臺北市立建國高級中學 生活科技科 學期專題

二年十班 毛柏毅

二進位加法器電路實作

概述

在現代電腦的資訊操作中，加法是不可或缺的基礎運算。在實際學習邏輯電路設計之前，我就有試著透過各種不同的方式來試作加法器，如今有機會認識並操作各種不同的電路設計軟體，我決定趁著這個機會來實際研究加法器的製作。在此我將先透過 CircuitVerse 測試其邏輯電路，再以 Multisim 模擬積體電路，並且做出於 MOSFET 的實現。

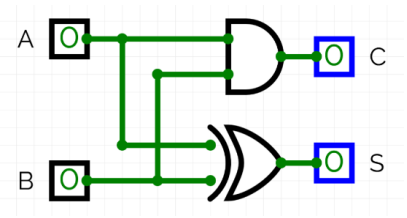
半加器

C		B	
		0	1
A	0	0	0
	1	0	1

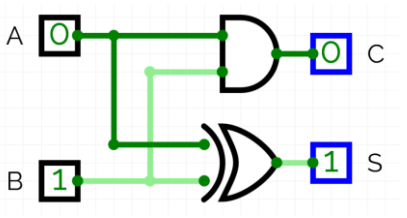
S		B	
		0	1
A	0	0	1
	1	1	0

在處理完整的加法器之前，需要先能夠製作基礎版的半加器。半加器接受 AB 兩個位元的輸入，並且輸出兩個位元 S 和 C。其中 S 代表兩輸入加總後的低位，而 C 則是代表加總時產生的進位，以數學式表達便是 A+B=2C+S。藉此可分析兩個輸出的真值表如上圖。從分析可以看出，事實上兩個輸入都可以個別對應到簡單的邏輯閘：S 對應 XOR、C 對應 AND。由此可透過 CircuitVerse 做出對應的數位邏輯電路如下。

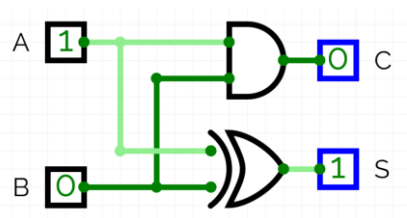
$0_2+0_2=00_2$



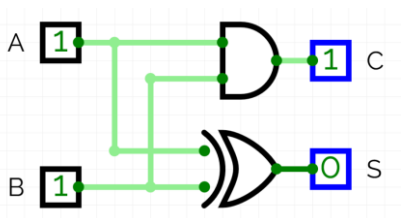
$0_2+1_2=01_2$



$1_2+0_2=01_2$

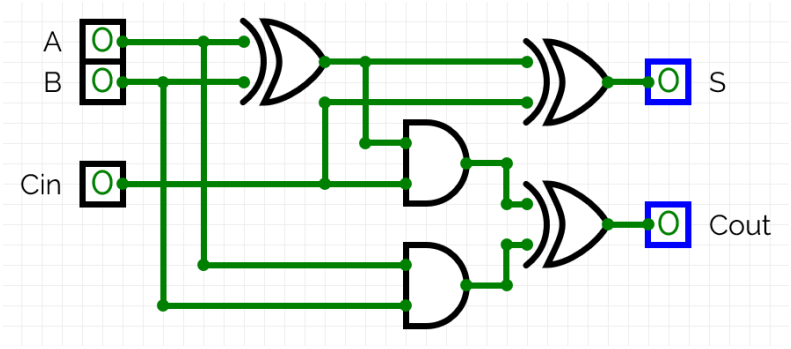


$1_2+1_2=10_2$



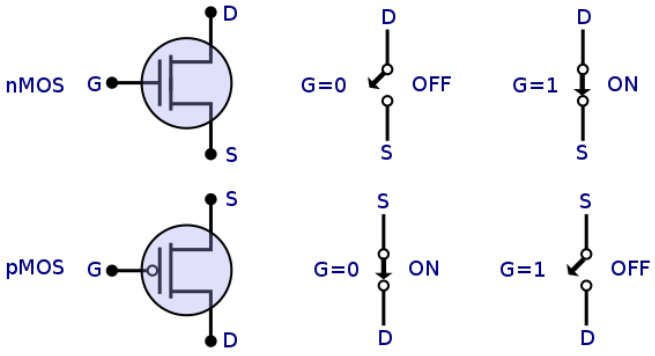
全加器[1]

然而半加器對於進位的處理不甚完善，因此我們必須利用全加器來做進一步的操作。相較於半加器，全加器多了一個輸入 C_{in}，代表前一位做加法後所得的進位資訊。在實務操作上，可以將全加器看作是輸出「三個輸入位元的總和和進位」(S 和 C_{out})。以數學式表達便是 $A+B+C_{in}=2C_{out}+S$。S 為真的情況，便是三個輸入中必須有奇數個 (1 或 3 個) 為真，可以透過 `A^B^C` 達到這樣的效果。而 C_{out} 則需要三個輸入中有至少兩個為真 (即相加大於 2，有出現進位)，可以透過 `(A&B)|(B&Cin)|(Cin&A)` 達到，經過化簡可變為 `(A&B)|(Cin&(A^B))`[2]。由此可做出對應的數位邏輯電路如下圖。



積體電路上的邏輯閘

不過在製作全加器的積體電路之前，我們得先能夠在積體電路上實現我們需要的基礎 MOSFET 邏輯閘，也就是異或閘和與閘。在此我們可以利用「互補式金屬氧化物半導體 (CMOS)」來製作，也就是透過 nMOS 和 pMOS 來實現所需的基礎邏輯閘。



[1] 本文所用之邏輯運算符號對照：`&`→AND、`|`→OR、`^`→XOR

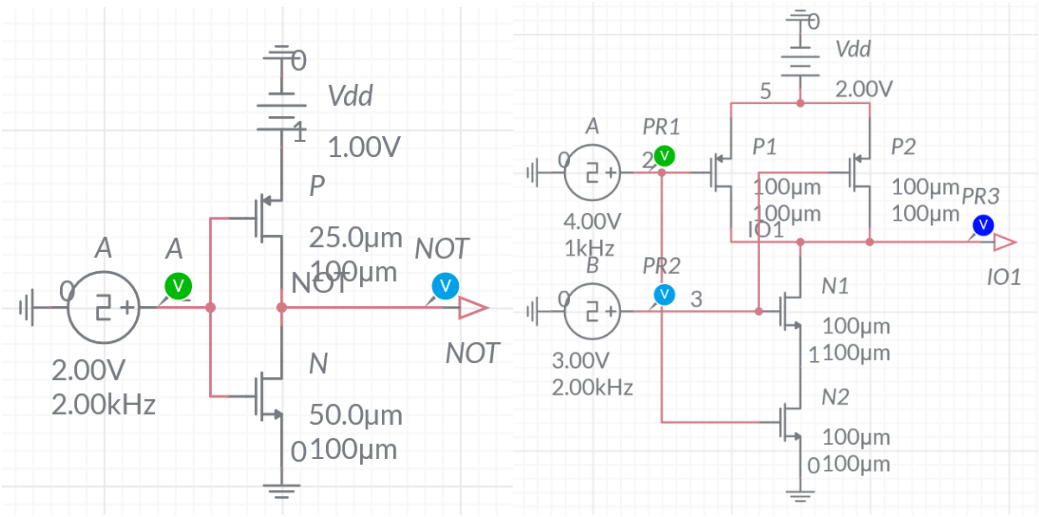
[2] 此處唯一的或運算可使用異或運算代替。由於理論上該運算不會出現參數皆為真的情況 (`A&B=1`→`(A^B)&Cin≠0`)，因此不會對計算結果產生影響。電路圖中即是使用異或閘進行操作。

nMOS 和 pMOS 有著三個端口，透過 G 端的電壓變化來影響 S 端到 D 端的通路。以 nMOS 來說，若是 G 端的輸入電壓大於閾值電壓，則會使得 S 端到 D 端間形成通路，反之則形成斷路。相較之下，pMOS 則是剛好相反，在 G 端輸入低於閾值電壓時形成通路，反之則形成斷路。有了這些資訊，我們便可以做出所需的邏輯閘了。

與閘

在 CMOS 上要實現與閘，最傳統的方式便是透過非與閘加上非閘來達成，也因此，我們需要能夠先做出 NOT 和 NAND 的積體電路實現。

首先是最簡單的 NOT，其 CMOS 實現如左下圖，僅包含了一個 nMOS 和一個 pMOS。當輸入端為低電壓時，pMOS 在輸出端與上方的電源之間形成通路，導致輸出端出現高電壓；當輸入端為高電壓時，則是下方的 nMOS 在輸出端與下方的接地之間形成通路，導致輸出端出現低電壓。

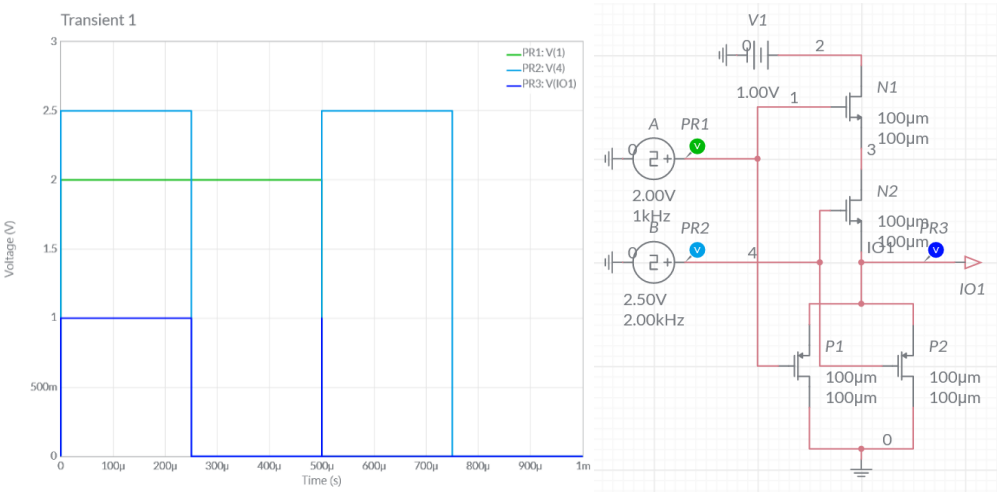


再來是 NAND，其 CMOS 實現如右上圖，包含了一組並聯的 pMOS 和一組串聯的 nMOS。若是兩個輸入端皆為高電壓，則下方的串聯 nMOS 會在輸出和接地之間形成通路，導致輸出端出現低電壓；反之若是有至少一個輸入為低電壓，則下方變成斷路，而輸出端會藉由其中一個或兩個 pMOS 與電源形成通路，導致輸出端出現高電壓。

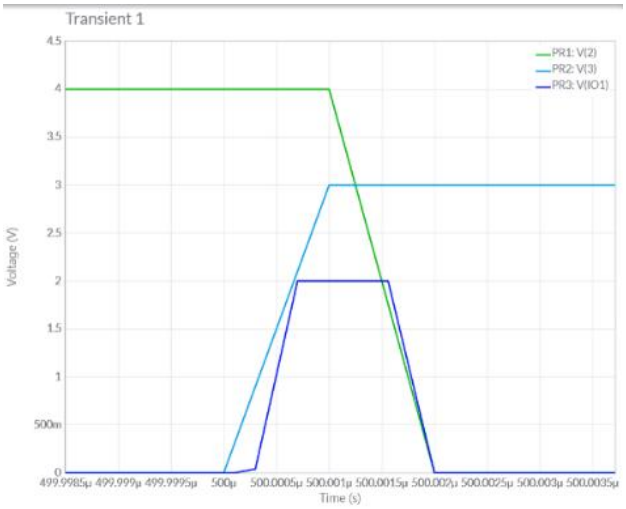
簡化版與閘

然而事實上在看到 NAND 的結構之後，我們可以發現：若是將其電源和接地倒置，便能夠直接達到反相訊號的效果。也因此，我們就只需要四個電晶體

就能夠運行，其 CMOS 實現如右下圖所示，此處我們將 pMOS 和 nMOS 倒置（而非電源和接地）。當兩個輸入皆為高電壓時，串聯的 nMOS 便會使得電源和輸出產生通路，進而輸出高電壓。反之若是至少一個輸入為低電壓，則下方並聯的 pMOS 便會使接地和輸出產生通路，使得輸出變成低電壓，其生成訊號如左下圖所示。(圖表線條顏色對應電路圖中電壓計數值；此外將不同電源的電壓調成不同的數值，以凸顯訊號變化的對應關係)



值得注意的是，在電壓變化圖中央處有一瞬間出現了突起 (如下圖之局部放大圖所示)。這是因為輸入端的信號生成器在切換的時候會有時間延遲，當原本高電位的訊號還沒來得及降低至閾值以下前，原本低電位的訊號就已經升高至閾值之上，在這裡的情況下，就會導致在判定上兩者皆為真，進而使輸出也跟著變為真。

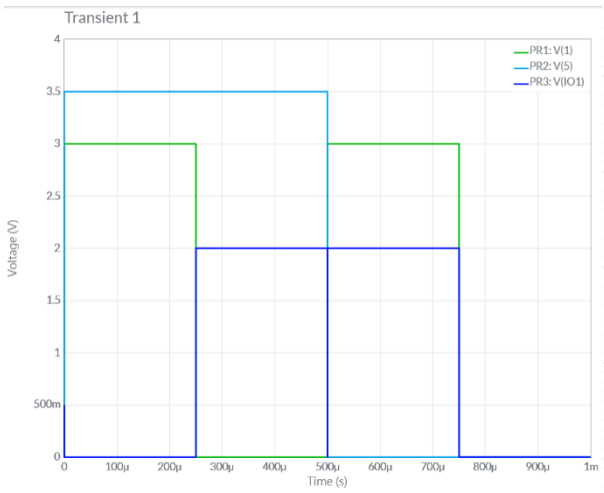


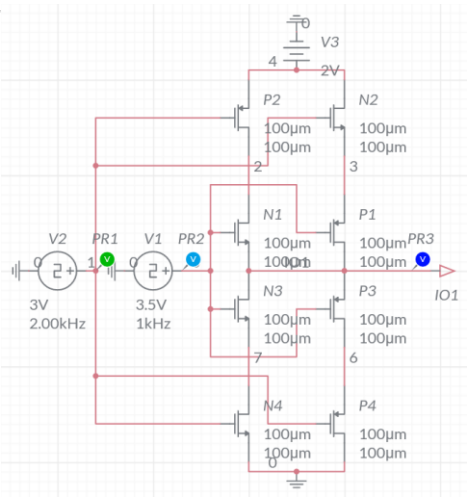
在此我們不妨少歇一會，思考一下前文所提 AND 和 NAND 實現的原理。當出現串聯的電晶體時，就等同於是將兩個電晶體所代表的訊號做且運算，例如兩個 nMOS 串聯，則代表需要兩個輸入皆為高電壓，該線路才會形成通路；

而出現並聯的電晶體時，則是形同對兩個電晶體所代表的訊號進行或運算，如兩個 pMOS 並聯，便代表需要至少一個訊號為低電壓，才能使得該線路形成通路。此外為了避免電壓無處可去而回流，或者是輸出端同時與電源和接地產生通路而出現預期外的分流，對於每一種可能的輸入，都必須要有相應且唯一的線路與輸出端接通（無論是電源或接地），方能維持穩定的輸出電壓和訊號。

異或閘

透過前文的分析以及異或閘的真值表，我們可以發現，異或閘需要對四種輸入組合分別進行處理（不像 AND、OR 可以將三種情況合併處理），也就是說總共需要八個電晶體，方能夠製作出可信且穩定的異或閘，其 CMOS 實現如右下圖所示。上半部的兩組串聯電晶體對應的是兩輸入相異的情況，此時將輸出與電源接通，產生高電壓；而下半部的兩組則是對應兩輸入相同的狀況此時將輸出與接地接通，形成低電壓。如此一來，便能夠形成如同左下圖的訊號變化。此外同樣地，在訊號切換之間會出現延遲，進而產生圖表中呈現出的突波。



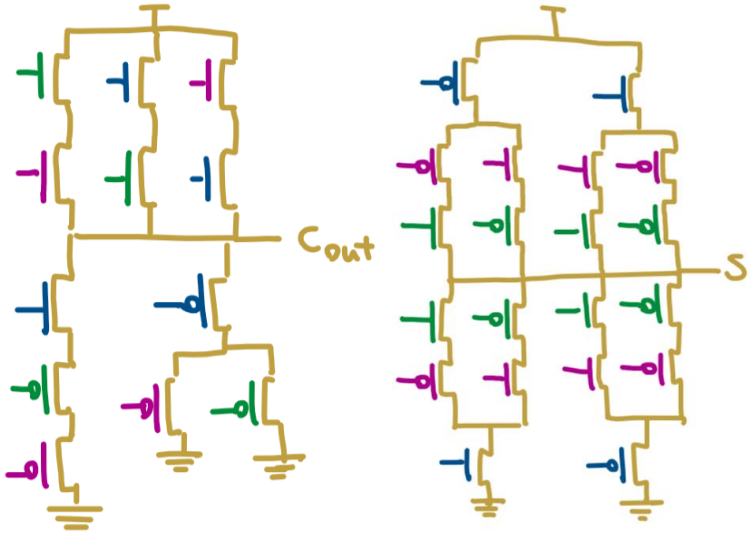


等等，全加器可以更簡單嗎？

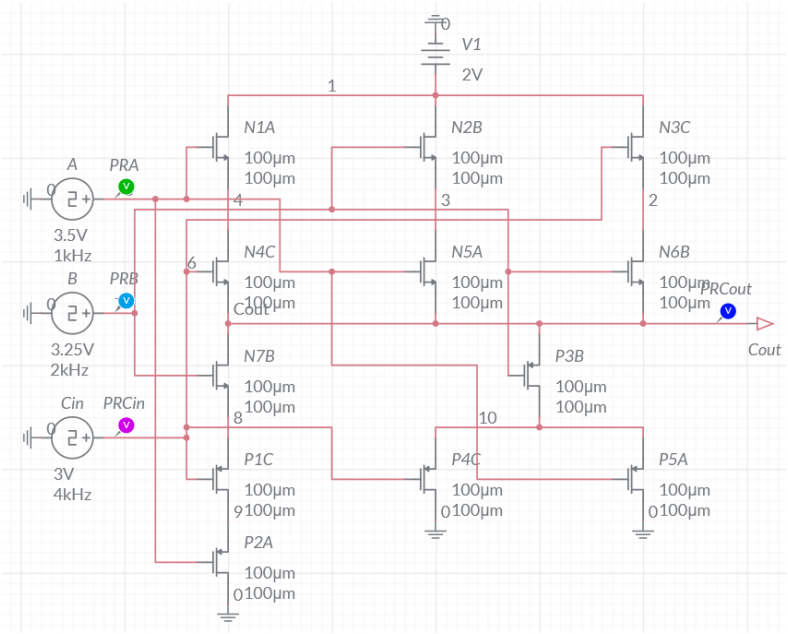
在前面操作完與閘和異或閘的實現之後，我對於 MOSFET 有更清楚的認知，也不禁讓我思考：是否能夠活用電晶體的特性，來將所需的電晶體數量盡可能壓低呢？畢竟在操作上這樣比較輕鬆，實務上也能夠降低積體電路的大小。事實上在實際應用的時候，全加器也有可能會透過「電晶體級」的電路，減少為了達成邏輯閘而產生的冗餘電晶體。

事實上，如果使用我們於前文用邏輯閘製作的設計來實現全加器，其總共需要至少 32 個電晶體。然而此處與其用`(A&B)|(Cin&(A^B))`操作 Cout，不

如直接利用原本的(A&B)|(B&C_{in})|(C_{in}&A)來製作，這樣一來透過適當的電路安排，便能夠將所需的電晶體數量減少四個，其設計如右下圖所示（紅藍綠三色分別代表不同的輸入端）。此外輸出 S 也可以被合併為單一的邏輯閘（事實上市面上亦有生產三輸入異或閘），但以我們目前的設計，所需使用電晶體的數量反而會增加四個，兩者互補之下並沒有辦法帶來太大的效益。儘管如此，為了呈現上的方便，以下仍會把 Cout 和 S 的電路設計和輸出分開呈現，因此在 S 的部分會採用三輸入異或閘的設計。

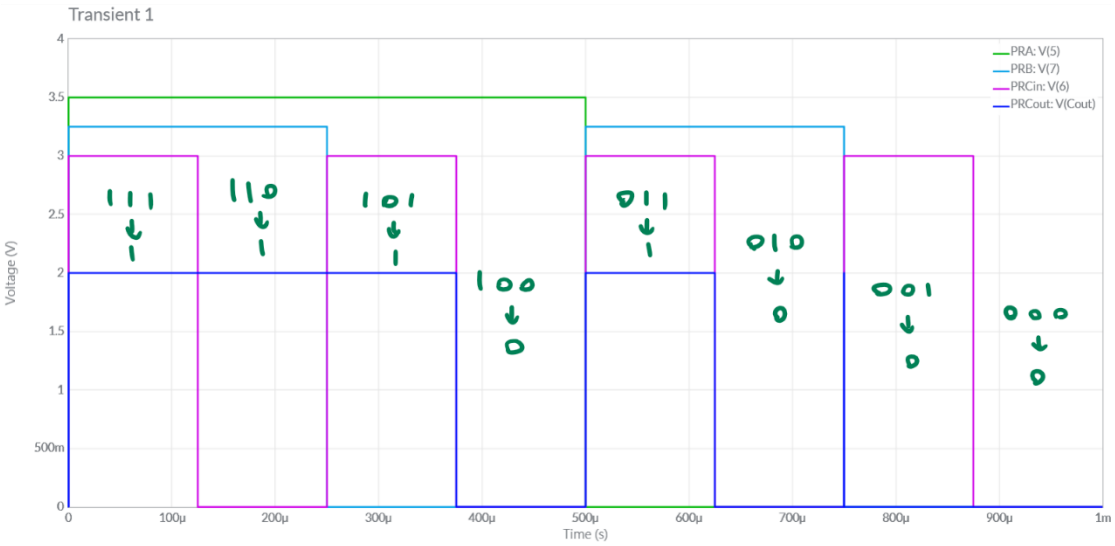


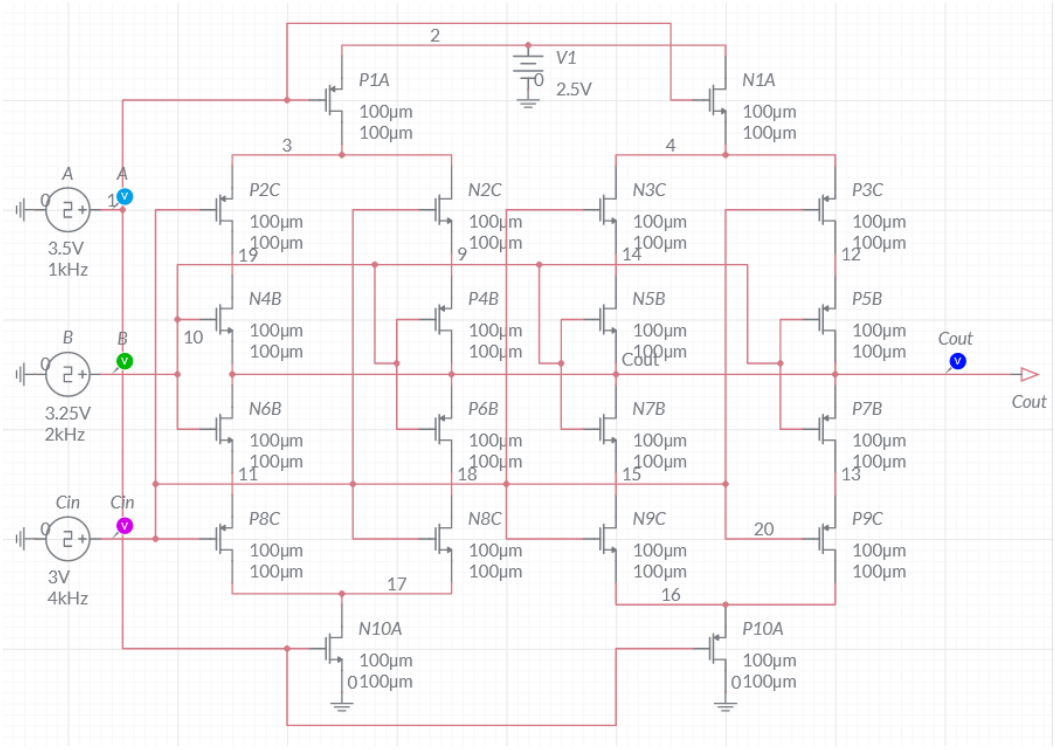
全加器的實現



有了上述的設計圖，我們終於能夠開始依樣製作 MOSFET 的全加器了。首先是 C_{out} 的部分，透過前述的設計圖，我們能夠做出如上圖的電路配置，其中

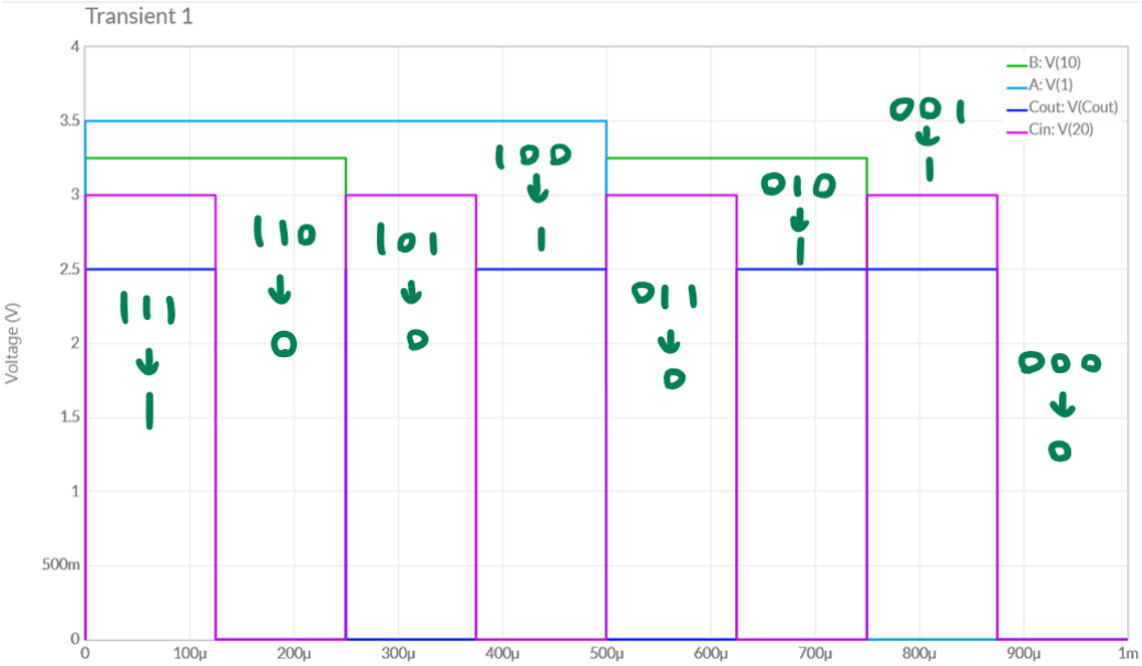
每個電晶體名稱的首個字母為其對應的種類，最後一個字母為其 G 端所接收的訊號源。若是分析這份電路設計，在上半部有著 AB、BC、AC 三組串聯電晶體，即(A&B)|(B&C_{in})|(C_{in}&A)；而下半部則是其相反的訊號，即(B&!A&!C)|(!B&(!A|!C))。由此所得到的訊號對應關係如下圖，可以看到對於每一種輸入組合，我們的電路都能夠輸出正確的訊號，並且維持原本輸入的電壓。





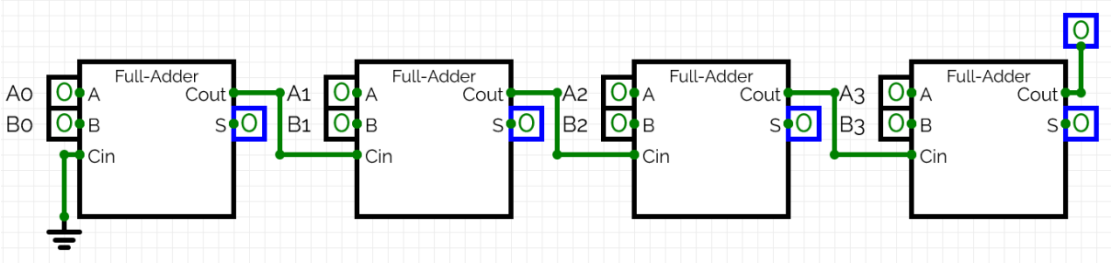
再來是 S 的輸出，由設計圖可做出如上圖的電路。同樣地，其中每個電晶體名稱的首個字母為其對應的種類，最後一個字母為其 G 端所接收的訊號源。

雖然電路看起來十分龐大，事實上就只是把三個輸入的所有可能狀態陳列出來，並且對某些部分做合併簡化（即輸入 A 所連結的四個電晶體），就此可以把輸出應為高電壓的所有輸入組合放在上半部，並將輸出為低電壓的組合放在下半部，這樣便能夠產生如下圖的訊號變化，並維持原本輸入的電壓。

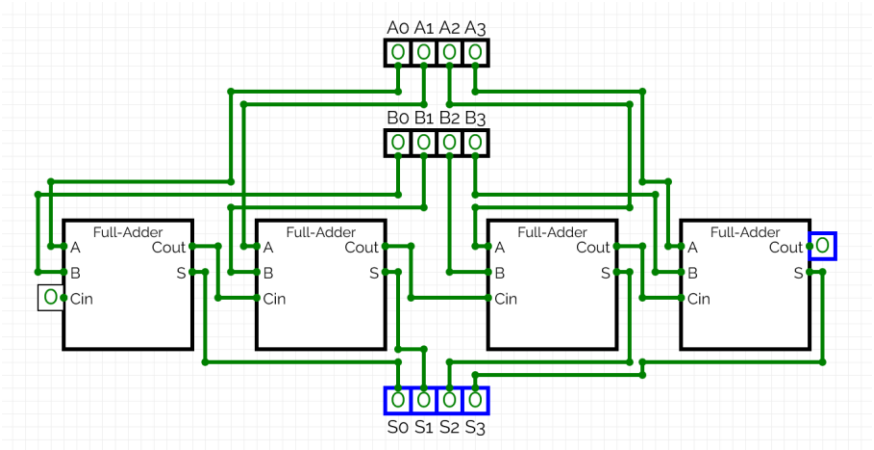


加法器

有了全加器，我們便可以輕鬆做到多位元的加法運算。只需要將數個全加器互相串聯在一起，再將各位元的輸入對應到各自的全加器，便能夠實現多位元加法器。以下圖的四位元加法器為例，將輸入 A 和輸入 B 的第一個位元接到第一個全加器的 AB 輸入端，並且將 C_{in} 接地（即輸入 0），此時會得到第一位元加總後所得的 S 和 C_{out}，其中 S 便是計算結果的第一位元，而 C_{out} 則是傳輸至第二個全加器，與輸入 AB 的第二位元做加總。在這樣層遞之後，到了最後一個位元的輸出，此時的 C_{out} 沒有下一個全加器可去，因此便形成了溢位，該位元的資料就會直接被捨棄掉。這樣一來，我們就能夠達到四位元資料加法的效果，輸入兩個四位元的數值，得到一個四位元的輸出。



有了上圖的架構之後，我們也可以把輸入和輸出單獨提出，並個別擺在一起以便觀察，其成果如下圖（注意其中最低位被擺放在畫面左方），最終便能夠做出 A+B=S 的四位元加法器。順道一提，目前所呈現的四位元加法器只有四個全加器，也就是可以處理四個位元的資料。不過現代計算機科學所常用的 32 位元整數或 64 位元長整數，要製作相對應的加法器，便分別須要堆疊 32 個和 64 個全加器。



專題延伸

雖然我們成功使用 MOSFET 製作出可以正常運作的加法器，但是它仍然有很多缺點有待改進。其中最重要的是在電晶體數量的部分，由於要維持輸出電壓的水平，上述的加法器一個就會使用掉足足 32 個電晶體，若是層層堆疊起來，其電路體積將會變得十分龐大，延遲也會變得很可觀。事實上在網路上有文獻稱能夠做出只需使用 8 個電晶體的加法器，如下圖。(其中也運用到了只需三個電晶體的 XOR）然而幾經嘗試之後我也發現其缺陷，即其運算過程中會出現電壓的耗損，因此最終還是決定使用如今所呈現的加法器設計。未來若是有機會，或許我們也能夠從這份設計圖下手，盡可能改良目前所製作的加法器，減少使用的電晶體數量，同時維持穩定可靠的輸入輸出。