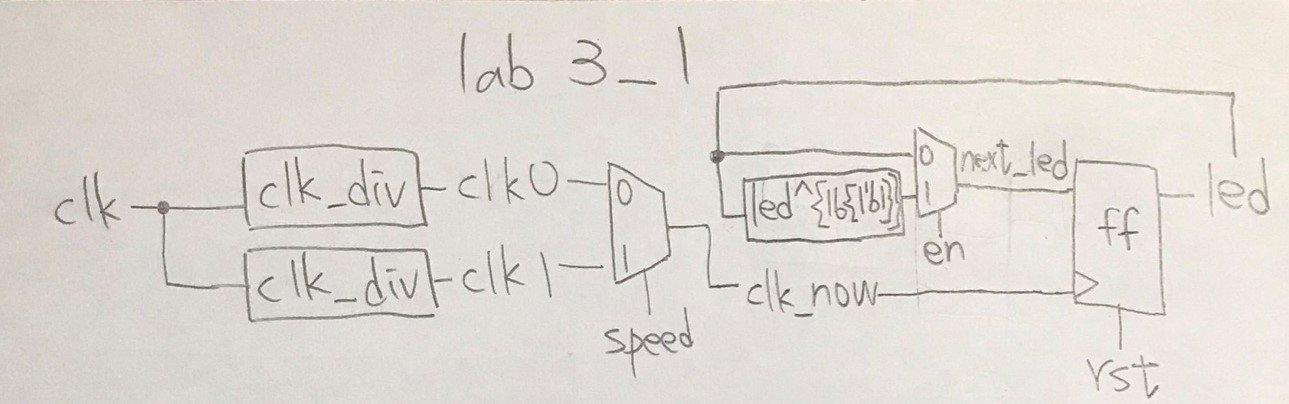
|  |  |
| --- | --- |
| **Lab 3** | |
| 學號: 108032053 | 姓名: 陳凱揚 |

1. 實作過程
2. clock\_divider

我以一個n bits的counter，每隔一個cycle加1，再將output接上這個counter的MSB，即可將clk除頻倍。

1. lab3\_1

如下圖1，我先利用前小題做出的clock\_divider，分別得到除頻和倍的clk1和clk0，接著以speed選出現在的速度，作為led的clk訊號，而led則是在en = 1的情況下，不停的在全亮和全暗間轉換。

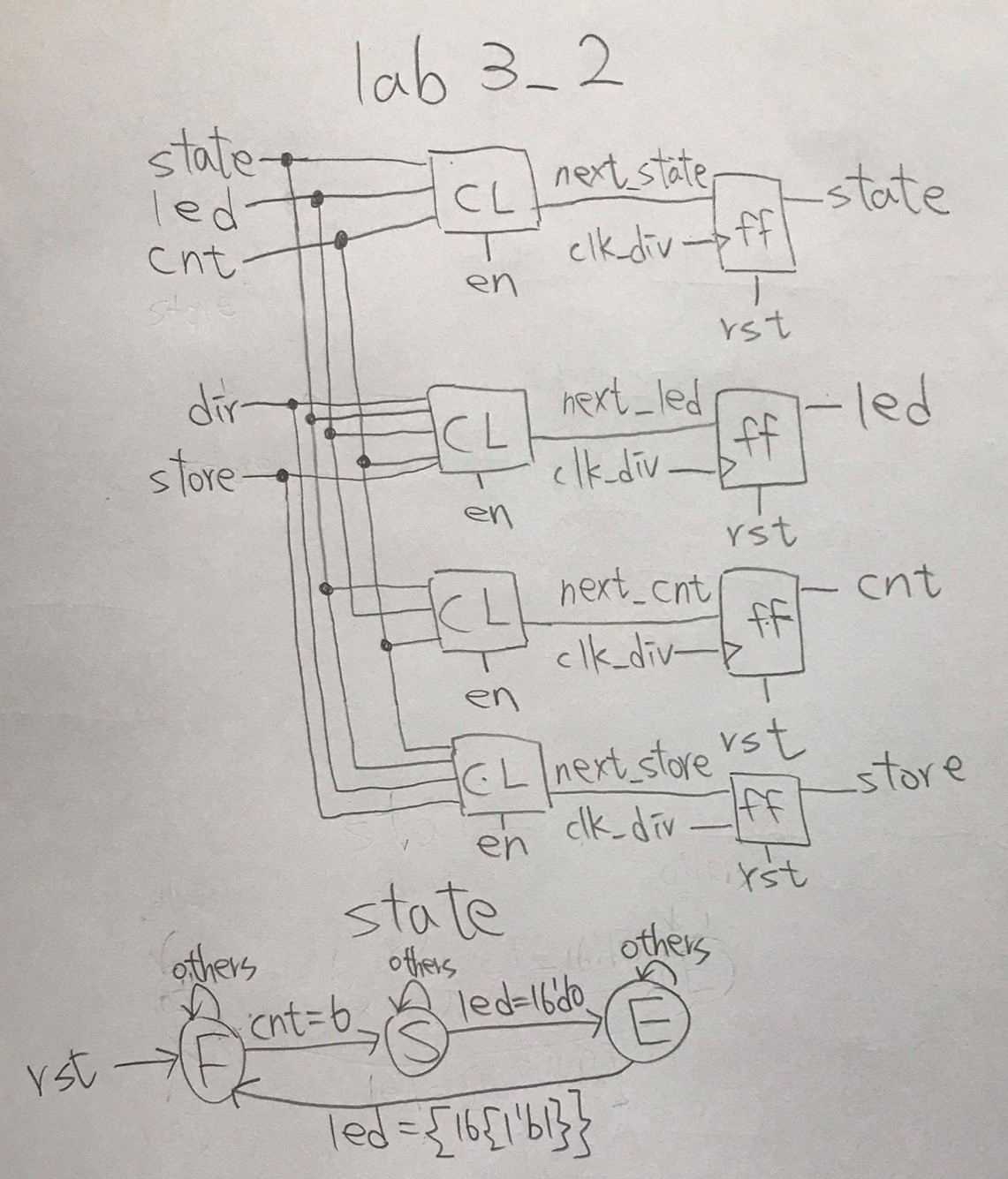


▲ 圖1

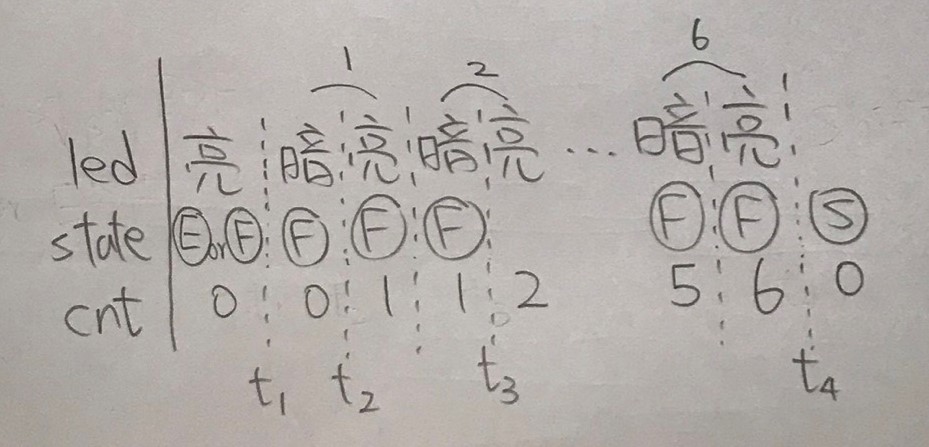
1. lab3\_2

如下圖2，我主要由4個filp-flop來記錄資訊，第一個是state，有FLASH, SHIFT, EXPAND等三種狀態，並依照state diagram來決定下一個state；第二個是led，會依照當前狀態及input來決定下一個cycle的輸出；第三個是cnt，於FLASH時使用，增加的條件為在FLASH狀態且led全暗，在其他state時皆為0，沒有功能，如下圖3，在t1時進入FLASH狀態，t2, t3時led全暗，因此皆增加1，接著在t4時，就恰好可以cnt = 6為條件進入SHIFT狀態；第四個是store，於SHIFT時使用，有48 bits，是led的3倍，用來儲存因為跑到邊界而不見的亮燈，可以在轉向時，依然保持有8顆亮燈出現，在其他state時，store皆沒有功能。

在code的部分，我運用了很多concatenate，像是在FLASH時，next\_led = led^{16{1’b1}}，在EXPAND時，next\_led = dir ? {1’b0, led[15:9], led[6:0], 1’b0} : {led[14:8], 2’b11, 1ed[7:1]}，來控制亮燈的擴張與縮小，而在SHIFT時，store的中間16個bits即代表led目前的亮燈，即使led2的亮燈因到達邊界而消失，store的左右16個bits依然能夠儲存住資訊，在dir轉向後，以 >> 和 <<推回原先的位元。



▲ 圖2

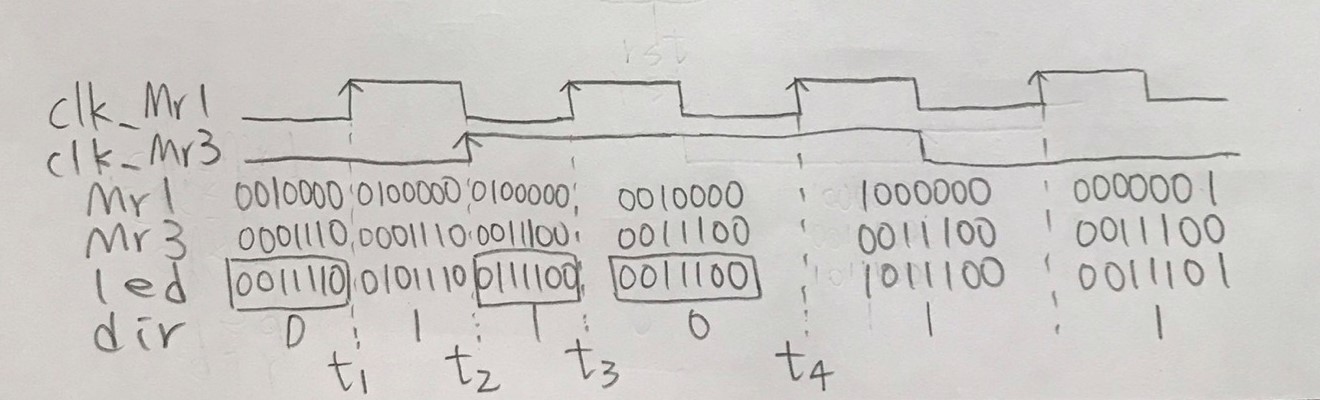


▲ 圖3

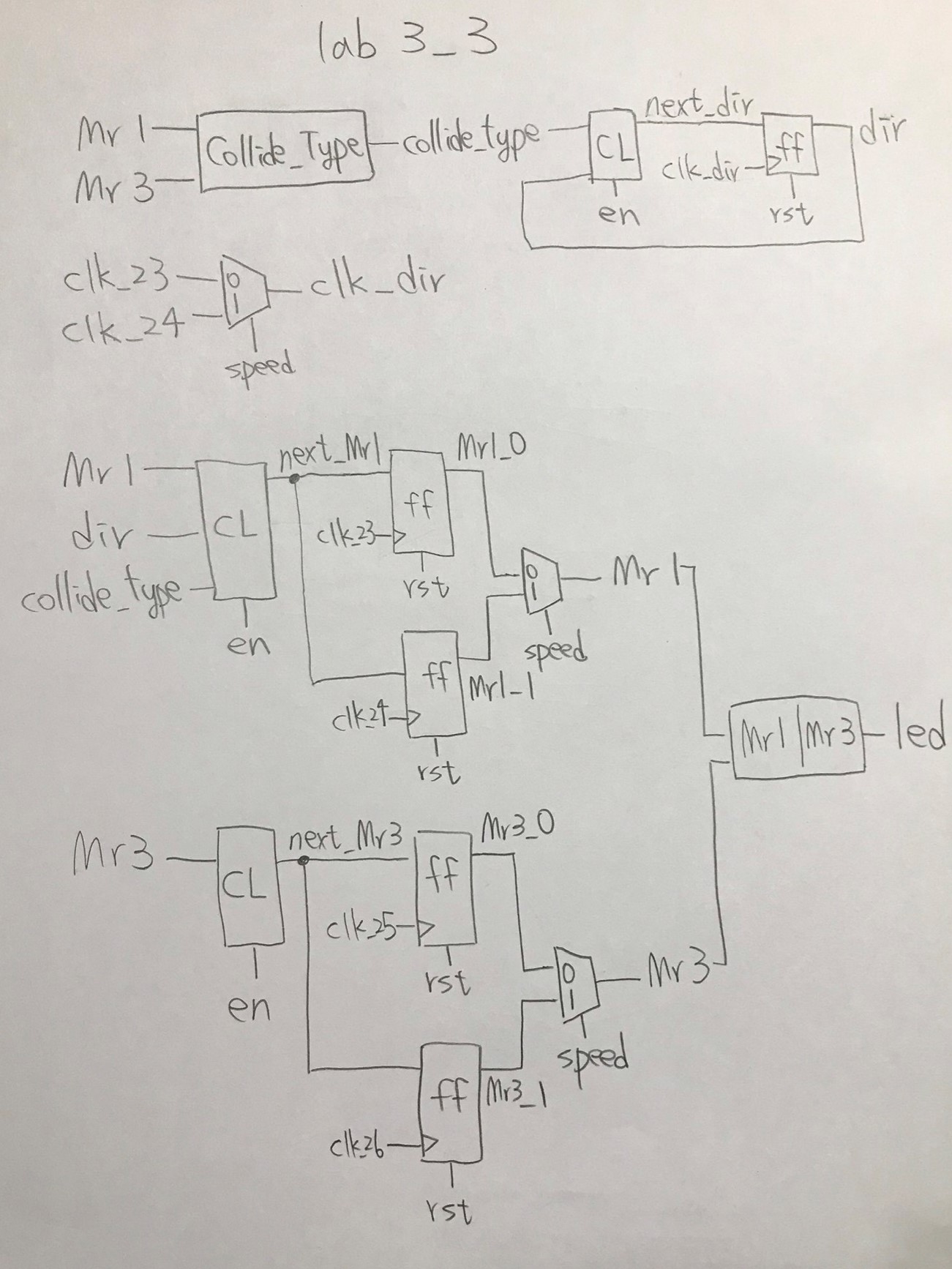
1. lab3\_3

如下圖5，先看下方的大架構，我以4個flip-flop儲存Mr1和Mr3的各兩種不同速度更新位置，再以speed作為select選出現在的Mr1和Mr3，而led = Mr1 | Mr3。在計算Mr3的下一個位置相當容易，就只要一直向左前進即可，next\_Mr3 = {Mr3[14:0], Mr3[15]}，而計算Mr1的下一個位置比較複雜，需要以目前的碰撞情況、目前前進方向、目前位置來決定。碰撞情況我寫成一個module來判斷，output有四種結果，NONE、TOUCH、OVERLAP\_EDGE、OVERLAP\_MID，而Mr1的下一個位置即是以collide\_type來決定碰撞時要往反向跳幾步，像是TOUCH只要往反向跳1步就能離開碰撞，另外兩種OVERLAP的情形分別是往反向跳2步和跳3步才能離開碰撞，NONE只要依原始方向前進一步，而collide\_type也會用來決定下一步的dir。

至於為什麼在一發生碰撞就會反向的規則下，還是有可能發現OVERLAP的情況，是因為Mr1和Mr3的clock不相等，以下圖4來看，Mr3的clock週期是Mr1的4倍長，原先Mr1向右前進，在t1時發生碰撞，collide\_type為TOUCH，因此向後退一步，方向改為向左；在t2時，Mr3向左前進一步，造成了在t3時，又發生了一次TOUCH碰撞，但這時Mr1的方向為向左，因此他往反向跳一步，反而更接近了Mr3；則在t4時，即檢查出了OVERLAP\_EDGE碰撞，讓Mr1向左跳了兩步，這時才結束了不停碰撞的情形。



▲ 圖4



▲ 圖5

1. 學到的東西與遇到的困難

我覺得活用位元運算和concatenate在硬體設計中很重要，能夠省下一些不必要的flip-flop，不僅讓程式碼更簡潔，設計出來也更有效率，而且在檢查某些狀態時，位元運算相當好用，可以很快的算出結果。此外，我覺得try and error也很重要，像是我一開始無法想出為什麼lab3\_3的碰撞會發生重疊的情形，所以我就先假設這個情況會發生並先設計實作，且自己寫testbench、跑模擬、觀察波形圖，透過這些方法，我很快就找出了發生的原因，比起憑空想像還要更有效率了許多。

1. 想對老師或助教說的話

有一天薯條家庭在聊天

孫子：爺爺你怎麼那麼怕貓

…

…

…

…

…

爺爺：因為我是老薯