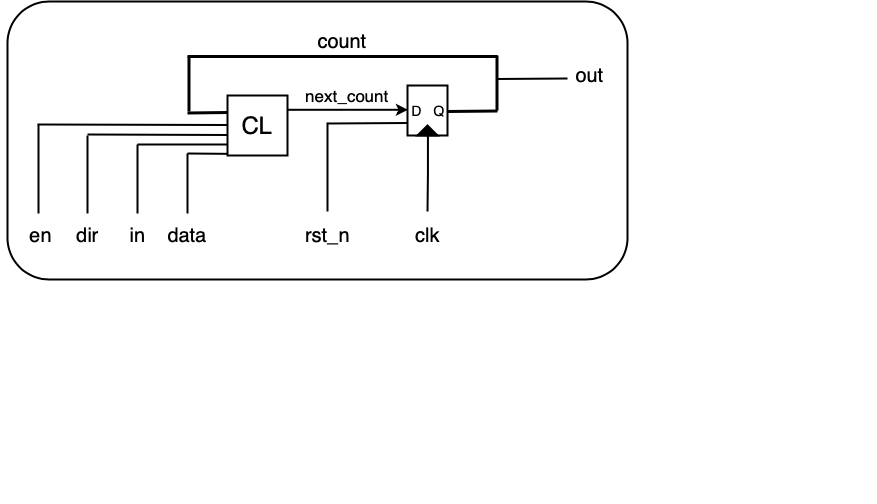
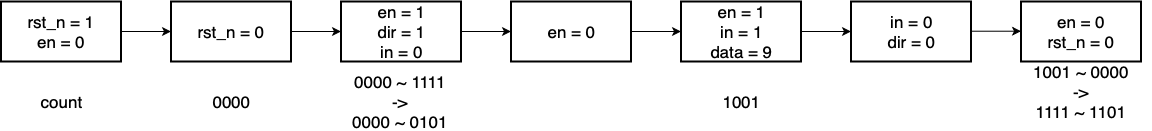
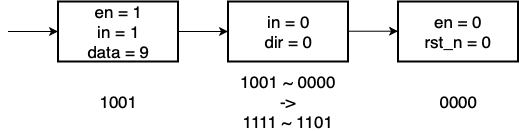
|  |
| --- |
| **EECS 2070 02 Digital Design Labs 2019**  **Lab 1** |
| **學號：107000115 姓名：林珈卉** |

1. 實作過程
   1. lab02\_1.v
      1. Block Diagram

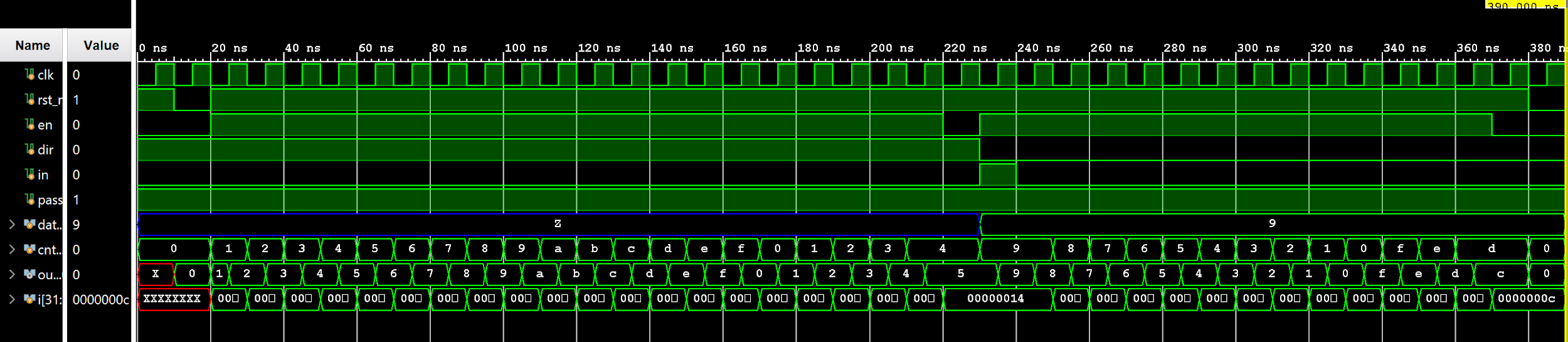


* + 1. 設計概念
       1. 宣告兩個 reg [3:0] 變數，count 及 next\_count
       2. 設置兩個 always block
       3. 一個 always block 由 posedge clk 及 rst\_n 來 trigger，負責將 next\_count 的值設給 count
       4. 另一個 always block 則判斷現在的 in 與 dir 的值，而決定要將 next\_count 加一、減一、不變，或者將 data 的值 assign 給 next\_count
       5. 每次 posedge clk 都將 next\_count 的值送給 count，形成一個 Combinational Circuit
       6. 將 count 的值 assign 給 out
  1. lab02\_1\_t.v
     1. 設計概念
        1. 將 clk 的 cycle 設成 10
        2. 設置 pass，初始值為 1，若有錯誤則變為 0
        3. 使用 for 迴圈檢查 out 的值，並將 tb 裡的 count 加一、減一
        4. 分成 7 個階段
           1. 程式開始時，rst\_n 為 1，en 為 0，count 尚未有初始值
           2. rst\_n 降為 0，count 被 reset 為 4’b0000
           3. en 升起為 1，dir 為 1，count 隨著 clock cycle 漸增，從 4’b0000 往上跑至超過 4’b1111，並回到 4’b0000 繼續往上增加
           4. en 降為 0，count 的值不動
           5. en 回到 1，in 也升起至 1，並將 data 的數值 4’b1001 送給 count
           6. in 降回 0，dir 為 0，count 隨著 clock cycle 漸減，從 4’b1001 往下跑至超過 4’b0000，並回到 4’b1111 繼續往下減少
           7. en 降回 0，rst\_n 降至 0，count 被 reset 為 4’b0000
        5. 若程式結束後 pass 仍為 1，印出 “ALL PASS!!!” 字樣

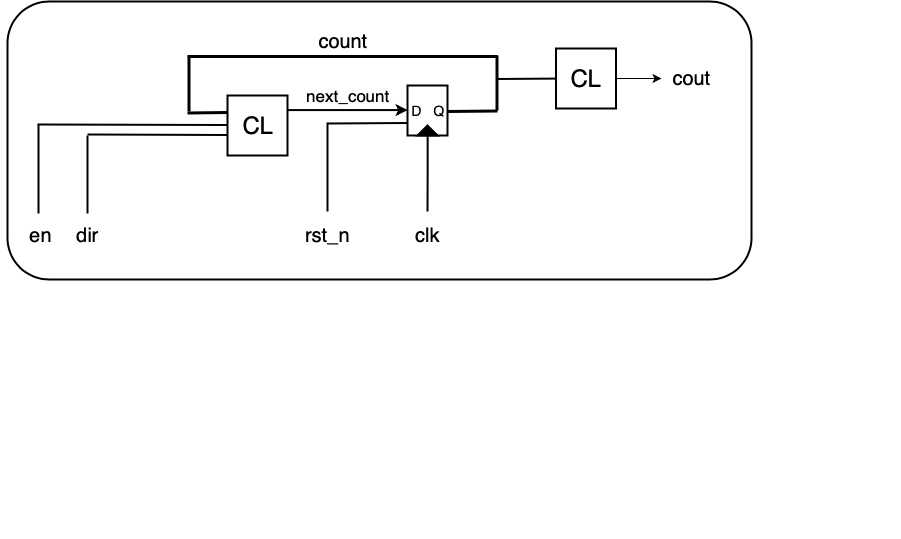




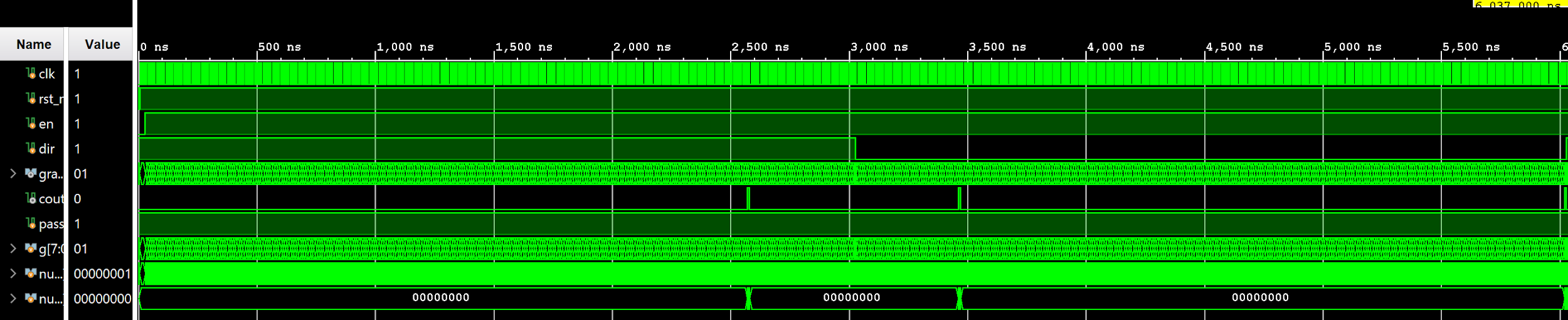
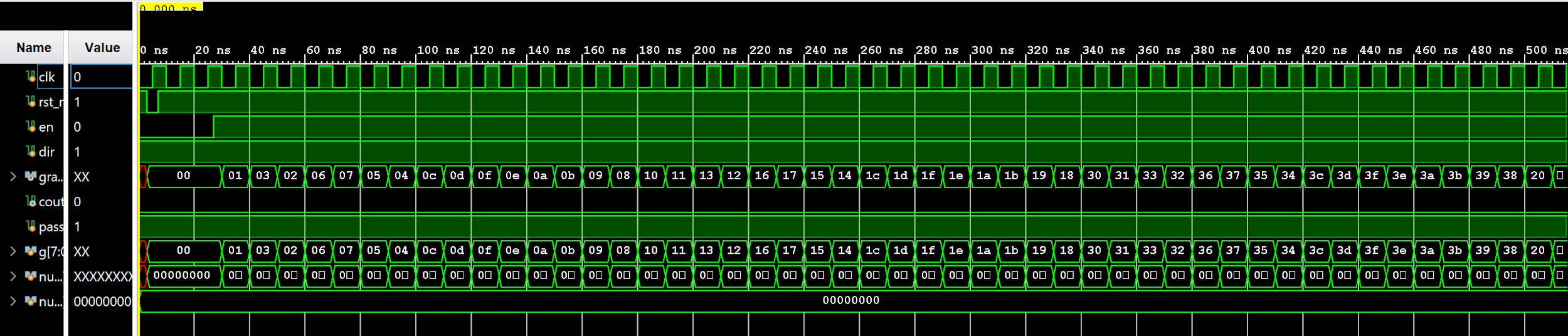
* + 1. Waveform



* 1. lab02\_2.v
     1. Block Diagram



* + 1. 設計概念
       1. 在原本有的 module lab2\_2 以外，再開一個 module gray\_1\_digit
       2. 在 lab2\_2 呼叫兩次 gray\_1\_digit，將 8 bits 的 output gray 分為兩個 4 bits 接至 gray\_1\_digit 去計算
       3. 在 gray\_1\_digit 中，宣告兩個 reg [3:0] 變數 count 及 next\_count，負責擔任 4 bits 的二進位的 counter，使用兩個 always block
       4. 在 gray\_1\_digit 中，gray[0] 為 count[0] 與 count[1] XOR 的結果，以 gate level 的方式寫出，gray[1] 及 gray[2] 亦是以這個方式寫出
       5. gray[3] 被 assign 為 count[3] 的值
       6. 將 gray[3:0] 部分的 cout 用 wire 變數接到 gray[7:4] 部分的 en
    2. Waveform



* 1. lab02\_3.v lab02\_3\_t.v

1. 學到的東西與遇到的困難
   1. 在 always block 中被賦值的變數皆需要是 reg，因此 lab02\_2 中 module gray\_1\_digit 的 cout 必須宣告為 output reg cout。
   2. 在呼叫 module 時一定要加上物件名稱，否則會出現 bug，且在 Console Bar 並沒有將錯誤原因寫得很明瞭，會很難找出錯誤。
   3. 在使用 xor x\_0(out, in1, in2) 時，好奇為何不須像呼叫 module 時，要寫 .out(out)。上網查資料後了解，原來這是 In Order 的寫法，只要照著既定順序寫上輸出、輸入的變數即可，而另一種方式則稱為 by Name。
   4. 減法不能寫為 next\_count = count - 4'b0001，必須寫為 next\_count = count + 4'b1111。
   5. 在撰寫 tb 時，要注意設定停留時間的長度，否則很容易跟另一個檔案執行的週期不同步，在驗證答案時將會出錯。
   6. 在撰寫 tb 時，使用 $display 來 debug 時最好將當時數值以 %b 輸出，可以更簡單明瞭。
   7. always #5 clk = ~clk 若寫成 (#5) 將會出錯。
2. 想對老師或助教說的話

* 謝謝助教在 lab 2 demo 時，非常細心地回答我的問題，也幫我檢查程式的邏輯錯誤，甚至為了學生們多留在教室將近半個小時！