|  |  |
| --- | --- |
| **Lab 3** | |
| 學號: 109021115 | 姓名: 吳嘉濬 |

1. **Lab Implementation**

以下是Lab3\_1的block diagram：

一張含有 文字, 筆跡, 兒童藝術, 圖表 的圖片

自動產生的描述

D flip flop可以暫存資料1個cycle，循環串聯連續5個D flip flops可以使資料每5個cycle輪回一圈。利用這個技巧，我們可以設計出每5個cycle一個循環的machine。

一張含有 文字, 圖表, 方案, 字型 的圖片

自動產生的描述

至於clock divider的部分，如果要設計一個頻率為原clock頻率/(2^n)的新clock，可以利用上圖的方法去設計，注意總共只需要n個DFFs就可以了。

Clock divider對應的code為：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

以下是利用#(XX)的語法去改變clock divider module的parameter n：



以下是Lab3\_1的FSM：

一張含有 文字, 筆跡, 字型, 行 的圖片

自動產生的描述

圖為Moore FSM。依照題義，可以直覺寫出lab3\_1的FSM。

在算出reg LD後，assign到wire led，加上constraint file把FPGA的pins map到Verilog I/O ports，即可讓FPGA板上的led登發亮。

利用以上clock divider的技術加上FSM，可以成功完成lab3\_1。

Lab3\_2是Lab3\_1的延伸，我將針對新增的內容作詳細說明。

以下為Lab3\_2的FSM：

一張含有 文字, 字型, 行, 筆跡 的圖片

自動產生的描述

根據題義可以畫出以上的FSM，並進而判斷出next\_LD。

一張含有 文字, 螢幕擷取畫面, 陳列 的圖片

自動產生的描述

上圖為在REGULAR mode和ESCAPE mode下判斷next\_LD的kernel code。不同於Lab3\_1，我直接以連續if … else if …來賦予next\_LD的值，而非細分成5個state；ESCAPE mode的部分，考慮不同的LD和dir決定下個cycle的LD和state。

Lab3\_3不同於Lab3\_1、Lab3\_2，他運用到了3個不同頻率的clock去影響著3條不同的蛇的移動，其中牽涉到對碰撞的反應。

以下為Lab3\_3的kernel code：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

左圖是Snake1在”自己的cycle”更新的資料，分別是dir1和pos1，即方向和位置，在這裡，我們不像前面lab直接存LD，因為這會使判斷boundary condition以及更新更複雜，於是我改存”位置”，詳細說明請見下面C part ”Problem Encountered”。

右圖是Snake11判斷自己next\_pos11和next\_dir11的地方。第一個if statement是default的情況，即沒有碰到boundary condition時一般採取的反應；第二個if statement是考慮遇到boundary condition時採取的應對，從code由上到下，依序是判斷：同時碰撞Snake1和Snake111時(即被夾在中間不能動彈)、只有碰撞Snake1時、只有碰撞Snake111時。根據遇到的boundary condition，會output出不同的next\_pos11和next\_dir11。

利用先前所提及實作clock divider的方法，加上3條蛇各自對應於左上方和右上方圖片的code，就能成功實作Lab3\_3。

1. **Questions and Discussions**

A. In lab3\_1, rst has the highest priority than any other signal (in most hardware

designs, it's true as well). How do you implement that?

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

在這個always block當中，rst是asynchronous positive reset，即rst一旦被trigger(set)，會立即執行always block(不用等到下一個clk\_div的posedge)；而在always block當中，第一個被優先判斷的就是rst signal，所以可以達到rst有highest priority的效果。

B. In lab3\_3, we simplify the direction policies of snakes to only apply at the moment

the snake is going to move. What if the policies affect every time snakes meet each

other? (Simply explain what you are going to change in your code).

在原本的code當中，我讓snake被trigger的當下，也就是snake對應的clk產生posedge時，再去看有沒有碰到其他蛇(或是牆壁)，做出相對應的判斷(下一個位置、下一個方向)。

至於如何達成Question B的效果，我們可以先有以下觀察：此特別情況發生在Snake1和Snake11相撞，或是Snake11和Snake111相撞。

我們可以在頻率高的蛇相撞的當下，去mark尚未被trigger、頻率較低的蛇。當頻率較低的蛇被trigger時，判斷是否被mark過，有的話就改變方向，並判斷出相對應的下一個位置。

1. **Problem Encountered**

一開始在設計lab3\_3時，以為可以比照lab3\_1和lab3\_2，直接算出每個cycle哪些led燈要亮，然後在module的末端assign led = LD;，如下圖：

一張含有 文字, 字型, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

結果在實作時發現，3條蛇能跑出的led output很複雜，如果先算出每條蛇的reg LD，利用”|” or operation去組合，最後再assign到led，會發現每條蛇的LD很難被直接計算出來。於是我改變儲存的資料，instead of存每條蛇會讓哪些led燈亮，我選擇去存它們的”位置”，我把”蛇的位置”定義為每條蛇最左邊的那個led位置，接著，我只要在最終assign led時，做一些修正，就能正確output三條蛇的樣子，即：

一張含有 螢幕擷取畫面, 文字, 字型 的圖片

自動產生的描述

Snake1對應1’b1<<pos1；Snake11對應2’b11<<(pos11-1’d1)；Snake111對應3’b111<<(pos111-2’d2)，我把這三項用”|” or operation去取聯集，即能得到想要的output。

1. **Suggestions**

謝謝助教在這次TA time幫我解惑。

這次lab3是第一次用FPGA板做lab，聽助教說以後的lab都是直接觀察FPGA板而非看waveform，這是不是意味著testbench的製作並非必要了呢，雖然我覺得製作testbench還蠻助於觀察個別訊號並debug就是了。

剛剛看了一下lab4感覺又是個兩三天跑不掉的作業量QQ，希望在實作時可以debug順利。