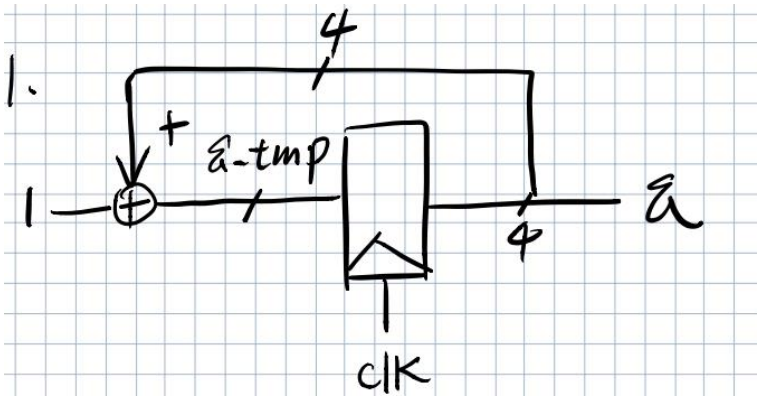
Pre-Lab3 Report

3-1. 4-bit synchronous binary up counter

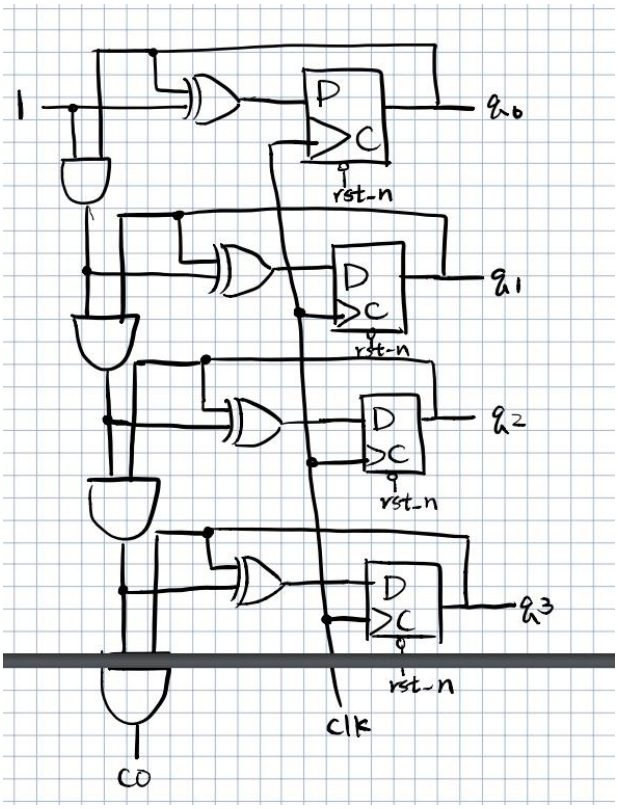
●Design Specification

1. **Input:** clk, rst_n;
2. **Output:** [3:0]q;
   (reg [3:0]q)因為 q 需要持續變動
3. q=q+1'b1; (當 clk 有變動的時候)
   q 會循環從 0 數到 15

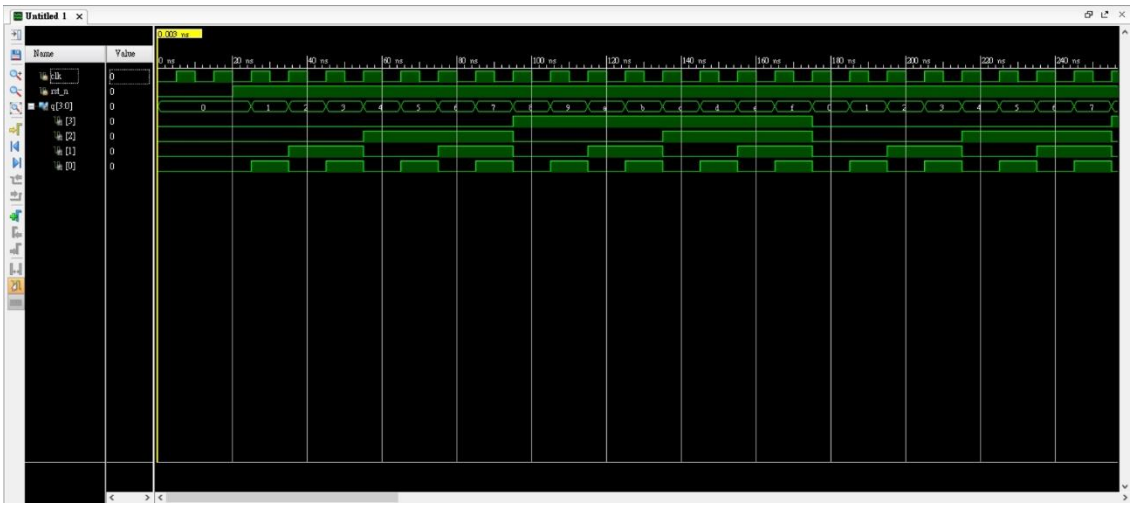


●Design Implementation

1.Logic diagram:



●Stimulation



●Reference

1. 參照網路資源(FPGA 4 student)，知道”forever”的用法。

Ex: 在 testbench 中

```
forever #5 clk=~clk;
```

即可讓 clk 定時持續變動

●Discussion

1. 測試結果與原本預測一樣，binary add up counter 持續每 clk 往上加到進位並　　循環。

2. 在網路上的資料中，testbench 不知道為什麼不用給 q[3:0]起始值，之後上課再請問教授或助教。

3. 在 Design Implementation 當中的 logic diagram 為馬教授上學期的邏輯設計課本中的資料，我在想是不是應該放最簡略的那個 block diagram 就可以代表這題的 logic diagram 了，因為在在 verilog 的程式碼中是寫 q_tmp=q+‘BCD_ONE; 而已，所以用比較簡略的 block diagram 來描述我覺得會比較好一點，不過為了保險起見我還是兩張圖都放了當作補充。

4. 在 testbench 中的我再最後測試的時候一直跑出錯誤的訊息，找了老半天，原來是忘記加 begin，而在 Report 也沒有寫得很明確說哪裡附近有問題，於是我就這樣浪費了好幾個小時在 debug……

3-2. 8 DFFs shifter register

●Design Specification

1. **Input**:clk,rst_n;
2. **Output:** [7:0]q;
   (reg [7:0]q)因為 q 需要持續變動
   q 的起始值為 8'b0101_0101;

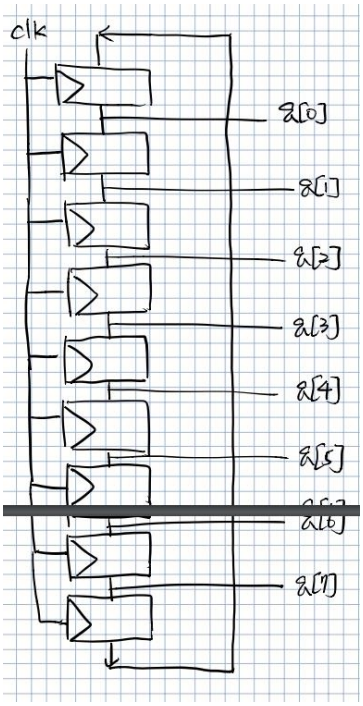
●Design Implementation

1.logic equation:

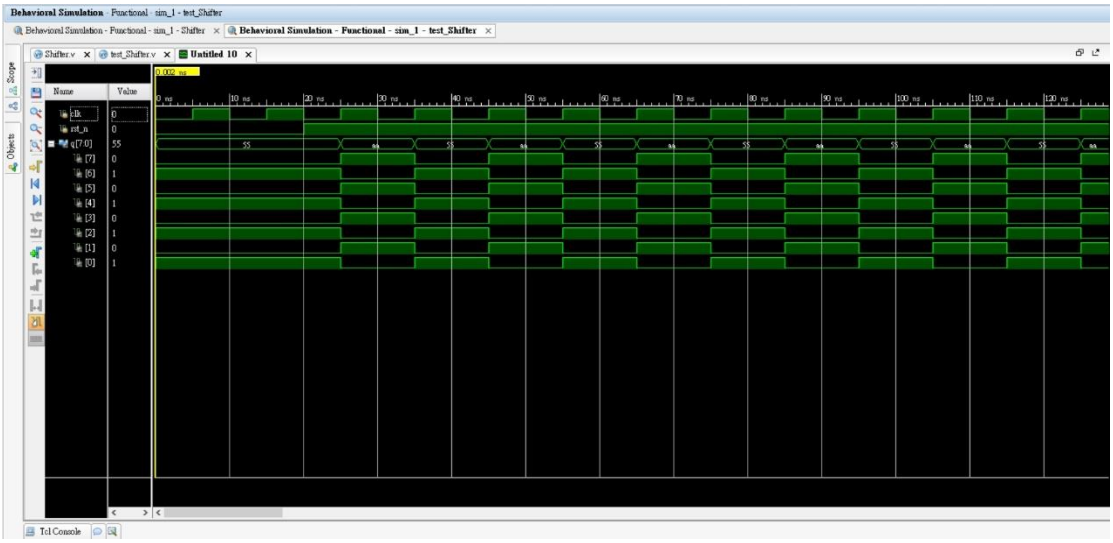
q[0]<=q[7];
q[1]<=q[0];
q[2]<=q[1];
q[3]<=q[2];
q[4]<=q[3];
q[5]<=q[4];
q[6]<=q[5];
q[7]<=q[6];

(當 clk 有變動且 rst_n 不等於 0 時)
就是 01010101 跟 10101010 一直互換

2.logic diagram:



●Stimulation



●Reference

1. 同樣參照網路資源(FPGA 4 student)，知道”forever”的用法。(兩個的testbench 基本上大同小異)

Ex: 在 testbench 中

```
forever #5 clk=~clk;
```

即可讓 clk 定時持續變動

●Discussion

1. 結果與實際預測一樣，Shifter register 如模擬中 01 一直互換，會跑出如棋盤格子一一般的圖案，看起來十分舒服。

2. 這個實驗就比較順，因為基本上會卡住都是因為對這個語言還不夠熟悉，常常少一點東西模擬跑出來就是藍色紅色好幾條，但是因為這次大多程式都已經在講義上了，其實只是改幾個數字放上來而已，況且兩個的 testbench 幾乎一樣，已經比較能掌握自己在做什麼了 XD。

●pre-lab3 Conclusion

這次的 prelab 只要寫 module 不用燒到 FPGA 板上，因此要寫 testbench。不過因為 sequentail ciucuit 的 testbench 比較不一樣，所以我有上網去查其他資料參考。這次實驗我學會了如何使用 implement counter 來表示 clk、D flip-flop verilog 的打法。

邏輯設計實驗課真的是一堂會讓我唯一想要在課後主動做作業的一門課(通常都是被 deadline 追殺)，因為我十分喜歡自己學過後真正自己運用出來的並且能改善自己的生活(雖然我現在還差很遠)，能夠利用自己寫程式碼去驅動一塊板子做出自己理想的設計十分的有成就感，期待未來能學到更多東西來完成更有用的作品。