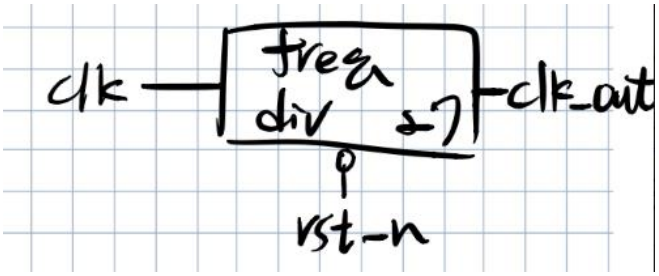
# Lab3 Report

## 3-1. Frequency Divider for $\frac{1}{2^{27}}$ clk




**●Design Specification**

**1. Input:** clk , rst_n

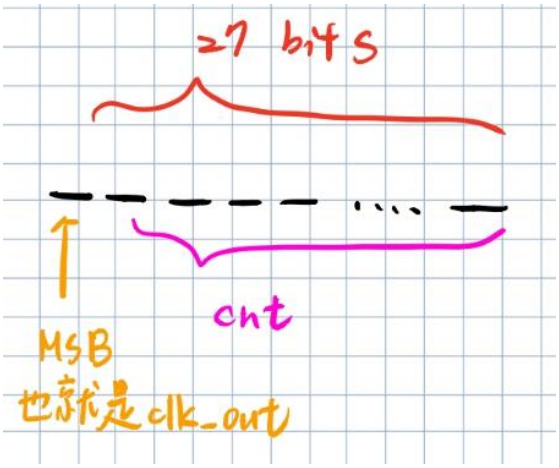
**2. Output:** clk_out

**3. Reg:** cnt,cnt_tmp,cnt_out;

(clk 用來對應板子本身的 Fcrystal; rst_n 是 reset; clk_out(MSB)是我們所要的結果; cnt, cnt_tmp, cnt_out 則都是中間會使用到的變數)

**●Design Implementation**

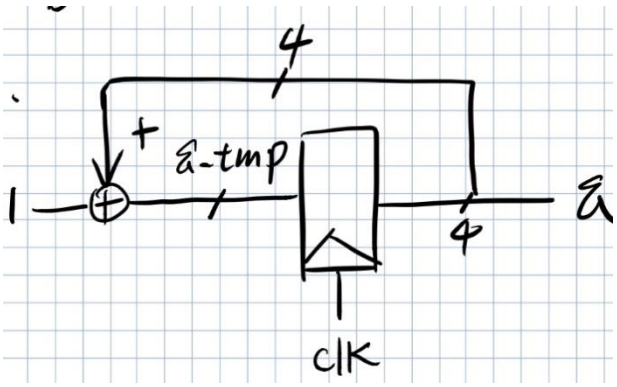
**1. Outline**




本功能接收板子本身的 clk(Fcrystal)(W5)來做除頻，因為原本的頻率太高，需要降低頻率來做更多運用。

因為這個實驗所需要的 clk_out 頻率是 2 的整數次方倍，因此只需要利用 overflow 的特性，讓 cnt 持續加一直到 27 位元滿之後(MSB=1’b1)，LED 燈會亮，之後前 26 六位數又會重新歸零，而產生一個週期是 Tcrystal $2^{27}$ 倍(約為一秒)的 Frequency Divider。

**2. Logic Diagram**




**3. Logic function**

cnt_temp= {clk_out,cnt}+ 1’b1

| I/O | Clk_out | Clk | Rst_n |
|---|---|---|---|
| Pin | U16 | W5 | V16 |

**●Stimulation**

此實驗用 FPGA 板展示，已在實驗課 DEMO 過。

**●Discussion**

1. 從這幾次的實驗開始，我有一點開始不太清楚要在結報 Design implementation 中要寫什麼，因為感覺寫出來的樣子就有點像是 verilog 中的程式碼了，但是馬教授說不可以直接貼程式碼上來，不然分數會很低，所以我就盡可能的說出我的設計方法，還有原理，之後上課再詢問教授或助教比較好的表示方法。

2. 這個實驗中的大部分程式碼都在馬教授的講義可以找到，雖然我打的大部分都跟講義上面一樣，但我還是花時間去理解裡面每一行的意義。

3. 除完頻的頻率大約是 0.75Hz 比下一題的 1Hz 還要慢了一些。

4. 做完這個實驗讓我學會了要如何改變板子上原本的 clk 來實現我們更多的應用，還有 basic circuit verilog 的打法

## 3-2.Frequency Divider for 1Hz

**●Design Specification**

**1. Input:** clk , rst_n

**2. Output:** clk_out

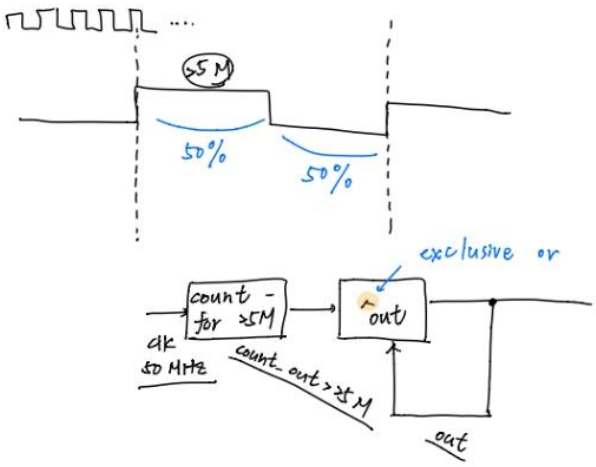
**3. Reg:** cnt,cnt_tmp,cnt_out;

(clk 用來對應板子本身的 Fcrystal; rst_n 是 reset; clk_out(MSB)是我們所要的結果; cnt,cnt_tmp,cnt_out 則都是中間會使用到的變數)

**●Design Implementation**

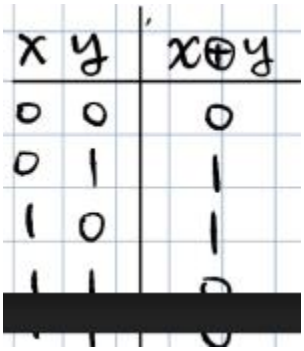
**1. Outline**

這題目有些部分跟上一題有點像，但是比較不一樣的部分是題目所希望的 clk 震盪頻率是 1Hz，不像上一題會是原本板子 Fcrystal 的 2 的整數次方倍，所以不能直接像上一題一樣簡單利用 MSB 接上 LED 發光。

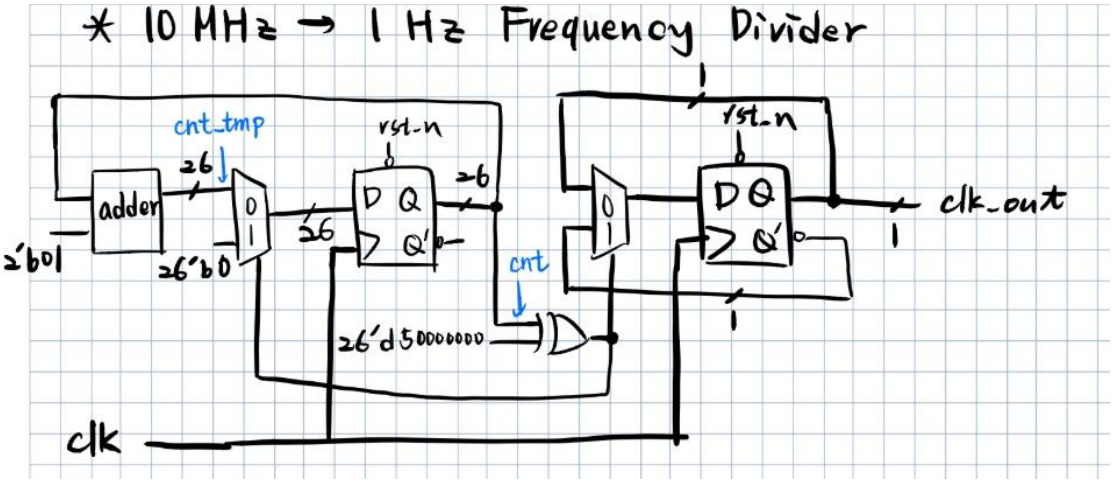



要利用 duty cycle 中有 50%是

0，50%是 1 的特性來製作。首先是要讓 cnt(26 bits)接上加法器持續加 1，後面再利用⊕(XOR)的特性(如附圖)，當 cnt=50M 的時候，XOR 結果出來為 0,會影響到左右兩個 Mux，造成右邊的 T-flip flop Toggle(clk_out 轉為~clk_out),產生 f=1Hz 的波型，也會讓 cnt 歸零，產生 1hz Frequency Divider。




**2. Logic Diagram**




**3. Logic function**

cnt_temp= {clk_out,cnt}+ 1'b1（但是 clk_out 主要的用處是在後面的 T-flip flop）

| I/O | Clk_out | Clk | Rst_n |
|---|---|---|---|
| Pin | U16 | W5 | V16 |

**●Stimulation**

此實驗用 FPGA 板展示，已在實驗課 DEMO 過。

**●Discussion**

1. 在我一開始的程式碼中，我將大部分的敘述都排在 always @*中而不是 always @(posedge clk or negedge rst_n)中，但是這樣跑出來會有非常多的 bug 但我不太清楚實際的原因是什麼，之後會在實驗課的時候詢問助教。

2. 這感覺是一個重要的原件，因為很多其他的 module 都會需要用秒來當 clock，也想到我之前悲慘的期末考，聽到助教說要 50%的 duty cycle 的時候還完全聽不懂那個是什麼，直到下學期才明白……

3. 做完這個實驗讓我學會了如何自由地操作板子上的 clock 的頻率。

## 3-3. Shifter demos on FPGA

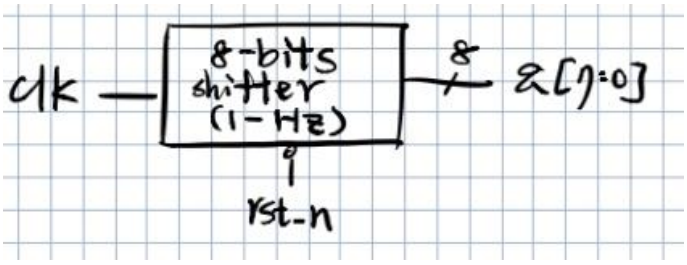
**●Design Specification**

**1. Input:** clk , rst_n

**2. Output:** q[3:0]

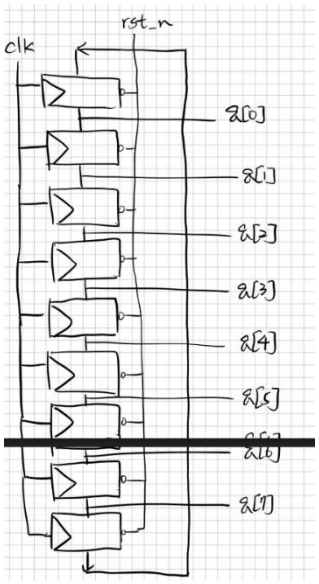
**3. Wire:** clk_d,q

(clk_d 是用來連接 Frequency Divider 和 Shifter 兩個 module 的 clk 用的)




**●Design Implementation**

**1. Outline**

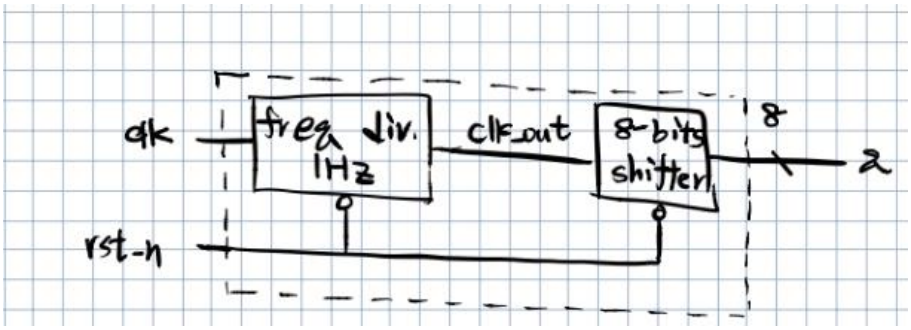



這題是要利用 pre-lab2 和 lab3.2 當作子 module 做成的 top module 來完成的題目，要完成一個燈會隨著 1Hz 頻率而移動的燈條。

Frequency divider 的設計原理在上面，所以主要說明 shifter 的原理。這邊以 3bits input 的 shifter 做說明，如左圖。

要先給[7:0]q 初始值 01010101，利用已經除頻的 clk 來做 D-Flip Flop，每到 trigger 的時候 q[0]<=q[7],q[1]<=q[6]……以此類推。會產生，每個燈看起來都只在左右搖擺的錯覺。

**2. Logic Diagram**



| I/O | q[7] | q[6] | q[5] | q[4] | q[3] | q[2] | q[1] | q[0] | Clk | Rst_n |
|---|---|---|---|---|---|---|---|---|---|---|
| Pin | V14 | U14 | U15 | W18 | V19 | U19 | E19 | U16 | W5 | V2 |

**●Stimulation**

此實驗用 FPGA 板展示，已在實驗課 DEMO 過。

**●Discussion**

1. 每次打錯一小部分的時候，要重新用 FPGA 板跑的時候都要再用一次 Generate Bitstream，實在要花很多時間阿……，所以之後我都學乖了，利用這個時間可以拿來打結報，妥善運用時間。

2. 這次實驗讓我第一次使用其他的 module 來做成一個 top module，讓我感受到這樣硬體語言的方便性，總覺得與之前學過的 c++有種既視感。

**●Reference**

1. 在 XILINX 的網站中，查到我錯誤的程式代碼(在 Report 中寫的)錯誤原因。因為實在是太長然後看不太懂，所以上網查詢資料。

i.e. ERROR: [Drc 23-20] Rule violation (NSTD-1)……

（錯誤的原因是我在把在做 xdc 的時候，忘記換到 I/O std 了，就是原本默認選項，結果我因為這個而看了大概一個小時……）

2.好奇常常案到的 “BITSTREAM”到的是什麼意思，而去查了維基百科，結果如下:

「一個位元流（bitstream 或 bit stream）是一個位元的序列。一個位元組流則是一個位元組的序列，一般來說一個位元組是 8 個位元。也可以被視為是一種特殊的位元流。」，有種似懂非懂的感覺。

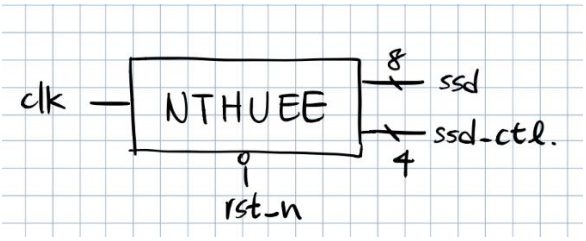
## 3-4. NTHUEE Shifter

### ●Design Specification

**1. Input:** clk , rst_n

**2. Output:** [3:0] ssd_ctl,[7:0]display

(clk 是板子上原本的時脈，rst_n 是 reset，ssd_ctl 是控制四個七段顯示器顯示的頻率，display 則是控制七段顯示器的亮暗)

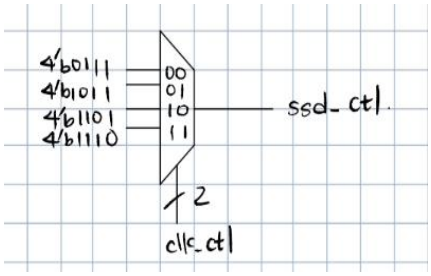



### ●Design Implementation

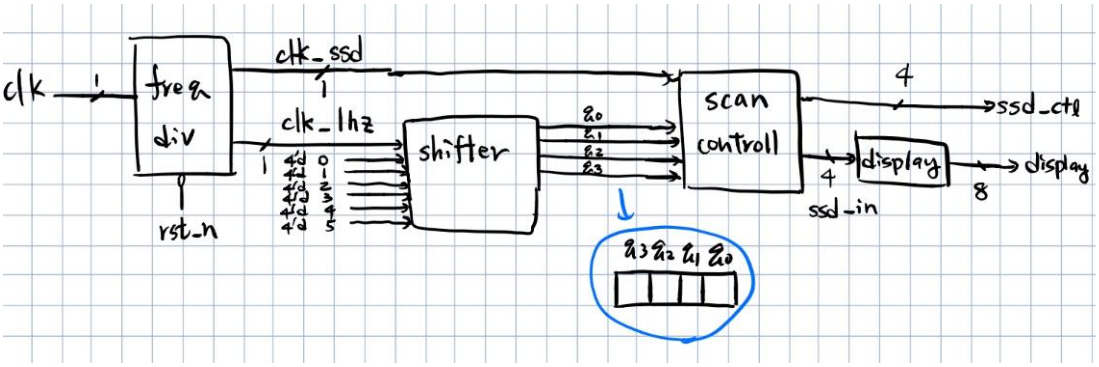
**1. Outline**

這個實驗主要是利用七段顯示器做出一個顯示 NTHUEE 的跑馬燈，會利用到很多之前實驗的 module 功能。

首先要先利用 frequency divider 把原本的 clk 轉成兩種，一個是頻率 1hz，一個則是利用人眼有視覺暫留而取的頻率(clk_ctl)(讓七段顯示器看起來是同時亮的)(如右圖)，之後再利用 shifter 讓每個每個 D flip flop 在 clk_1hz trigger 的時候把自己存的數值到相鄰的 D flip flop，做成跑馬燈的效果。之後的 scan control 則是利用 clk_ssd 控制每個七段顯示器顯示的時間，讓人眼看起來像是四個都亮著不同的文字，最後的 display 則是類似 decoder，把原本 binary 的數字解碼成 NTHUE 五種文字，最後顯示在顯示器上。




**2. Logic Diagram**




| I/O | d[7] | d[6] | d[5] | d[4] | d[3] | d[2] | d[1] | d[0] | Clk | Rst_n |
|---|---|---|---|---|---|---|---|---|---|---|
| Pin | V14 | U14 | U15 | W18 | V19 | U19 | E19 | U16 | W5 | V17 |

**(d=display;s= ssd_ctl,)**

| I/O | S[3] | S[2] | S[1] | S[0] |
|---|---|---|---|---|
| Pin | W4 | V4 | U4 | U2 |

## ●Stimulation

此實驗用 FPGA 板展示，已在實驗課 DEMO 過。

## ●Discussion

**1.** 這題真的非常困難，原本天真地想要自己直接寫出來，但是奮鬥了三四個小時發現真的是錯誤一堆，很難 **debug**，於是就開始翻講義、到處找資料，才生了出來。

**2.** 寫完這題之後讓我更清楚 **reg**、**wire** 的差別，之前我一直搞不清楚什麼時候要用 **reg** 什麼時候要用 **wire**，但其實 **wire** 只有用在兩個地方，一個是 **always@***中左邊的變數，還有 **testbench** 中而已。

**3.** 這讓我知道了不同工作 **module** 的重要性，要是我之前都有好好把各種功能的 **logic diagram**、還有 **design inplementation** 弄懂的話，我想也不會讓我花到整天才把這個寫出來…….

## ●Reference

**1.**在 **ptt** 的 **electronics** 版中，了解**=**、**<=**的差別，前者是代表要等待上一行回應後，在執行；後者是代表不等待上一行回應，完成馬上執行，也就是說觸發的時候同時執行，所以是用在 **flip-flop** 中。

**2.**在 **Verilog HDL** 教學講義中，了解 **reg** 是有記憶性的，預設值是 **x(**最好要初始化**)**；連接線 **Net(wire**、**wand**、**wor)**是沒有記憶性，預設值是 **z**，而且絕對不能把兩個 **wire** 連在一起。

## ●Conclusion

寫這次的作業是我花過最久時間完成的作業，幾乎把我周末兩天的時間都用光了，就是第四題這個大魔王，讓我把程式碼重新砍掉再寫好幾次，可能是我對 **verilog** 的語法還不夠熟悉才會有這樣的結果吧……

完成最後一個實驗的時候還蠻有成就感的，原來路上隨處可見的跑馬燈或是時鐘，要讓它顯示需要這麼多的步驟還有原理，真的是越讀書越發現自己的苗小阿，真的還有很多東西需要學。**(**很感謝以前各種偉大的工程師讓我們現在的生活這麼便利**)**