1. A 4-bit ripple adder as ahown at Fig.1 is designed with Fully Complementary Static Logic Gate for FA. Input signals are A[3:0], B[3:0] and Cin which are provided by a unit size inverter. Outputs are Sum[3:0] with loading of 5 unit size inverters connected in parallelism (FO5). You shall provide SPICE simulation results of timing and power waveforms.

(1) Try your best to design the fastest adder. First, show your block diagrams in terms of the 1-bit Full-Adder(FA). Second, show the circuit schematic of each block. Use logic effort concepts (you do not have to write down the procedure) to design transistor widths. **Describe your design concept.** (40%)

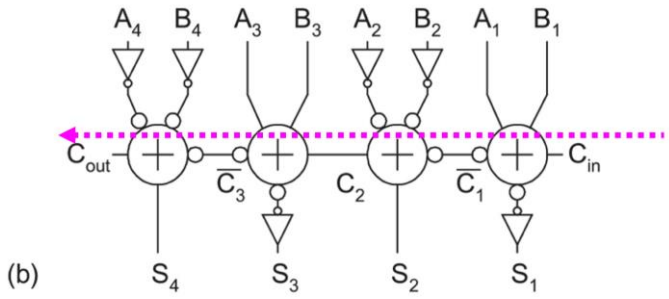
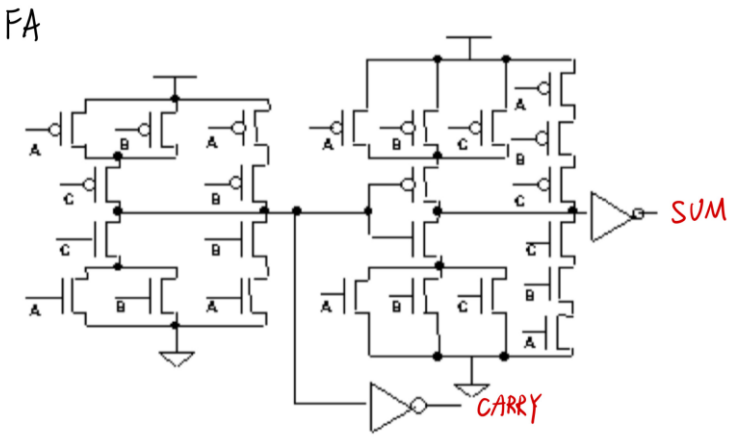
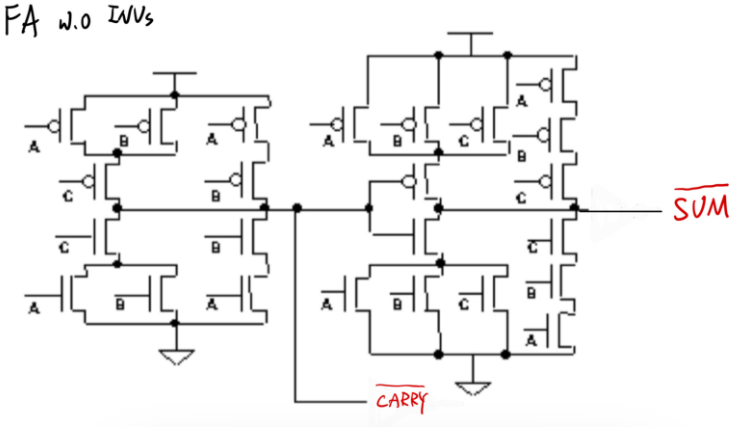



FIG 10.12 4-bit carry-ripple adder

圖一



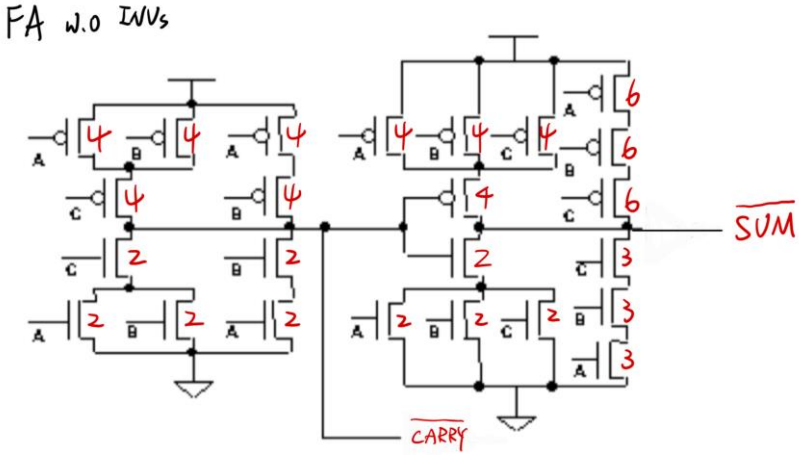

圖二



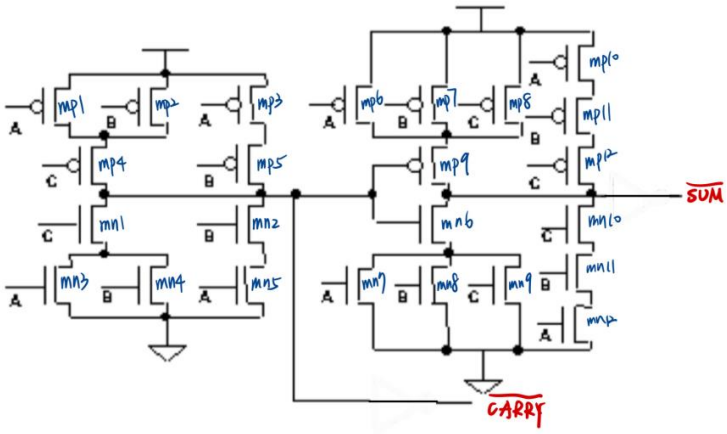

圖三

題目要求為設計盡可能快的 adder，因此參考講義 4-bit ripple adder 的架構後，選擇了圖一的這個架構。原因是在 critical path 上雖然每個 CARRY 的輸出是反向的，但是可以透過串接四級反向的 full adder 來改進原先使用圖二這種 full adder 在 critical path 上會多四個 inverter 的 delay。因此我最後使用的 full adder 架構如圖三所示。此外，雖然在圖一上的 A2,B2,A4,B4,S1,S3 上都有多加上 inverter 以確保邏輯的正確，但是這並不會使我的電路速度變慢，因為其不是在我最關心的 critical path 上。

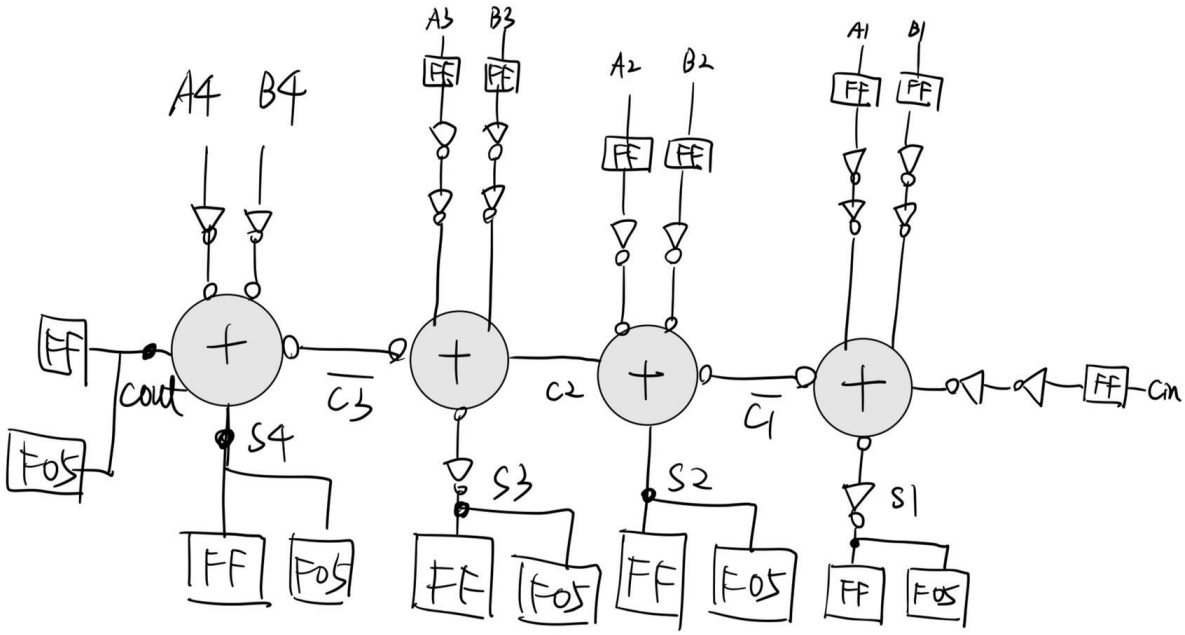
接下來，對 1bit FA 進行 sizing(nfin_p : nfin_n = 2 : 1):




上圖可以看成產生 CARRY 的一個 cluster，產生 sum 的為一個 cluster。此外，CARRY 的反向又會接到下一級的 full adder，因此可以視為在中間這點有了一個 branch。此外在 On path 上更加考慮了 input unit size inverter 與 output 的 FO5 還有多出來的 input 與 output 的 DFF 一起納入 logic effort 的計算。最後計算出來的結果如下:




| | mp1~mp9 | mn1~mn9 | mp10~mp12 | mn10~mn12 |
|---|---|---|---|---|
| FA1 | nfin=5 | nfin=2 | nfin=7 | nfin=3 |
| FA2 | nfin=6 | nfin=3 | nfin=9 | nfin=5 |
| FA3 | nfin=7 | nfin=3 | nfin=11 | nfin=5 |
| FA4 | nfin=8 | nfin=4 | nfin=12 | nfin=6 |

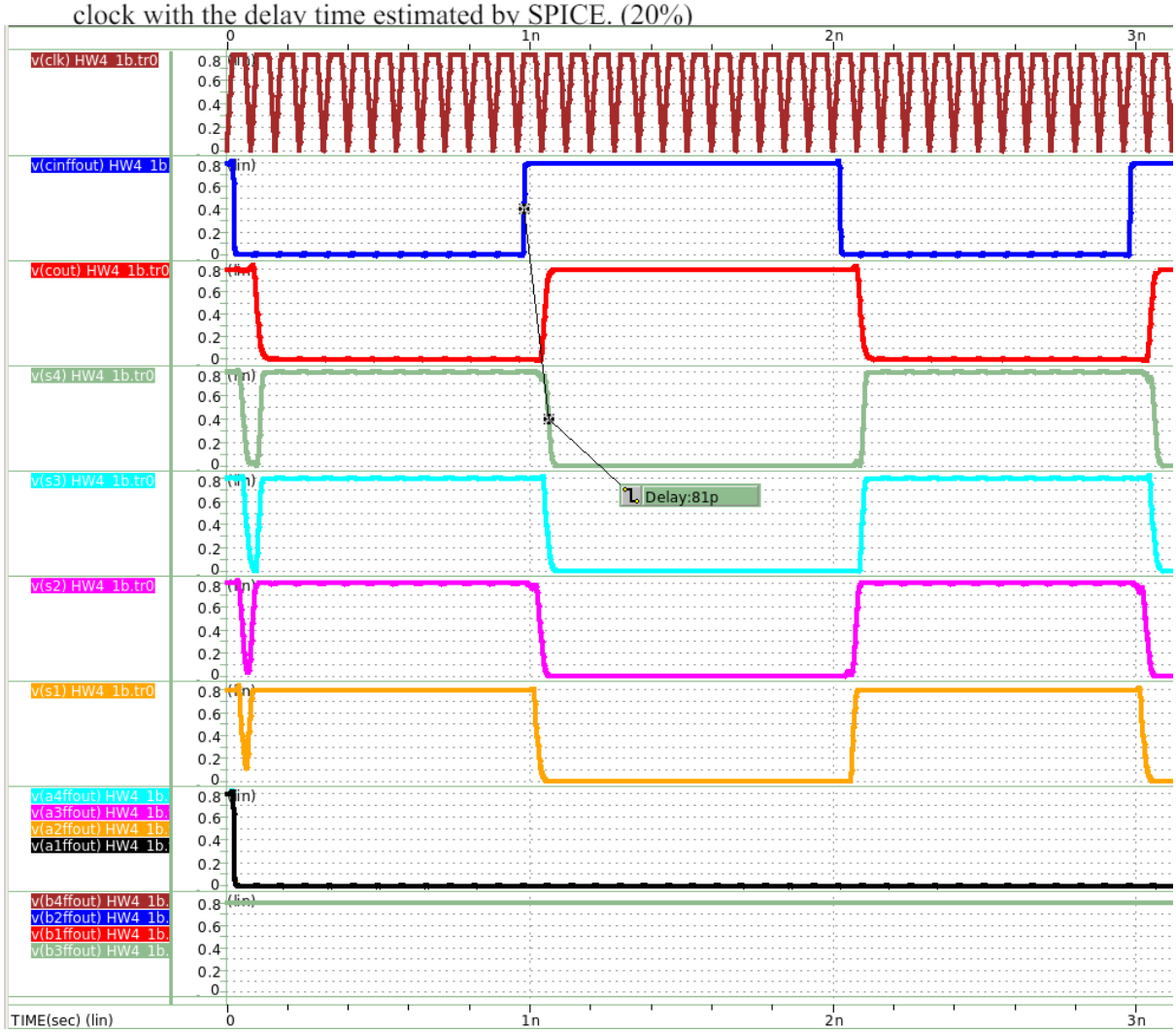



上圖為最終的 4bit carry ripple adder 的設計。在 input 端都會先過一個 positive edge trigger 的 DFF，然後使用 unit size inverter 提供 input pattern 給個別的 Full adder。而這四個 full adder 單元中，輸出的都是 cout bar 與 sum bar，由此來減少 critical path 中的 inverter propagation delay。也因此，需要將 S1 與 S3 加上反向，才能夠確保其 function 是正確的。此外，如同題目所說，在所有的輸出節點 S1~S4 與最後的 cout 端，我也都掛上了 DFF 與 FO5 的 delay，而其中 FO5 是由五個 unit size 的 inverter 並聯組成，來模擬外部的附載。而量測該電路的 max propagation delay 可以透過觀測 critical path 中 cin 到 S4 為該 carry ripple adder 的 critical path。此外，對於該電路，我們使用 worst case pattern: 000011110 to 000011111 (A[3:0]@B[3:0]@Cin).因為每個 sum bit 都需要重新充放電，有 carry 進位的需求，來量測這個電路的所有 input pattern arc 中最長的 propagation delay 為何，如此可以用以界定若是要在一個 clock cycle 中運算完這個 combinational cell 最悲觀需要多少時間。

以下是我透過上面提及的 logic effort 計算出來的 sizing:

| | mp1~mp9 | mn1~mn9 | mp10~mp12 | mn10~mn12 |
|---|---|---|---|---|
| FA1 | nfin=5 | nfin=2 | nfin=7 | nfin=3 |
| FA2 | nfin=6 | nfin=3 | nfin=9 | nfin=5 |
| FA3 | nfin=7 | nfin=3 | nfin=11 | nfin=5 |
| FA4 | nfin=8 | nfin=4 | nfin=12 | nfin=6 |

(2) Based on the design of (1), run SPICE to find the the propagation delay time (with pattern from 000011110 to 000011111 (A[3:0]@B[3:0]@Cin). Determine the maximum propagation of a clock with the delay time estimated by SPICE. (20%)




上面這個波型有幾個可以討論的點:

(a)

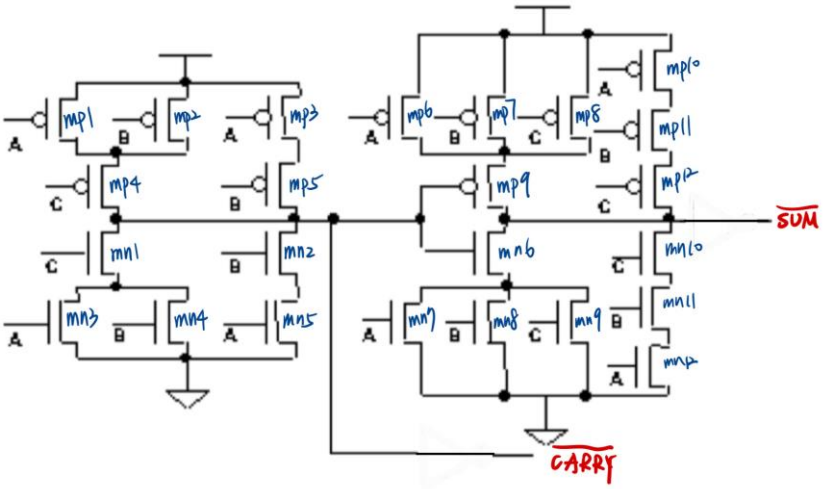
可以觀察到在最一開始大約 0.1ns 附近，可以看到有一些奇怪的 glitch，推測來自 flip flop 本身內部的 internal node 沒有被初始化，而導致的 glitch，當 clk posedge 時，便都變成題目所規定的 input pattern 的輸出。

(b)

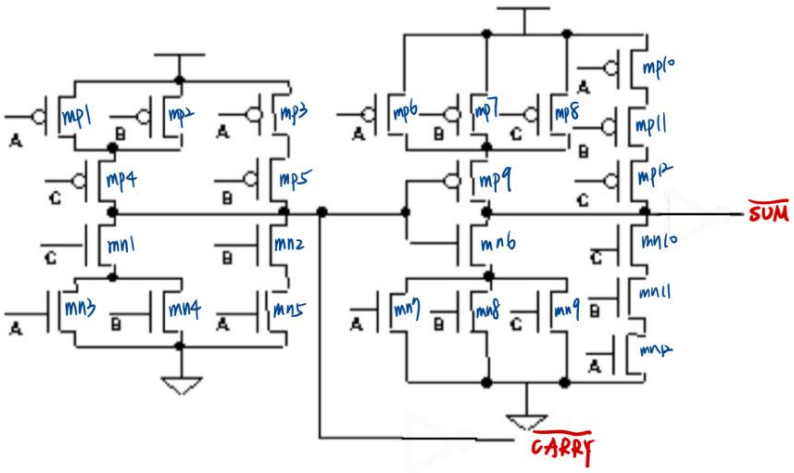
當時間介於 0~1ns 之間時，輸入為 000011110 (A[3:0]@B[3:0]@Cin)，可以看到輸出的 cout 及 sum 為 01111(cout@S[3:0])，而當時間介於 1~2ns 之間時，輸入變為 000011111 (A[3:0]@B[3:0]@Cin)，可以看到輸出的 cout 及 sum 為 10000(cout@S[3:0])，與預期相同。

(c)

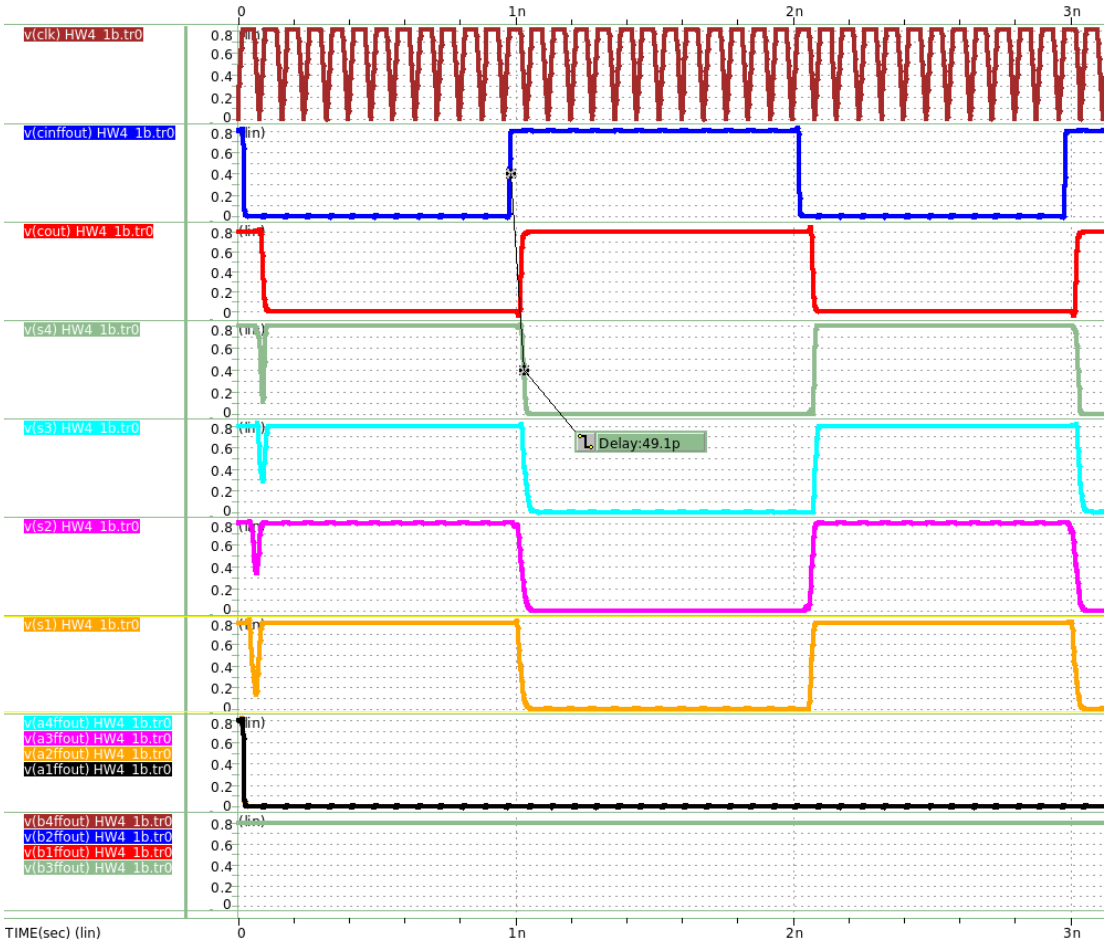
透過上面討論 critical path 的結論，透過量測 cin 到 s4，可以量測到該 carry ripple adder 的 max propagation delay 為 81ps。但是這僅僅是透過 logic effort 計算出來的，我想對於實際電路來說，還有許多條件我們沒有考慮到，因此我們可以再進一步來優化。




經過不斷嘗試與 sizing 後，我發現，透過調小上圖右半邊電路的 nfin(非 critical path)，並且加大 critical path 上的電晶體，可以有效的減少 propagation delay。此外，因為這次設計的 metric 為 performance，因此面積與功耗並不在這次設計的優化方向。因此透過不斷嘗試(理論上這邊應該要寫個 automation tool 來 optimize，但時間太短了)，最後我試出來一個還不錯的結果，sizing 結果如下圖:




| | mp1~mp5 | mn1~mn5 | mp6~mp9 | mn6~mn9 | mp10~mp12 | mn10~mn12 |
|---|---|---|---|---|---|---|
| FA1 | nfin=10 | nfin=8 | nfin=2 | nfin=1 | nfin=3 | nfin=2 |
| FA2 | nfin=20 | nfin=12 | nfin=4 | nfin=2 | nfin=5 | nfin=3 |
| FA3 | nfin=30 | nfin=16 | nfin=5 | nfin=3 | nfin=7 | nfin=4 |
| FA4 | nfin=40 | nfin=24 | nfin=12 | nfin=6 | nfin=14 | nfin=6 |




可以看到透過上面所提到的 sizing 邏輯，可以看到功能仍然正常，並且 propagation delay 減少了許多。

| | Logic effort version | Optimization version |
|---|---|---|
| Max propagation delay | 81ps | 49.1ps |

因此，可以估計 max frequency 為最多是 1/49.1ps=20.3GHz
(但理論上還有 tpcq 及 setup time 要考慮進去，所以可以大約估計 15GHz)
在這之中，因為共用了四次不同大小的 full adder，因此若是可以將某幾個 transistor 要用幾個 fin 使用類似參數的方式傳入 subckt 中，那會使得 spice deck 更加的優雅且易於更改。因此上網查詢 spice 的語法後，發現真的可以有這樣的傳參數的方法:

```
.subckt FA A B C CB SB pfin1=4 pfin2=4 pfin3=6 nfin1=2 nfin2=3 nfin3=3
*   subckt description
.ends
```

此外，不得不稱讚業界做 ip 抑或是 synthesis tool 的厲害，透過程式及相關的最佳化演算法，去透過 either 是 sizing 或是 place & route 的方法來進行多 metric 的最佳化。

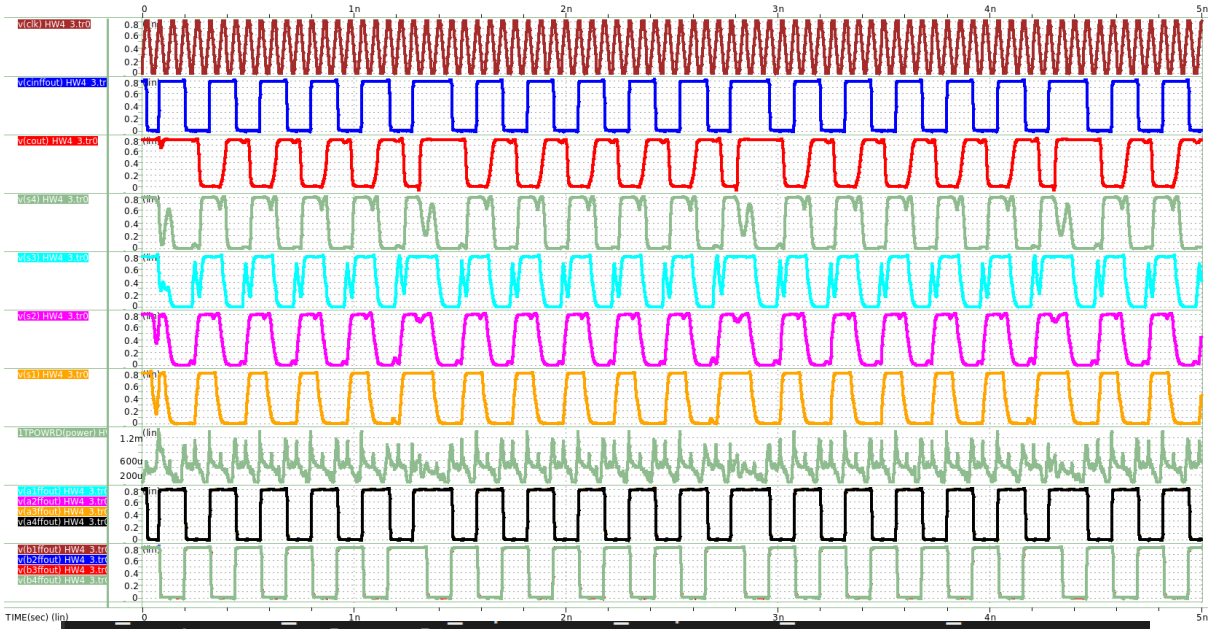
(3) Run SPICE to get the average, peak and leakage power dissipation and energy/bit, respectively of this adder with loading (FO4) when working at the maximum working frequency. (20%)

(When running SPICE simulation in (3), use the follwing input patterns)

| Average and peak power | Leakage power |
|---|---|
| A[3:0] is from 0000 to 1111 | A[3:0] is 0000 |
| B[3:0] is from 1111 to 0000 | B[3:0] is 1111 |
| Cin is 0/1 by turns | Cin is 0 |

/*You should provide the SPICE input description, timing and power waveforms*/

我這邊使用 clock period 為 60ps，雖然我的 carry ripple adder 的 max propagation delay 的 propagation delay 為 49.1ps，但是為了不要有 setup time fail 及希望看到波型不要那麼多 glitch(因為 clk 變動太快，電容還沒充滿電就又放電，還沒放玩店就又充電的情況)，我將 clk period 調整為 60ps。但是又規定 rising time 及 falling time 為 20ps，所以基本上 clk 很接近三角波了。



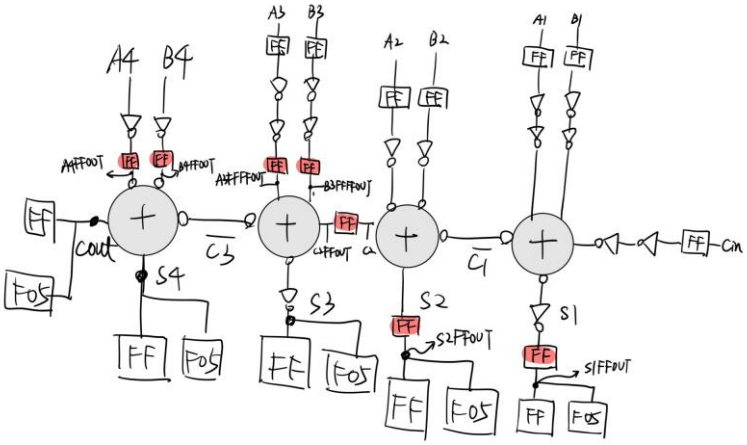
```
21:13 dic092@ee24[~/HW4]$ cat HW4_3.mt0
$DATA1 SOURCE='HSPICE' VERSION='Q-2020.03-SP2-2 linux64' PARAM_COUNT=0
.TITLE '*   simulator setting'
 avgpower         peakpower         temper           alter#
  4.183e-04         1.395e-03          25.0000         1
```

```
21:19 dic092@ee24[~/HW4]$ cat HW4_3_leakage.mt0
$DATA1 SOURCE='HSPICE' VERSION='Q-2020.03-SP2-2 linux64' PARAM_COUNT=0
.TITLE '*   simulator setting'
 leakagepower     temper           alter#
  2.244e-07          25.0000         1
```

而 energy/bit 則使用(avg_power * simulation time) / 4 = 4.183e-04W * 5ns / 4 = 522.8fJ

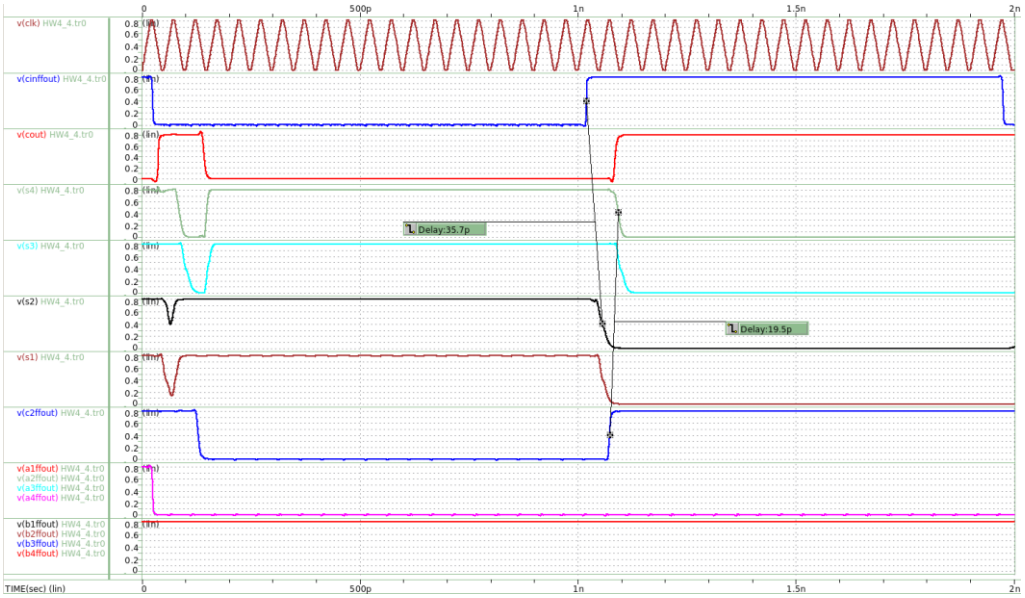
| Avg power | Peak power | Leakage power | Energy/bit |
|---|---|---|---|
| 4.183e-04W | 1.395e-03W | 2.244e-07W | 522.8fJ |

(4) Add one pipelining stage using the designed D register into the 4-bit ripple adder as shown at Fig.1(b). Run SPICE to find the the propagation delay time (with pattern from 000011110 to 000011111 (A[3:0]@B[3:0]@Cin) between pipelining stages to determine the maximum working frequency of the clock with the delay time estimated by SPICE. (20%)




(紅色是 pipeline register)

這邊將 pipeline 切在第二個 bit 與第三個 bit 中間，為求各級 pipeline 可以盡可能的平均，使 clk frequency 可以越大越好，亦即每即 pipeline stage 的 period 越小越好，這是我選擇切在中間的原因。此外，因為 S1 S2 會先算好，因此要先用個 FF 存起來，而至於 A3B3A4B4 也要等一下才進來，才可以拿到正確的 C2 值，所以也先用個 FF 存起來。待前一級算完後，得到正確的 C2 後，再將 A3B3A4B4 拿進來計算最終的結果。因此如何決定切完一級 pipeline 後的 clk period/frequency 要是多少呢?可以透過量測兩級 propagation delay 取最大的那個來決定。




以下是模擬結果(使用 clk period=50ps): 第一級的 propagation delay 是 35.7ps，而第二級 propagation delay 是 19.5ps。因此理論上，可以將 clk period 設為大約 40ps 附近。但是因為題目規定 clk tr tf 供需要 40ps，因此我選定 clock period=50ps。此外也可以發現第二級的 propagation delay 較小，原因是因為我沿用前面的 sizing 結果，FA3 與 FA4 都比較多 fin。因此可以再 sizing 微調一次，讓兩級 propagation delay 可以更平均。