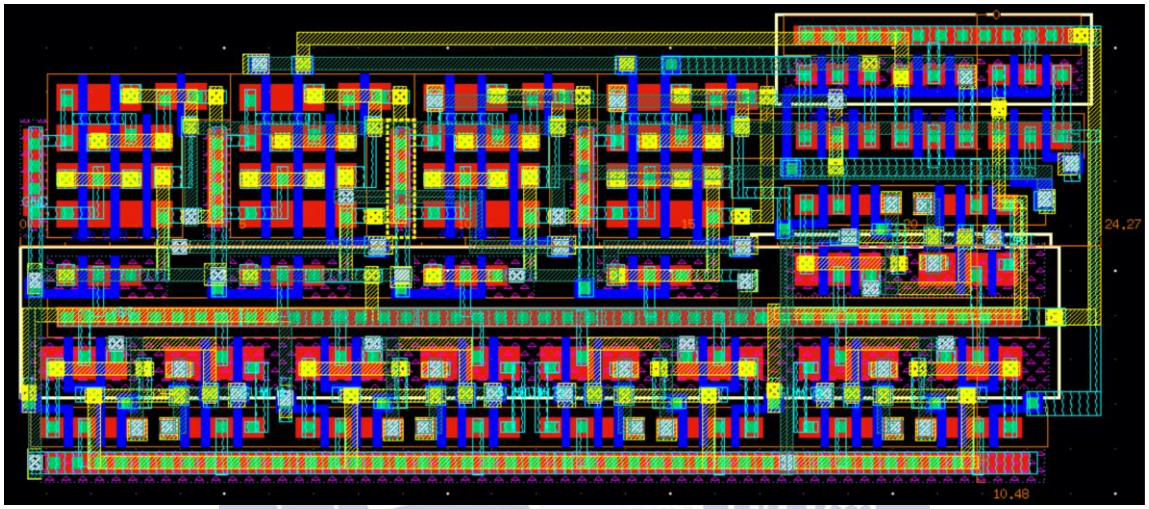
2023 NYCU EE VLSI Lab Report

Lab03 4-Bit Full Adder + DFF

Student ID: 110511277 Name: 蔡東宏 Date: 2023/11/17

I. Layout result

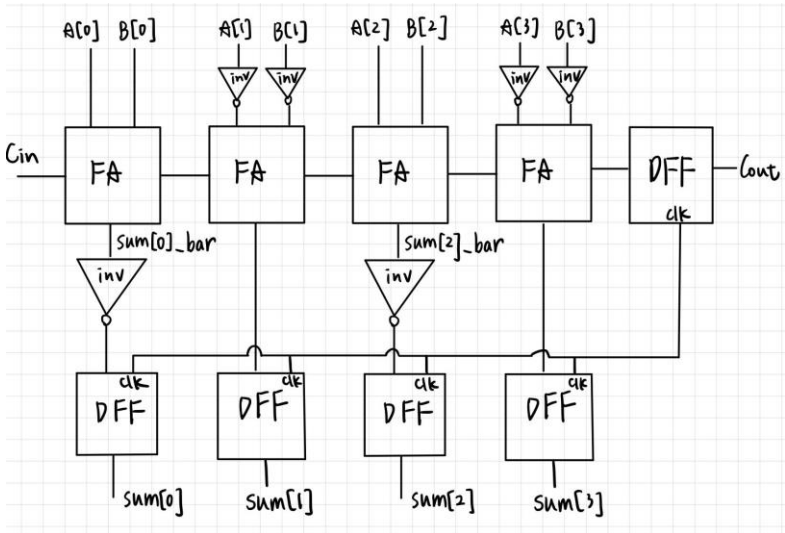
1. Layout picture with ruler



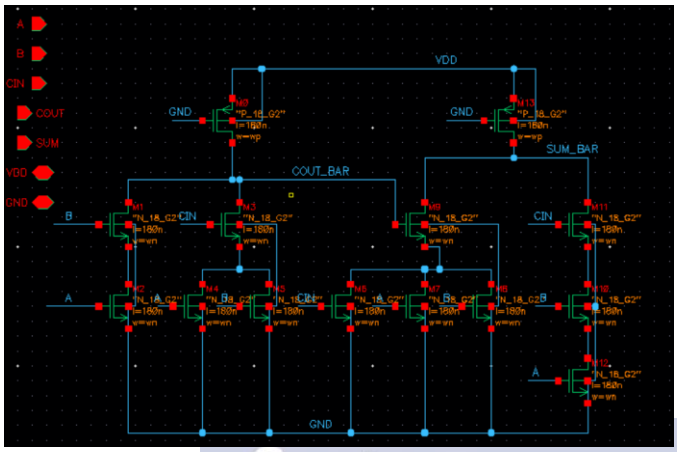
2. Design concept

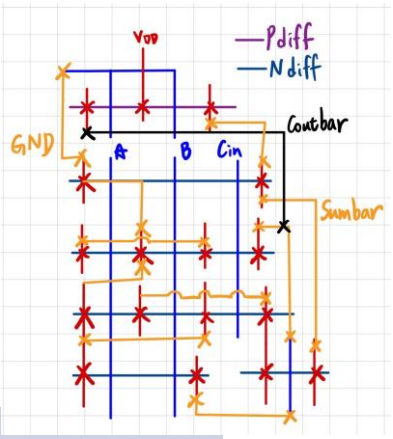
(1) Circuit Schematic

FA_4bit schematic:

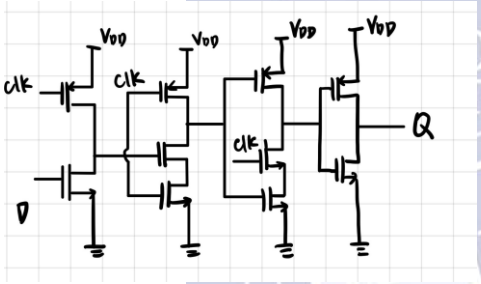


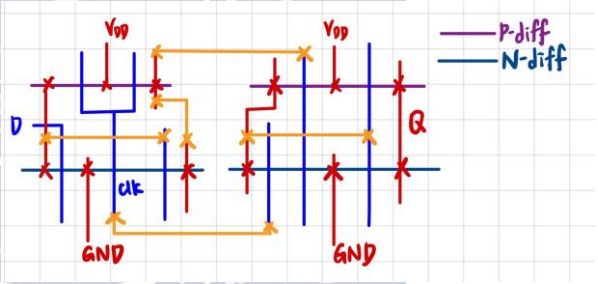
FA_1bit schematic and stick diagram:



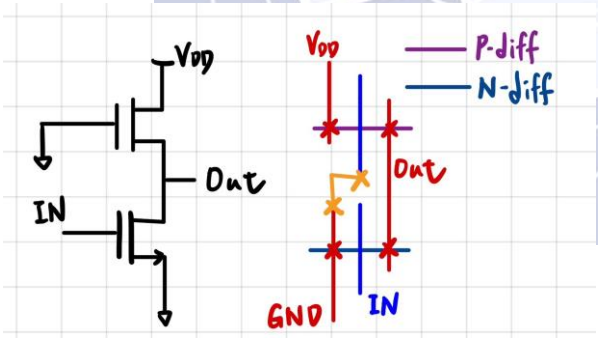


DFF schematic and stick diagram:



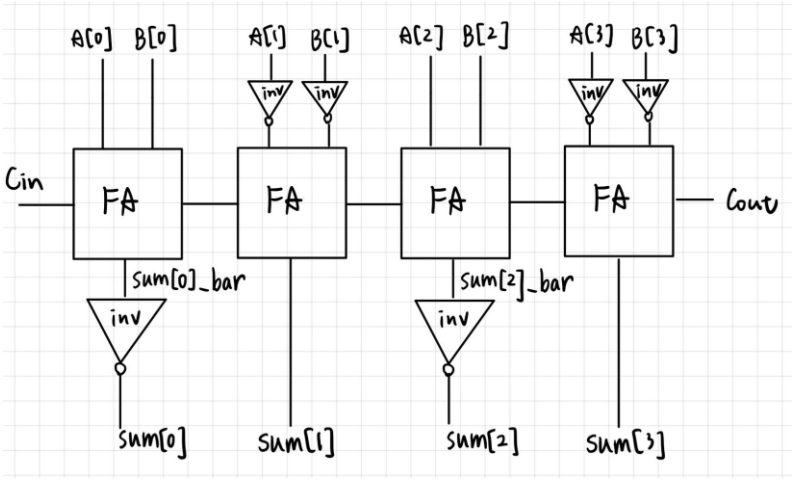


Inverter schematic and stick diagram:



(2) Summary of structure (number of transistor / logic gate is used)

我這次的 4bit full-adder 架構是用 ripple carry adder 的架構，我的 FA-1bit 是用 CCMOS 的架構，並將它改成 pseudo nmos，減少面積的同時並減少 delay，除此之外，我還將 FA 最後輸出端的 inverter 拔掉，如此一來，整體的架構能夠改成如下圖一，FA 內部總共減少了 4*2 個 inverter，但需在 A[1]、B[1]、A[3]、B[3]、SUM[0]、SUM[2]端多加 inverter，整體而言，少了兩顆 inverter，且我的 inverter 也有改成 pseudo nmos 的版本，用來減少 delay。所以我的 FA-1bit 用了 14 個 transistor，DFF 用了 10 個 transistor，inverter 用了 2 個 transistor，總共用了 14*4+5*10+6*2=118 個 transistor。

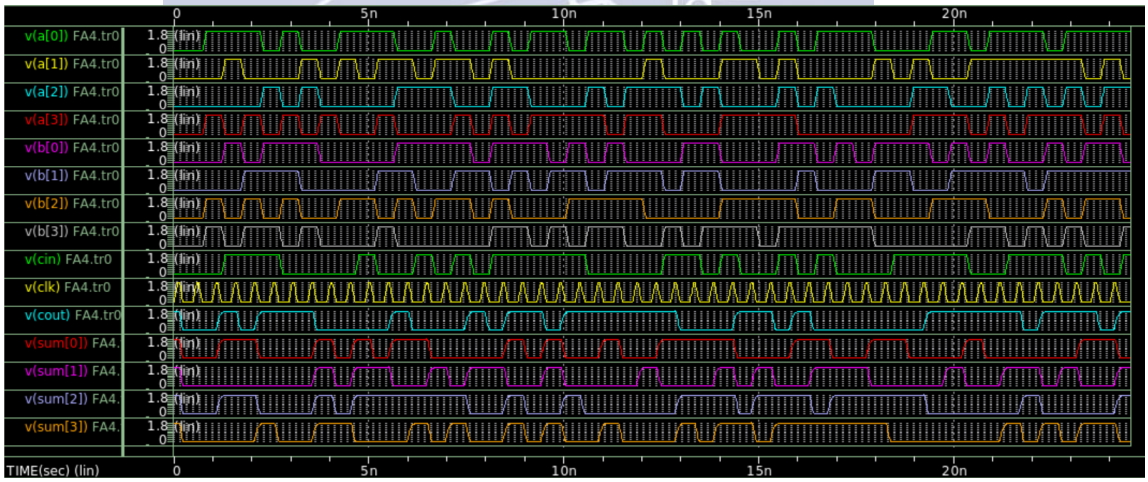


(圖一)

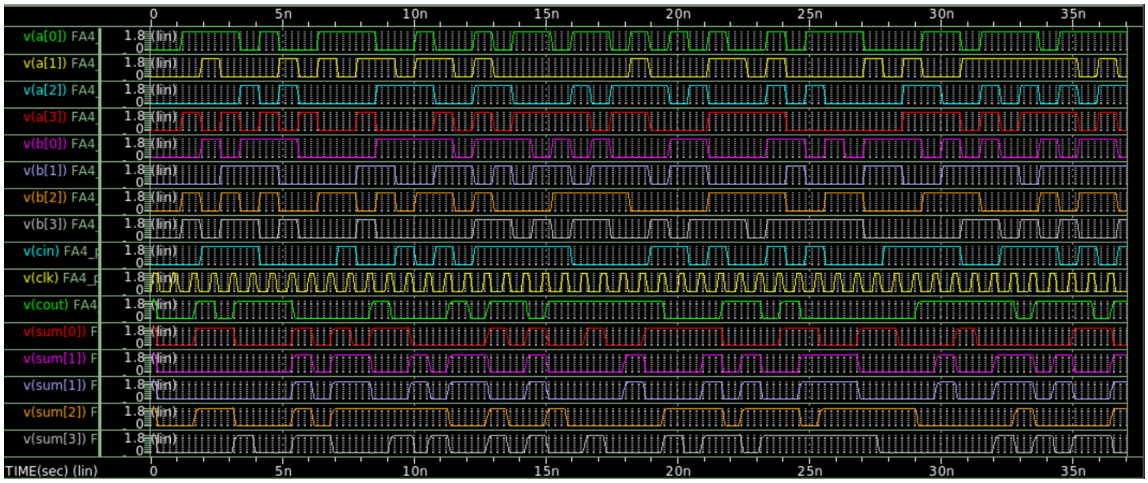
II. Simulation result

1. Output waveform

(1) Pre-sim (Output waveform)



(2) Post-sim (Output waveform)



(3) Performance list (TT case under worst case input pattern)

Maximum operation frequency	Pre-sim:2.041GHz (period=0.49ns)
	Post-sim:1.351GHz (period=0.74ns)
Average power	Pre-sim:2.264mW
	Post-sim:2.139mW
Layout area	24.27 * 10.48 =254.35(um^2)
4-bit full adder structure	Ripple carry adder

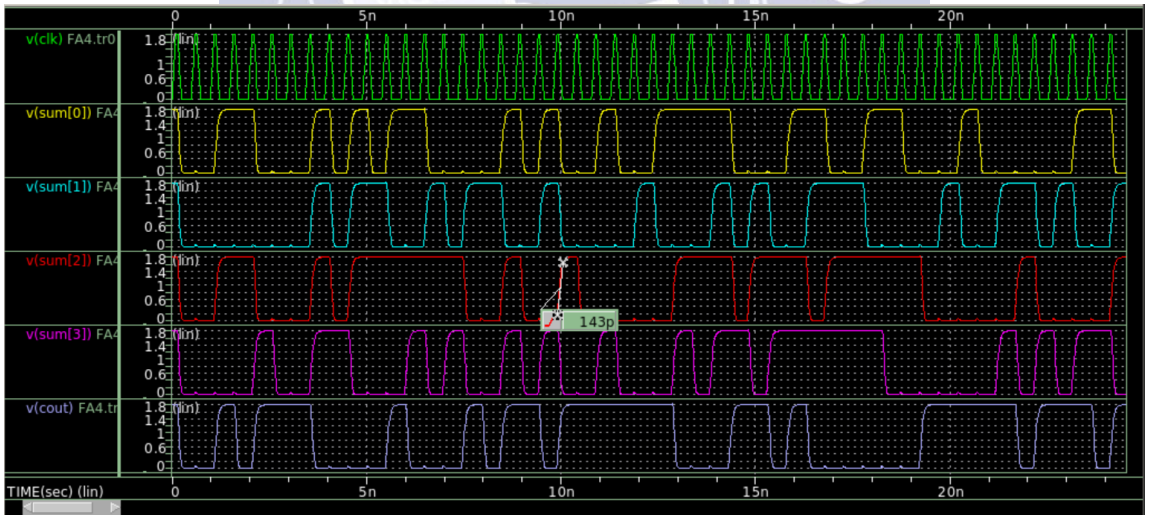
Table 1: Output Simulation Summary

	Spec.	Pre-sim	Post-sim
Worst Rise Time	< 0.8ns	143ps	164ps
Worst Fall time	< 0.8ns	87.6ps	103ps
Worst Propagation Delay	N/A	427ps	687ps
Average Power	N/A	2.624mW	2.138mW

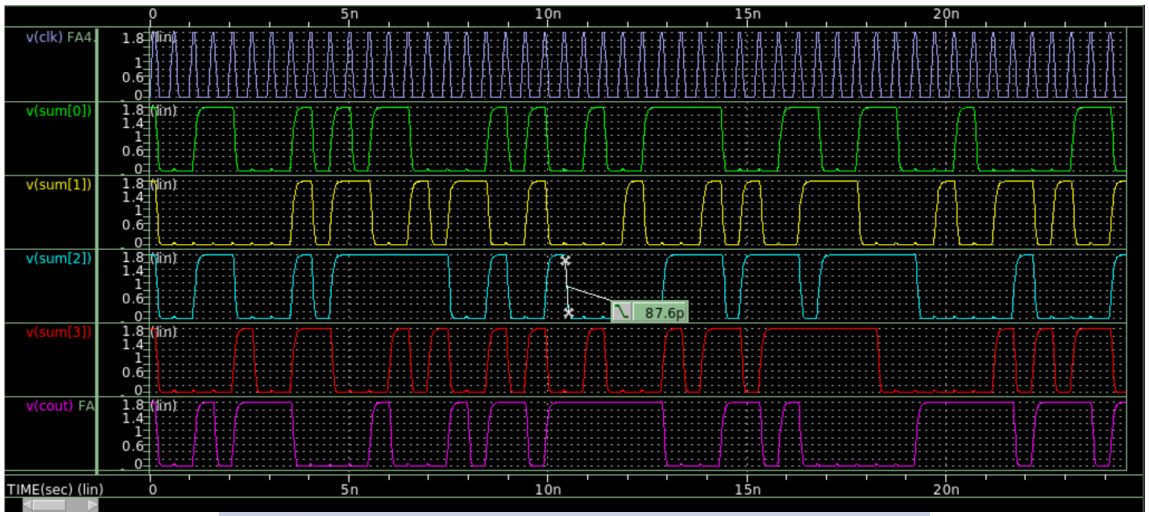
(*paste measurement result of HSPICE, i.e. .mt0)

Pre-sim:

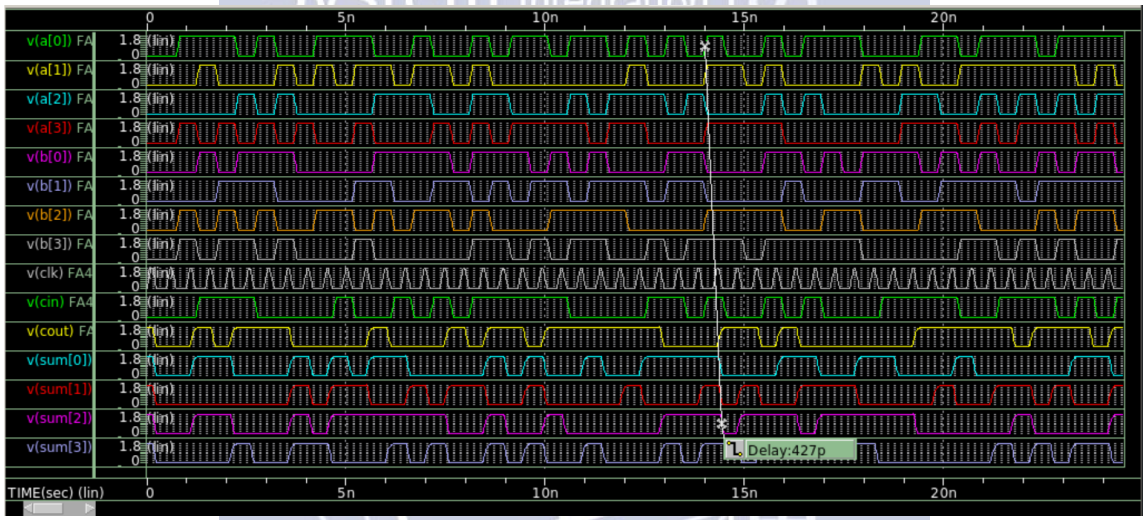
Worst rising time:



Worst falling time:



Worst propagation delay:

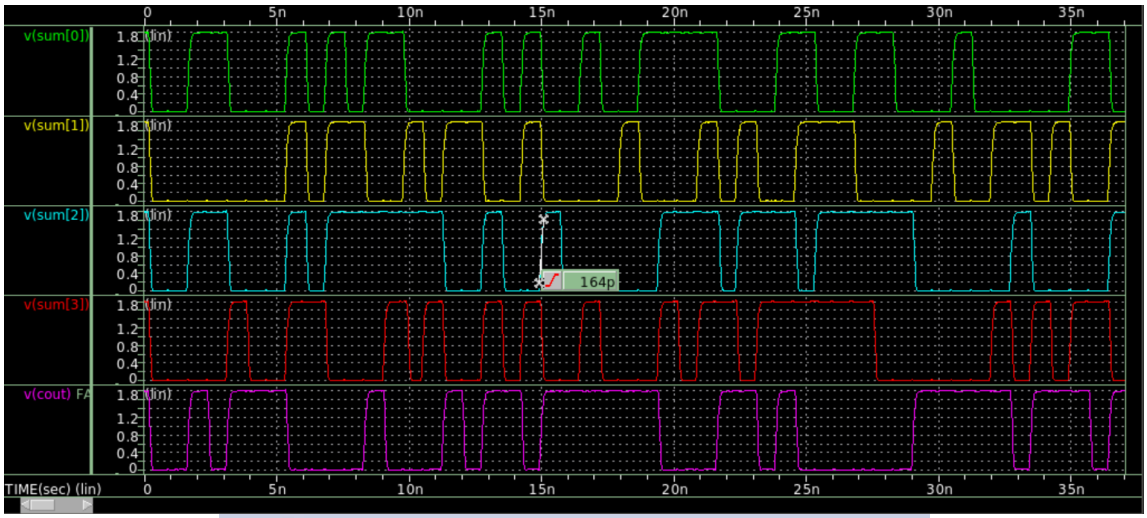


Avg_power:

```
$DATA1 SOURCE='HSPICE' VERSION='Q-2020.03-SP2-2 linux64' PARAM_COUNT=0
.TITLE '****  lab03: 4-bit fa  ***'
 iavg             pavg             temper          alter#
 -1.258e-03        2.264e-03         25.0000        1
```

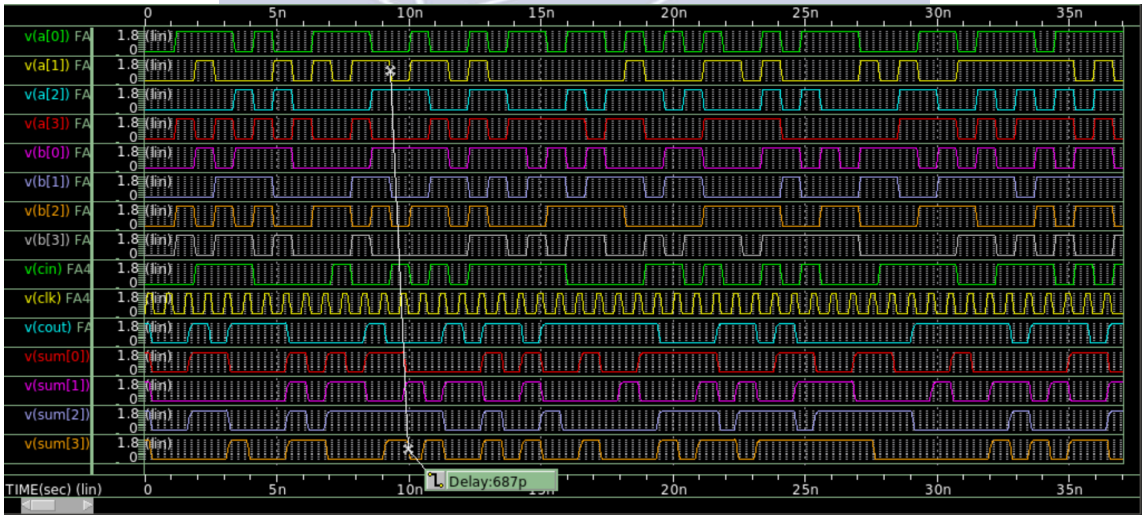
Pre-sim:

Worst rising time:



Worst falling time:

Worst propagation delay:

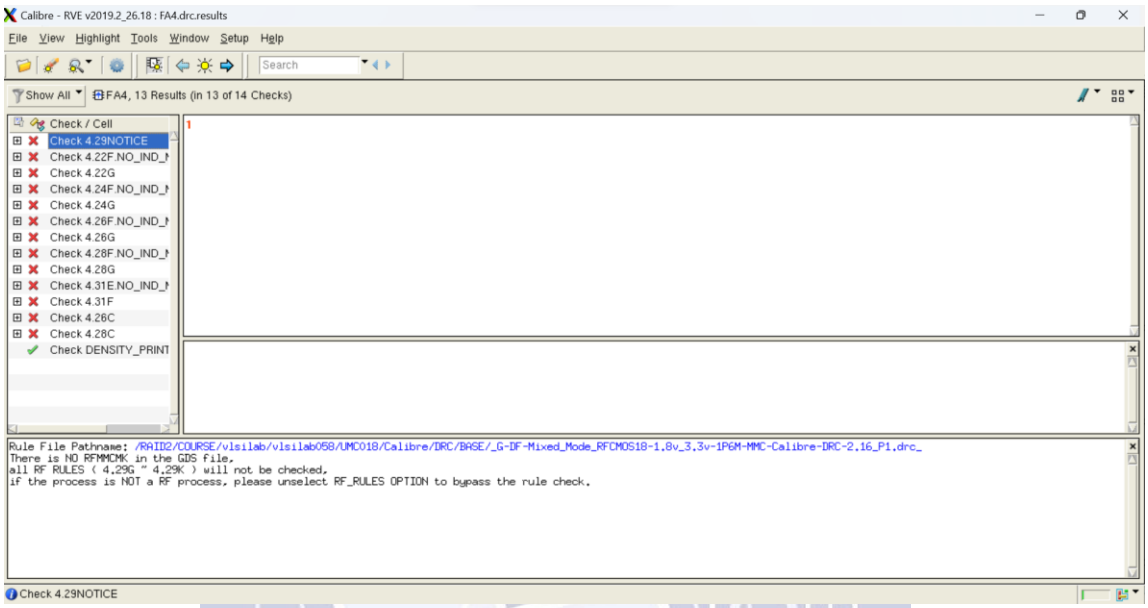


Avg_power:

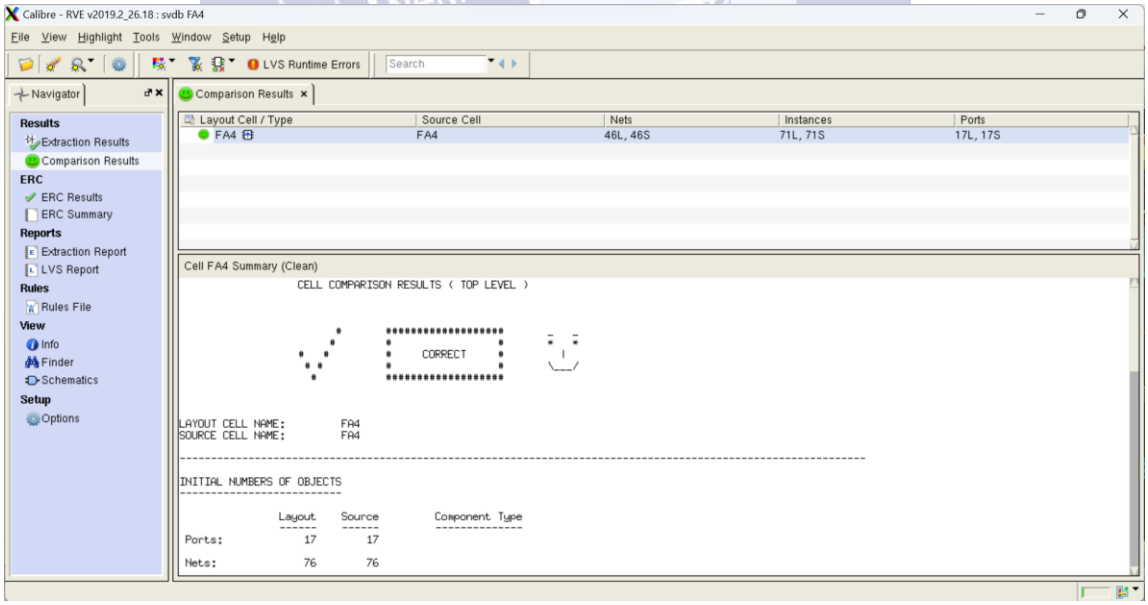
```
$DATA1 SOURCE='HSPICE' VERSION='Q-2020.03-SP2-2 linux64' PARAM_COUNT=0
.TITLE '****  lab03: 4-bit fa  ***'
 iavg             pavg             temper          alter#
 -1.189e-03        2.139e-03         25.0000        1
```

III. Verification result

1. DRC



2. LVS



IV. Discussion

1. Compare Full Custom to Cell-based design and list their pros and cons.

Full custom 的設計是從頭到尾都是設計者自行設計，從最基本的單位 transistor 開始設計，所以 full custom 能夠達到更高的效能，而 cell-based design 是由已經兜好的電路往上設計，像是能夠直接取用 cell library 的 inverter、DFF、MUX 等邏輯電路。

Full custom 的優點: 高性能、面積較小(因為每個 transistor 的 size 都能夠更自由的調整)

Full custom 的缺點: 消耗更多人力、製作的時間較久。

Cell-based 的優點: 節省人力以及製作時間、設計上更彈性(因為有已經寫好的 cell library)

Cell-based 的缺點: 面積較大、消耗更多功率。

類比電路的設計通常是用 Full custom 的設計，為了追求更高的性能以及 performance，而數位電路的設計通常是用 Cell-based 的設計，因為數位電路只有 high 和 low。

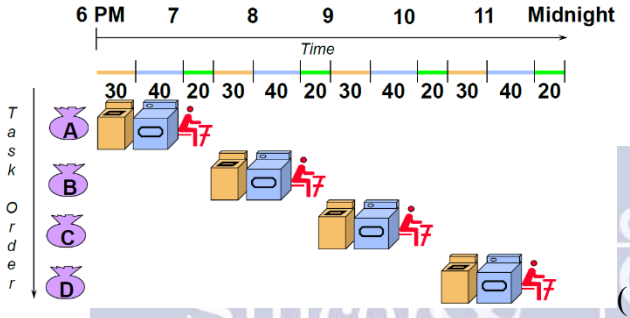
2. How to eliminate performance variation between Post-sim and Pre-sim?

Pre-sim 和 post-sim 的差異最主要來自於寄生的 RCC，所以要減少差異的方法就是想辦法減少寄生的電阻和電容，電阻正比於長度，反比於寬度，而電容正比於長度以及寬度，若要同時減少電阻和電容的話，需盡可能地讓走線越短越好，但如果將兩條導線靠得太近的話，容易造成金屬與金屬間的寄生電容，而造成 delay 變大，若將導線變寬的話，雖然能有效降低寄生電阻值，但也會讓寄生電容值變大，導致 RC delay 沒有變好。

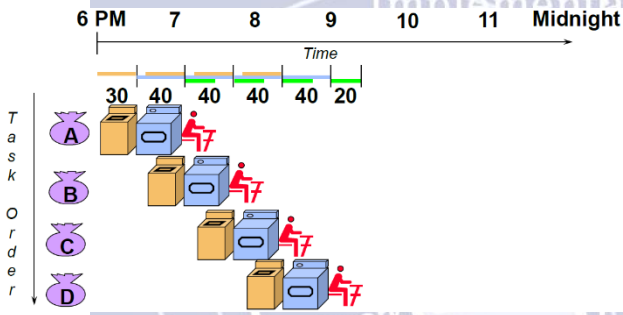
共用 diffusion 也是一個方法去減少 pre-sim 和 post-sim 之間的差異，因為共用 diffusion 可以降低寄生電容值，讓某一顆 transistor 的 source 或 drain 與另一顆的 source 或 drain 共用，原本有兩倍的 Csb 或 Cdb 能夠變成只有一倍，因為在 pre-sim 時沒有考慮到共用的問題，所以共用 diffusion 也可以減少電容，導致 RC delay 變好。

3. **Please discuss what has influence on speed while pipeline regs are placed at different location.**

加 pipeline 是為了減少一個 cycle 內 combination 的運算時間，在 combination 的某個節點加入 DFF，如此一來，當我第一筆的資料完成第一級運算時，下一筆資料可以進來第一級運算，不用等第一筆資料全部算完再進來，因此能夠提升電路運行速度。



(沒有加 pipeline 示意圖)



(有加 pipeline 示意圖)

4. **Discuss the difference between different adder structures and their own advantages/disadvantages. Talk about your final choice, including how and why you design in this way, and the optimization you made.**

選擇 adder 的架構是這次 performance 的關鍵，常見的架構有 ripple carry adder、carry lookahead adder、carry selector adder，ripple carry adder 是將某一級的 cout 傳到下一級的 cin，ripple carry adder 的硬體容易實現且需要的面積較小，但他的缺點是若有很多級的 adder，它的 critical path 是從第一級 cin 到最後一級 cout 或 sum，critical path 太長會造成 delay 太長，carry lookahead adder 是透過 input 獨立生成每一級的 cin，它的優點是它的 critical path 不會太長，能夠減少整體電路的 delay，但它的電路相對複雜，需要用更多硬體去實現，而 carry selector adder 是提早算出 cin 為 0 或 cin 為 1 的結果，等到上一級的 cout 算完之後，再用 mux 去選擇正確的結果，此架構在速度與面積取得相對的平衡，它的速度比 ripple carry adder 快，比 carry lookahead adder 慢，但面積比 ripple carry adder 大，比 carry lookahead adder 小。最後，我決定使用 ripple carry

adder，因為考慮到這次的 Lab 只有 4-bit，用 carry lookahead adder 以及 carry selector adder 不會快非常多，但考量到 delay 以及 layout 的複雜性，我還是選擇使用 ripple carry adder，並且我的 1bit_adder 是使用 CCMOS 組成，並使用 pseudo nmos 來減少 delay。

5. Summary + Learning

這次的 lab 需要自己設計整個 4bit full-adder，從 pre-sim 開始就要花時間去想架構，還有花時間在調整電晶體的參數(W)，我 pre-sim 花了滿多時間在想辦法減少我的 clock period 並減少我的電晶體數量，讓我的 layout 面積可以小一點，在畫 layout 的時候也想了滿多種擺法，因為這次沒有規定 pmos 要在上面，nmos 要在下面，所以我花了很多時間擺放 layout 讓我能夠盡可能減少浪費的空間，最後再接線的時候也是想辦法讓線能夠越短越好，減少 delay 讓我的 performance 可以更好。