1. 這次作業的我可以說是寫了兩次。第一次是剛出的時候自己寫的，我當時以為 register 開越少面積就會越小，雖然這兩者有些關係沒錯，但好像影響不大，主要還是看合成的結果。透過把一些程式碼換位置(和 HW01 的想法一樣)，我把面積從 28XX 壓到 22XX，同時也發現了若是要把 latency 壓到 0 的話，out_valid 可以直接和 cnt 相等，所以我這部分也少用了一個 D-flipflop。最後我只開了 cnt, cnt_minus_one 以及 out_reg 三個 register 就完成了第一次嘗試。
2. 然後室友寫完之後我們對了一下面積，他竟只有 16XX，所以我就看了一下，然後發現他開了一堆 register，邏輯的部分我擔心會被影響思考所以就大概看ㄌ一下而已。後來我就回去修我的作業，才知道原來導致面積變大的不是 register 的數量，而是那些一顆一顆的 module 總共開了幾個。舉例來說，我原本的程式有一部分是圖一那樣，然後把重複的部分用另外一個 assign 直接寫，透過這樣，我的面積就從 22XX 掉到 16XX。

```
if(cnt == 1) begin
  out_reg = (out + in_num*2/10);
  out_reg = (out_reg + in_num*2%10)%10;
  out_reg = (out_reg == 0) ? 15 : 10 - out_reg;
end
else begin
  if(cnt[0] == 1) begin
    out_reg = (out + in_num*2/10);
    out_reg = (out_reg + in_num*2%10)%10;
  end
  else begin
    out_reg = out+in_num;
  end
end
```

圖一

```
assign Greater_than_ten = (in_num > 4) ? 9 : 0;
assign Rem = (cnt[0] == 0) ? in_num : (in_num*2 - Greater_than_ten);
assign Add = (cnt == 0) ? 0 : (out+Rem)%10;
assign out_reg = (cnt == 1) ? (Add == 0) ? 15 : 10 - Add : Add;
```

圖二

3. 想特別提到的就是 Rem 那一行，Rem 代表的是那個 cycle 要加到 out 的值，所以才會看到 in_num 要乘上的加權。正常來說會寫成 in_num*2/10 + in_num*2%10，但因為我們的數字小，前面那項只會是 1 或 0，所以可以再寫成 1 + in_num*2%10，然後我的 Vscode 就自己跳出

```
Greater_than_ten + in_num*2 - 10 * Greater_than_ten
```

所以最後就直接把它化簡成看要減 9 還是不變。原本寫成 9*1 的話會再多一個乘法器，我改掉之後面積少了 10，所以感覺一個單元在工作站的 EDA tool 應該面積都是 10?

4. 附上兩次架構圖：