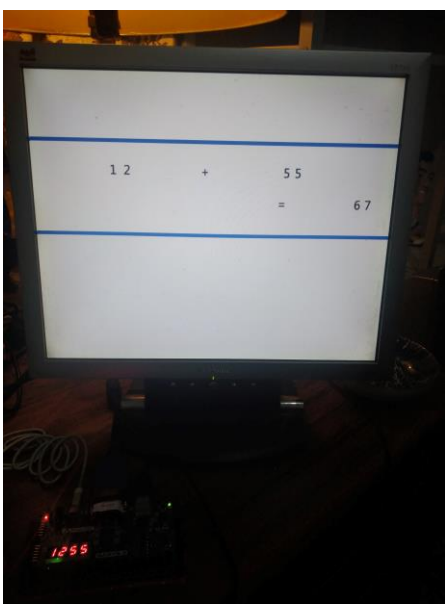
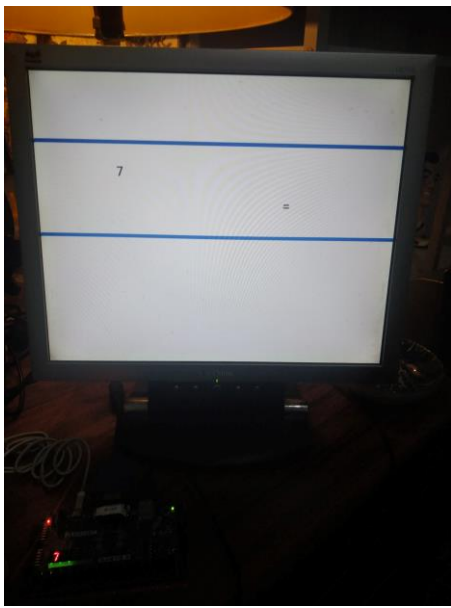
B. I/O pin assignment(表四)：

I/O	vgaRed[0]	vgaRed[1]	vgaRed[2]	vgaRed[3]	vgaBlue[0]	vgaBlue[1]	vgaBlue[2]
LOC	G19	H19	J19	N19	N18	L18	R18
I/O	vgaBlue[3]	vgaGreen[0]	vgaGreen[1]	vgaGreen[2]	vgaGreen[3]	hsync	vsync
LOC	J18	J17	H17	G17	D17	P19	R19
I/O	clk	rst	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	segs[6]
LOC	W5	V17	W4	V4	U4	U2	W7
I/O	segs[5]	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]	point
LOC	W6	U8	V8	U5	V5	U7	V7
I/O	PS2_CLK	PS2_DATA	operator[3]	operator[2]	operator[1]	operator[0]	state[3]
LOC	C17	B17	L1	P1	N3	P3	V19
I/O	state[2]	state[1]	state[0]				
LOC	U19	E19	U16				

↑ 表四：I/O pin assignment of 2

C.功能說明：

(A)剛 program 完板子的狀態。	(B)搬開 V17 開關後進入初始畫面。	(C)輸入第一個數字的十位數。
		
(D)按下 enter 後，即可輸入第一個數字的個位數。	(E)按下想要做的運算(a -> + ; s -> - ; m -> *)	(F)輸入第二個數字的十位數。
		

(G)按下 enter 後，即可輸入第二個數字的個位數。	(H)按下 enter 後，即會顯示結果。	(I)按下數字，即可進行下一次的運算(接著從(C)繼續相同過程)。
		

D.功能說明：

本題結合 Lab9_3 和 Demo2 來完成可以顯示在螢幕上的二位數加減乘計算機。

關於螢幕的部分，最重要的地方是 Mem_addr_generator。首先把圖片做好，產生.coe 檔，並且用 verilog 產生 Block_mem_gen，接著便要用 Mem_addr_generator 來抓所要的圖片的記憶體位置，並且決定它要顯示在螢幕的哪個部分。

在抓記憶體位置的部分我花了很多時間，因為我是直接產生有 0,1,2,3,4,5,6,7,8,9,+,-,*,=的圖片，抓他們在圖片中的位置，雖然我畫圖的時候已經有對齊了，但是實際看.coe 檔並不如預期的整齊，所以要找到兩個數字的中間點比想像中的困難，用了我不少時間，也因此第三題我改用其他的方法增快效率。

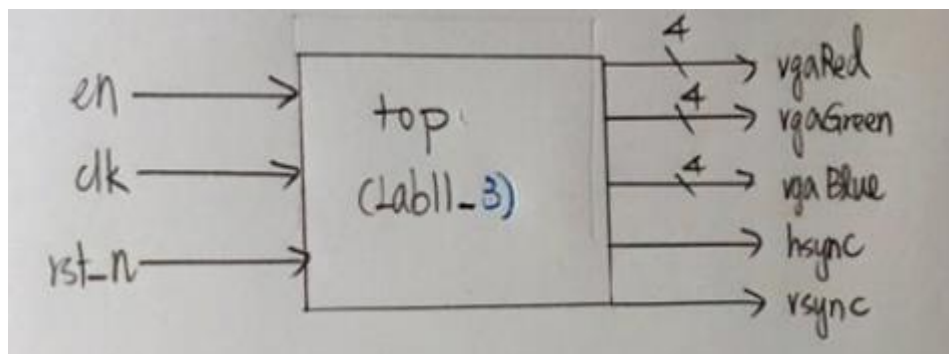
3.1

(1) Design specification :

A. Inputs and outputs(表五)：

Inputs	rst_n, clk, en
Outputs	vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync
↑ 表一：Inputs and outputs of 3.1	

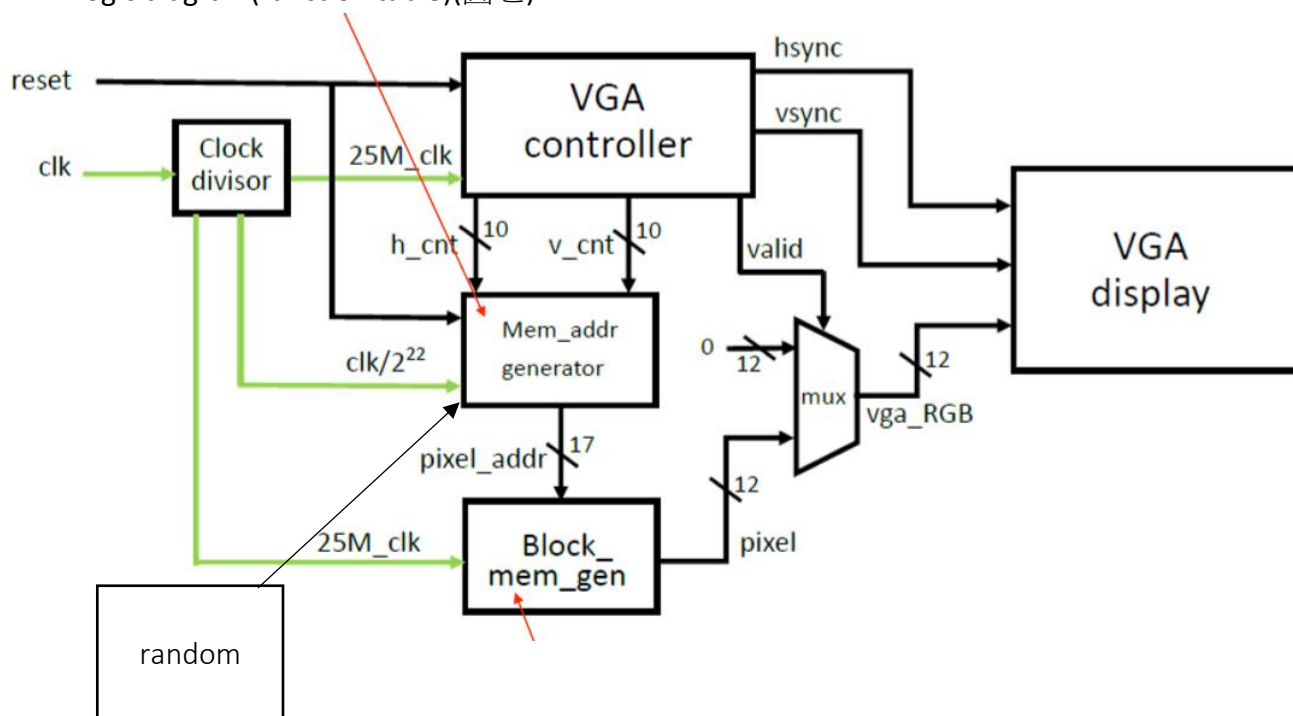
B. Block diagram(function table)(圖六)：



↑ 圖六：The block diagram of 3.1 of 3

(2) Design implementation：

A. Logic diagram(function table)(圖七)：



↑ 圖七：logic diagram

B. I/O pin assignment(表六)：

I/O	vgaRed[0]	vgaRed[1]	vgaRed[2]	vgaRed[3]	vgaBlue[0]	vgaBlue[1]	vgaBlue[2]
LOC	G19	H19	J19	N19	N18	L18	R18
I/O	vgaBlue[3]	vgaGreen[0]	vgaGreen[1]	vgaGreen[2]	vgaGreen[3]	hsync	vsync
LOC	J18	J17	H17	G17	D17	P19	R19
I/O	en	clk	rst_n				
LOC	T17	W5	U17				

↑ 表六：I/O pin assignment of 3.1

C.功能與做法說明：

本題為隨機產生七種俄羅斯方塊在螢幕的中上方。

首先是 **random module** 產生 0~7 的 **random number** (產生頻率為 1 秒)，方法採用老師 po 在 ilms 上的講義，簡單來說就是利用 **shift register** 然後去取幾個中間過程的元素做 **exclusive or**。

再來是最重要的部分：這次畫圖我用的方法是產生七個方塊各自的.coe 檔，再把他們全部拼在一起，形成一個有 76400 pixels 的大.coe 檔，再用這個.coe 檔利用 **verilog** 產生 **Block_mem_gen**。這麼做的好處是再抓 **memory** 的時候可以更精準的知道每個圖形的位置在哪裡，就不會有圖片不乾淨的問題。

最後是 **Mem_addr_generator** 決定什麼時間要再螢幕中上方顯示什麼方塊。

3.2

3.2 和 3.1 的差別在於要能後讓方塊往下掉，所以我在決定方塊在螢幕上顯示位置的上下界部分做了改變，讓它會隨時間變化，如此便能達到往下掉的效果。需要特別注意的點是長方形俄羅斯方塊和其他方塊要分開處理，因為它的高度只有一個方形，但其他的高度有兩個。

5. Discussion

本次的 Lab 讓我更了解 **VGA** 的使用方法，也讓我的期末 **project** 更進一步。