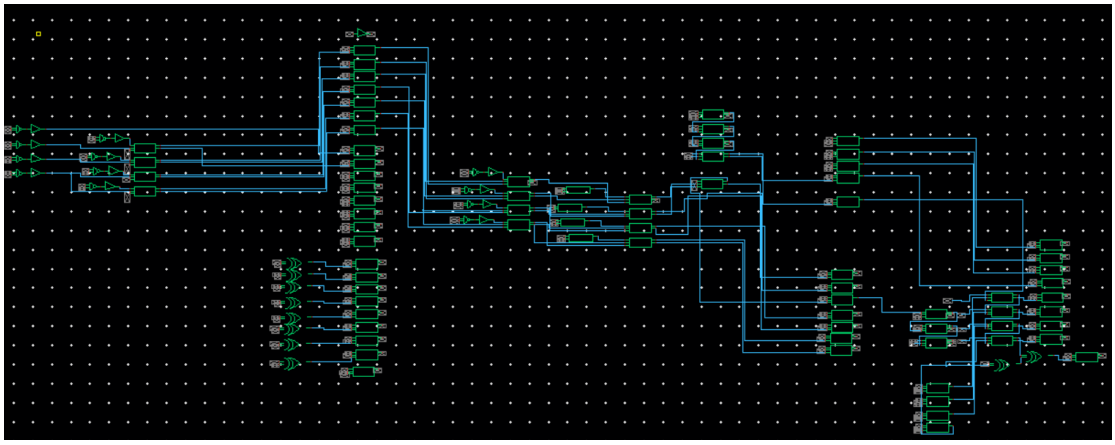
Schematic Design



Summary of Structure

Presim 架構：

我們的架構是利用 array multiplier 加上 sign extend，與其他輔助硬體來做 signed multiplier。

1. **array 乘法：**

   carry save adder + carry ripple adder，最後一個 partial product 加入 two's complement 機制。

2. **Array 加法：**

   用普通的 8bit ripple adder，而 OUT[8]是用兩個 XOR 產生

3. **Pipeline 切法：**

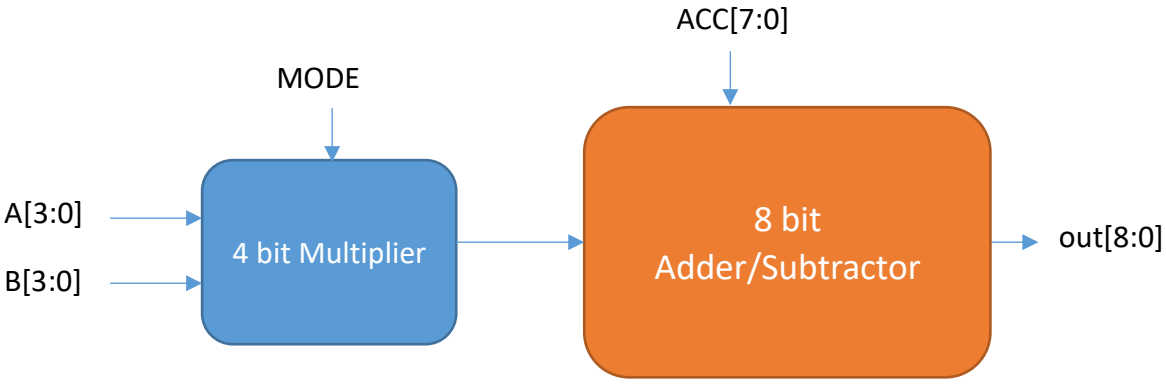
   1. 第一級: 1st CSA
   2. 第二級: 2ndCSA、3rd CSA、4th CPA[0]、8bitFA[0:3]
   3. 第三級: 4th CPA[1:3]、8bitFA[4:7]、OUT[8]

加減法部分是用 two stage 8bit ripple adder 來實現，sign extend 則是需要特別設計，乘法前三級的每級只需要將最前面的 sum 做 extend，到下一級就可，但是最後一級需要判斷 A[3]是 0 or 1，若是 0 即可直接全部填 0，若是 1 則需要把 B[3:0]做 2 補數運算並且 sign extend，而 OUT[8]需要對 partial_sum[7] xor ACC_2complement[7] xor COUT 來得到。

Verilog 架構：

A. 大致架構及所用元件

Verilog 的部份我們用模組化的設計，先用 gates 兜成需要的元件：Full Adder、Half Adder、Four-bit Multiplier (利用 FA、HA)、Eight-bit selective Adder/Subtractor (利用 FA)；整體的架構就是利用 four-bit multiplier 先對輸入 A、B 進行乘法，然後輸入進 8bit 的加減法器依照 mode 去和 ACC 進行加減得到答案，如下圖：



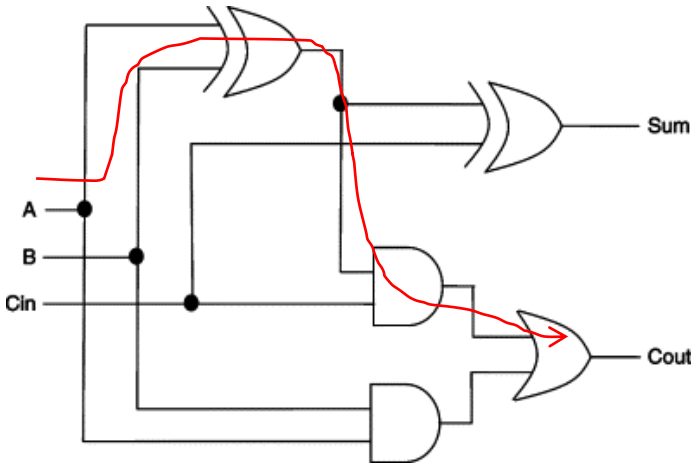
B. 元件設計討論

因為 multiplier 的部分由 FA 組成，架構適用標準的 carry-ripple adder，較需巧思的就是減法補述的部分，而補數的設計概念也已經在 presim 以及接下來的 discussion 討論，所以以下將對 Verilog 裏頭設計 FA 進行比較詳細的探討。

如下圖有兩種 FA 的設計方法，兩者的 critical path 也如標示所示，可以看到，利用 XOR 實現的 FA 有較短的 critical path，時間上較用 NAND 實現的 FA 來的有效率

（比較兩者的 critical path delay，前者的效率約為後者的 $\frac{3\ ns}{1.9\ ns} \approx 1.58$ 倍）

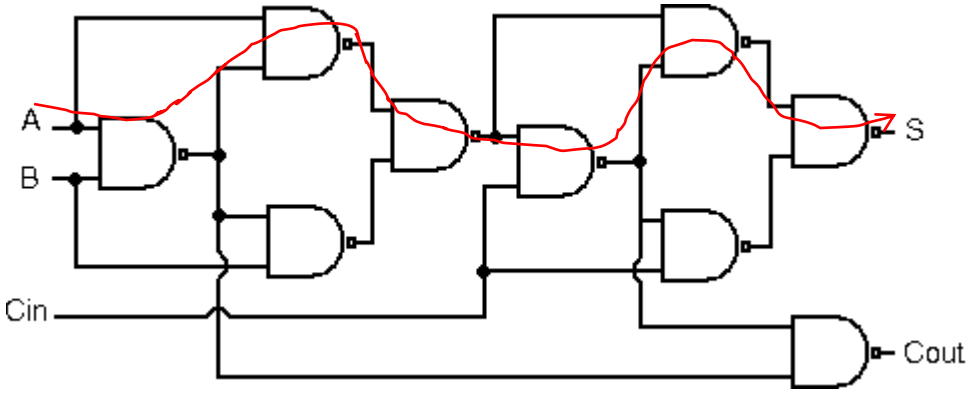
所以在這次的 FA 設計中，我們選擇使用此架構來實現我們的 FA。



critical path delay time = 2-XOR$_{delay}$ + 2-AND$_{delay}$ + 2-OR$_{delay}$

= 0.7 + 0.6 + 0.6

= 1.9 ns



critical path delay time = 2-$NAND_{delay}$ * 6

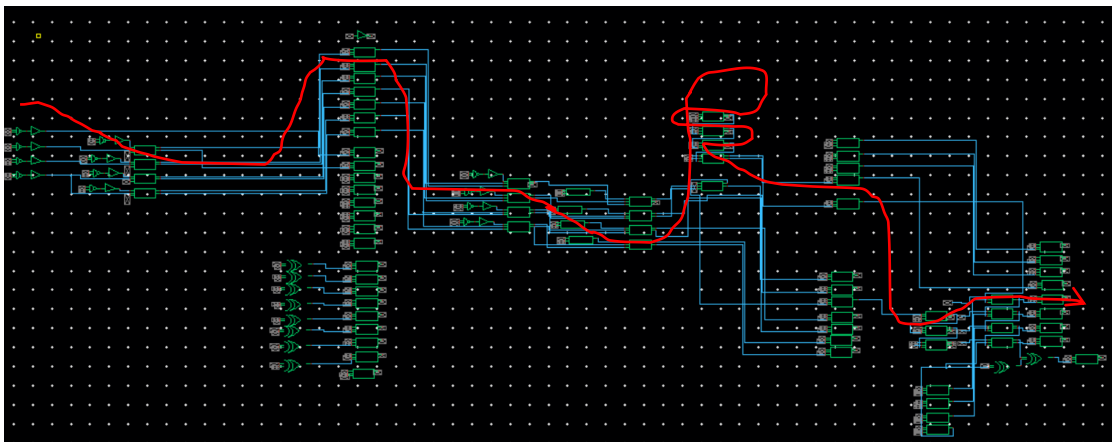
= 0.5 * 6

= 3 ns

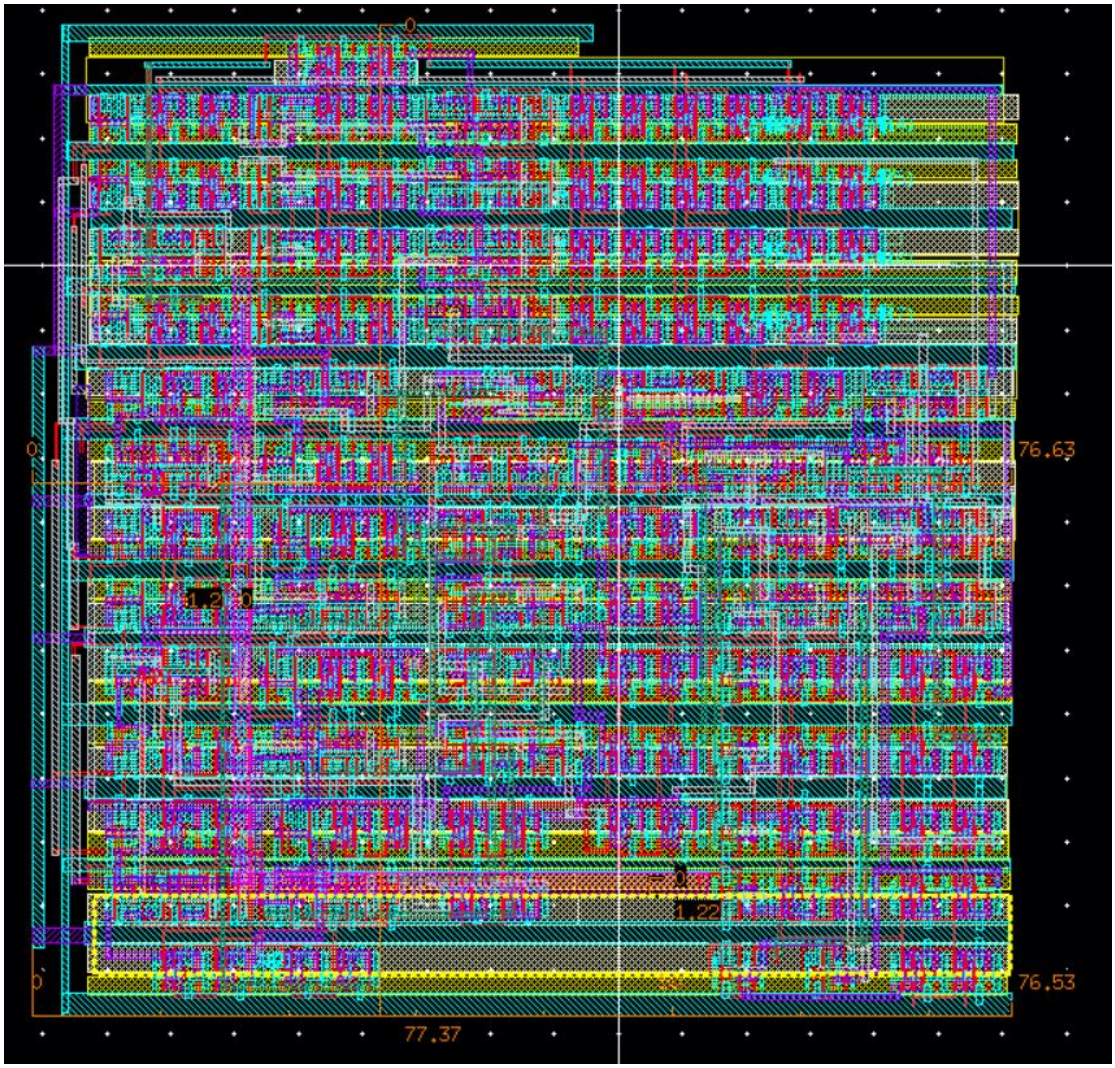
最後整體 Verilog simulation 出來的 clock speed 是 13.3 ns

Discussion of Worse Case Pattern

我們這次的 presim 結果 clock period：1.24ns (frequency：0.81GHz)，當 clock period 為 1.23 的時候，第一個會錯的訊號為 out[4]，而就此輸出訊號溯回分析，因該是因為這條就是 critical path，如下圖。此 critical path 經過了 4 個 FA、3 個 DFF 以及一個 NAND 和 inverter，原因可能是因為在進行乘法中加法的時候第五位為 most significant bit (更大位數不明顯因為有用到 CSA)，所以在 carry-ripple 的過程中就有較長的 delay，以至於把 clock 壓低的時候，這個輸出會先錯。



Layout Photo



Discussion

1. 對於 signed 的處理我們想了非常的久，後來才知道對於 signed 來說，sign bit 的 partial product 必須把它減回來，所以需要多加上 2 補數的設計；
2. 對於任何 signed 的加減法我們必須多考慮一個 bit，例如 8bit 加法我們需要用 9bit 的 FA 才能求出正確的 sign bit，否則可能會有 overflow 的問題。
3. 目前的 critical path 落在中間的級數中，提早做的 8bitFA 我們切了 4bit 與 CSA 同步做，這樣可以節省很多時間，因為異於典型的 carry-ripple adder 不需要讓 carry propagate 到最後一位的加法以致延長 critical path 的 delay，而是「記住」每一位加法的 carry 在最後加起來就好。
4. 我們利用了第一級 pipeline 前的時間，讓他提早做，並且在算出某些值之後直接提早讓他做相加，節省時間。
5. 在進行 layout 排成的時候，我們盡量將 input signal 或是 wiring 相連接的模

組相鄰排放，這樣可以避免把線繞太遠，為實際設計圖帶來不必要的複雜度和增加 layout 的清晰度，在進行 LPE 的時候也可以減少過長線路帶來的額外 delay。

Thoughts of Final Project and Course

我認為這個 final project 讓我意識到 pipeline 分配的重要性，若非常妥善的利用 pipeline 每一級的時間，便可以把整個 cycle time 拉得非常短。所以其實整個 project 除了對 signed 的處理需要比較花時間去思考，把邏輯弄對後，大多數的時間幾乎都花在如何緊密的切 pipeline 了。

對於課程我覺得非常的扎實，無論是理論或是實作都非常的需要時間去吸收消化。

圖片來源：

Full Adder with XOR：https://www.sciencedirect.com/topics/computer-science/full-adder

Full Adder with NAND：https://www.researchgate.net/figure/Full-Adder-realization-using-NAND-gate_fig2_331834266