HW02 Identification Number Check

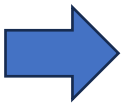
1.設計方法 & 遇到的困難與如何解決

由於本次設計中，in_id 值是由負源提供的，而後續的計算需要正向源輸出，這導致只有半個 cycle 可以進行運算。為了避免 external delay 所造成的 time violated，我在 input port 以及 output port 都使用了 DFF 來隔離，這樣可以確保這些運算具有完整的一個 cycle 進行運算。在權重部分，我利用計數器的設計來對應每個權重，並確保最後的檢查號碼計算不會將總和計算在內。

然而，這種設計導致了 3000cyles 的 latency。我將 out_legel_id 和 out_valid 拉到 in_id 提供的第 10 個 cycle，也就是最快可以輸出的時間，這樣能將 latency 降為 0。然而，在執行 02_SYN 時發生了 time violated，這表明在運算過程中我花費了太多時間，需要對其進行優化！

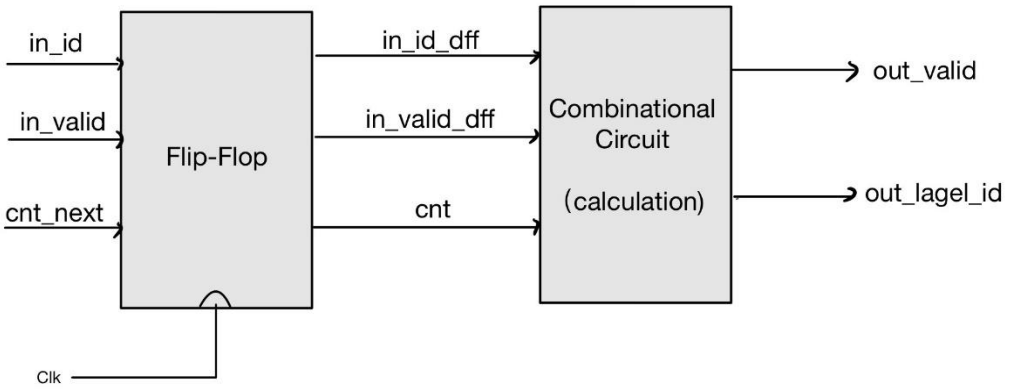
在對每個值（char）進行權重運算時，以及將值丟入總和時，我都使用了 in_valid_dff 擋起來。然而，這兩者僅需選擇一個即可，因為它們是相互影響的。此外，我發現每次計算餘數都是將所有總和加起來後再計算餘數，但我可以僅將每次的 in_data 進行 mod10 運算，然後將其值丟入總和中進行加總。這種方法可以大大減少硬體每次進行 mod10 運算的計算量，最終成功將總延遲變為 0 個 cycle，並使 Time report 的 result 為 Met。

```
always_comb begin
    if(in_valid_dff)begin
        if(cnt == 0) char = (in_id_dff/10) + (in_id_dff_rem)*9;
        else char = in_id_dff * (9-cnt);
    end
    else char = 0;
end

assign sum_rem = sum - (sum/10)*10;
always_comb begin
    if (( 10-sum_rem == in_id_dff) && cnt == 9 ) out_legal_id_comb = 1;
    else if((sum_rem ==0 &&  0 == in_id_dff) && cnt == 9 ) out_legal_id_comb = 1;
    else out_legal_id_comb = 0;
end
//sum
always_ff @( posedge clk or negedge rst_n ) begin
    if(!rst_n) sum<= 0;
    else if (in_valid_dff)  sum <= sum + char;
    else sum <= 0;
end
```



```
always_comb begin
    if(cnt == 0) char = (in_id_dff/10) + (in_id_dff_rem)*9;
    else char = in_id_dff * (9-cnt);
    sum_next = (sum + char)%10 ;
end

always_comb begin
    if (( 10-sum == in_id_dff) && cnt == 9 ) out_legal_id = 1;
    else if((sum ==0 && in_id_dff == 0) && cnt == 9 ) out_legal_id = 1;
    else out_legal_id = 0;
end
//sum
always_ff @( posedge clk or negedge rst_n ) begin
    if(!rst_n) sum<= 0;
    else if (in_valid_dff)  sum <= sum_next ;
    else sum <= 0;
end
```

2.架構圖



3.心得報告

這次的作業其實只要順順的寫，就都能很容易就能把硬體刻出來。只不過硬體並不是軟體，我們還需要考慮面積以及速度等優化的層面。

這次作業中有個很玄的地方，就是我的 sum_next 的 bit 數不需要那麼大，在 02_SYN 合成時，也有收到相關的提示，但我將 bit size 調製適當的大小時，面積反而卻變大，這是讓我不解的地方。

此外這次我更特別針對 coding style 方面的改善，以前都是僅使用 Verdi 以及跑 01_RTL,02_SYN,03_GATE，查看有甚麼問題，這次有開始使用 Spyglass 來檢查自己的程式碼，不僅把 error 消除掉，更是把所有跳出 warning 的問題所改善，我相信這樣的實踐，能讓我的 Verilog 寫得更加完整。

另外我接著有試著使用 Table 的方式重新寫一遍，發現面積更可以到 4000 左右，不過那已經超過 demo1 時間了，所以會發現用不同邏輯的運算，會使面積差距甚大，並且每套邏輯縮面積都有極限，我是將原先 7900->7251，認為已經沒甚麼地方能再做修改，若有也不會差太多。

```
Library(s) Used:

    slow (File: /usr/cad/umc018/Synthesis/slow.db)

Number of ports:                           11
Number of nets:                           367
Number of cells:                          347
Number of combinational cells:            332
Number of sequential cells:                15
Number of macros/black boxes:               0
Number of buf/inv:                         70
Number of references:                      72

Combinational area:               5801.241649
Buf/Inv area:                      748.440022
Noncombinational area:            1450.310402
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  7251.552051
Total area:                 undefined
```

```
data required time                                 3.74
data arrival time                                 -3.74
-----------------------------------------------------------
slack (MET)                                       0.00
```

面積:7251

Slack:0

Total latency:0

```
        Congratulations!

You have passed all patterns!

Total latency:           0 cycles
```