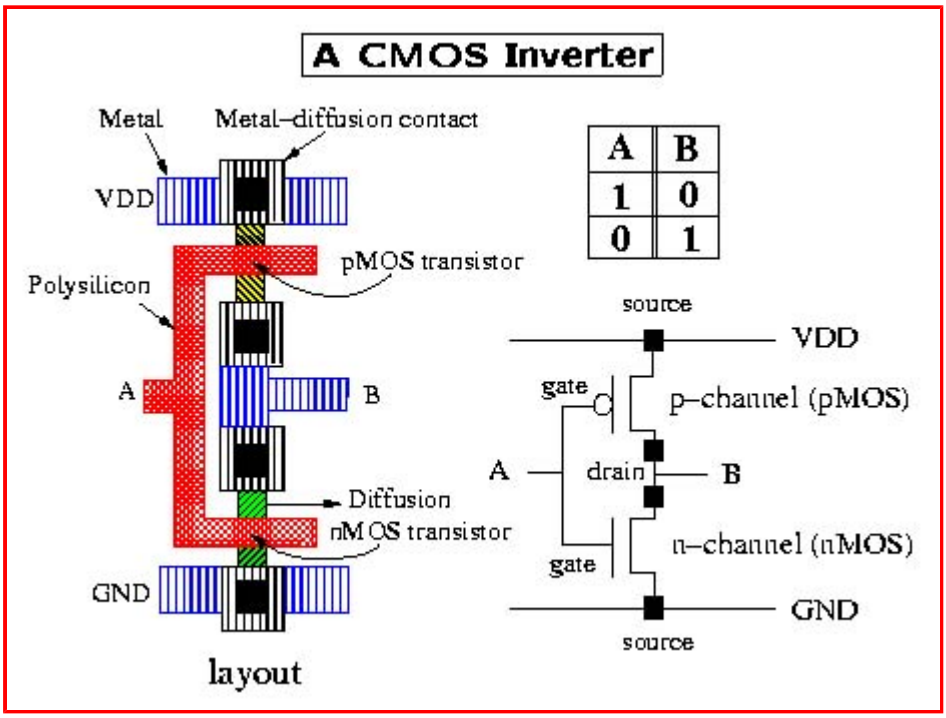
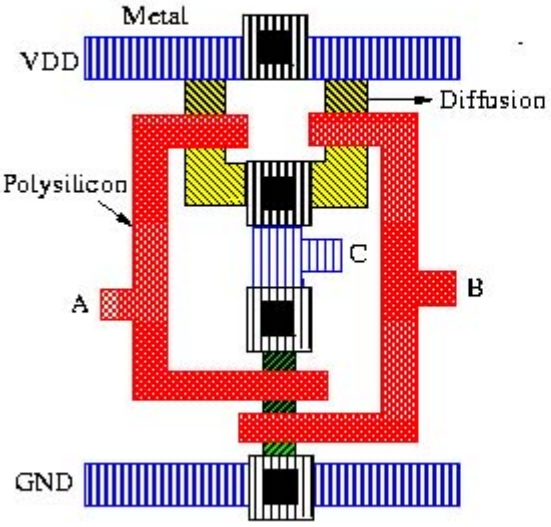
一. 前言

到目前為止,我們已經學會使用電路圖描述邏輯電路,以及如何編譯它們, 模擬它們,最後再將它們下載到XS40電路板上.看起來它們好像跟您預期的一樣.但是當我們在編譯與模擬設計檔案時,很容易就忘記了,事實上電子元件有其自己的特性.這個章節會討論與分析一些邏輯電路的電子元件特性.如果了解這些簡易的電子元件特性,就可以了解所設計的電路為什麼有時候並不能如原本預期地運作了.

二. I/O之特性-indeterminate state



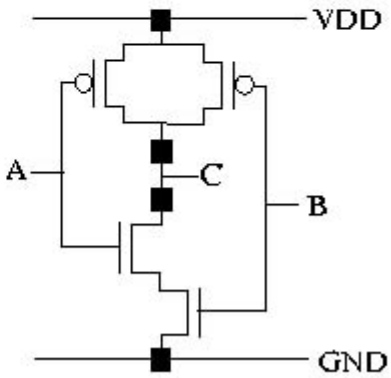
A CMOS NAND Gate



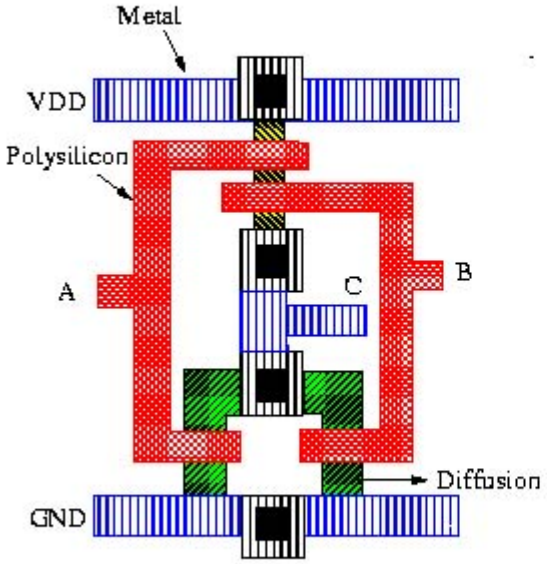
layout

in1 in2 out

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

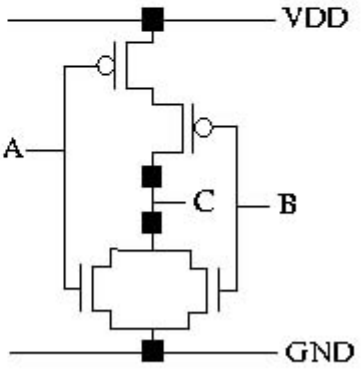


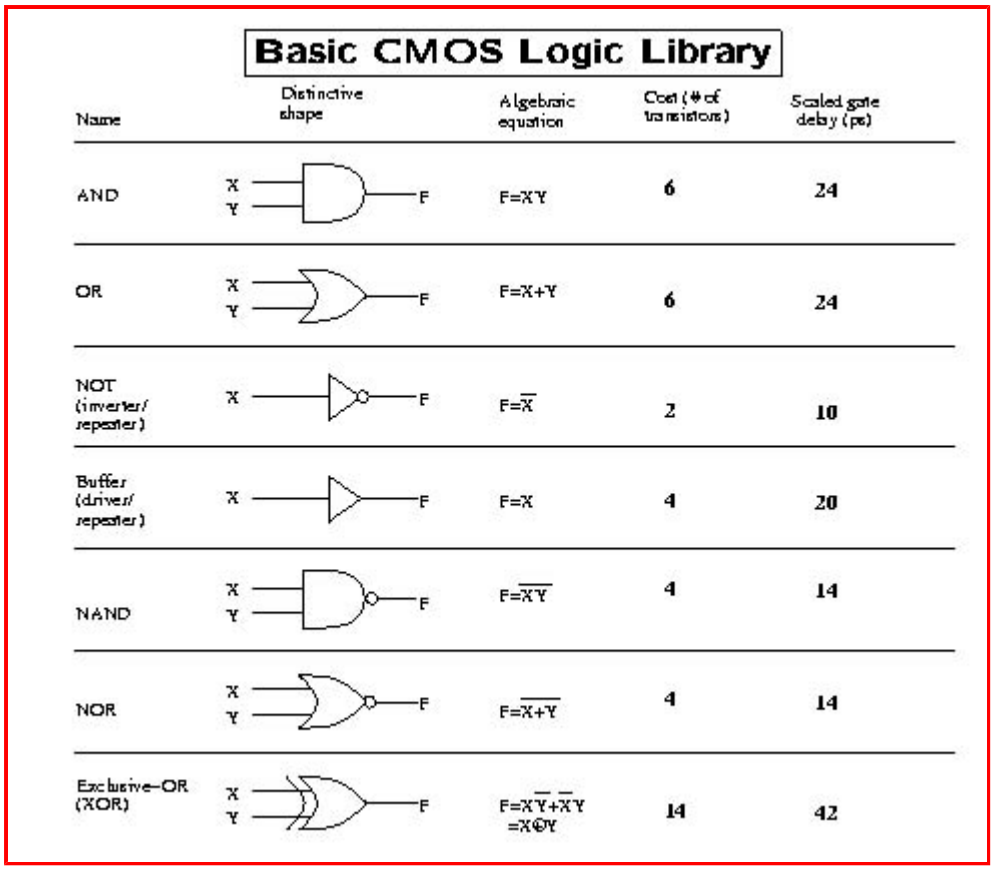
A CMOS NOR Gate



layout

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0





Basic CMOS Logic Library

Name	Distinctive shape	Algebraic equation	Cost (# of transistors)	Scaled gate delay (ps)
AND	X, Y → F	F=XY	6	24
OR	X, Y → F	F=X+Y	6	24
NOT (inverter/ repeater)	X → F	$F=\overline{X}$	2	10
Buffer (driver/ repeater)	X → F	F=X	4	20
NAND	X, Y → F	$F=\overline{XY}$	4	14
NOR	X, Y → F	$F=\overline{X+Y}$	4	14
Exclusive-OR (XOR)	X, Y → F	$F=\overline{X}Y+X\overline{Y}$ $=X\oplus Y$	14	42

一個簡易的標準輸出反相器如下圖(a)所示.它是由兩個電晶體所組成的, 其中:T_h電晶體是被連接在反相器的輸出接腳與+5V電源供應器之間;T_l電晶體則是連接在反相器的輸出接腳與接地端之間.

如果我們將反相器的輸入端接地(形同輸入0,如下圖(b)),電晶體 T_l就變成off,而電晶體T_h就轉變成on. 此時 T_l表現像是一個打開(open)的開關,而T_h則是緊閉 (close)的開關.結果在輸出接腳端到+5V的電源供應端之間就會產生一條低電阻的路徑.這樣子就會將輸出端接腳的電壓升高到+5V.(形同輸出1)

當輸入端被固定在+5V時(形同輸入1,如下圖(c)),則產生與之前相反的動作;電晶體T_h就變成off,而電晶體T_l則轉變成on. 也就是T_h表現像是一個打開(open)的開關,而T_l則是緊閉(close)的開關.結果在輸出接腳

端到接地端產生一條低電阻的路徑. 這樣子就會將輸出端接腳的電壓拖曳到+0V.(形同輸出0)

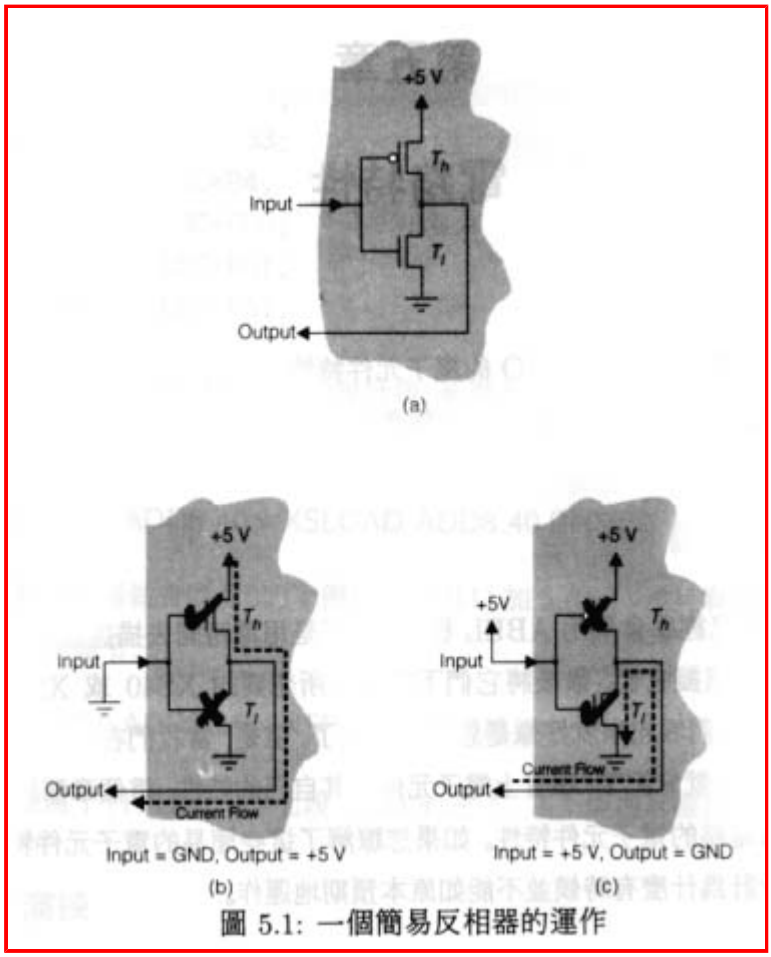


圖 5.1: 一個簡易反相器的運作

假如我們輸入的電壓值介於+5V到+0V之間的話會發生什麼樣的情況呢? 在這種情況之下,T_l和T_h兩個電晶體就會出現相互角力的情形,一方面T_l嘗試要將輸出端電壓拖曳到+0V,同時另一方面T_h也嘗試著要把輸出端電壓升高到+5V.結果輸出端的電壓將會在+5V的電源供應端與接地端之間,處於不明確的狀態.

這個問題的關鍵在於,當輸入多靠近+5V或是+0V時,我們才能判別此輸入屬於+5V或是+0V呢?這就得依據電晶體T_l和T_h細部電子元件的特性來決定.對於TTL的電子元件特性而言,一個有意義的低電壓邏輯是介於0V到0.8V之間,有意義的高電壓邏輯是介於2.0V到5.0V

之間.對於 CMOS的電子元件特性而言,電壓分佈的範圍就不太一樣,對於CMOS的電子元件特性而言,一個有意義的低電壓邏輯是介於0V到1.0V之間,有意義的高電壓邏輯是介於3.5V到5.0V之間.

如果我們沒有連接任何訊號到輸入端,那麼電路的狀態將會是不確定的, 因為沒有實質的電壓輸入驅動電晶體,所以此時的電晶體可以是on或是off. 所以,其相對應的輸出端電壓也就可以介於0V到5V之間的任何電壓值.如此一來,就會造成了電路處於不明確的狀態.

為了避免上述的情況發生,有些IC在每個輸入端到+5V的電源供應器間多加了一個提昇電阻(pull-up resistor),如下圖(a)所示.當輸入端沒有連接任何訊號時,提昇電阻R_p,就會將輸入端的電晶體提昇到+5V.此提昇電阻通常具有很大的電阻值(大於10K).所以,當您想要將輸入電壓降低到接地值(0V),你就不需要提供那麼高的電壓去做這件事了.(如果想用一個電阻小的提昇電阻,那麼當您要將輸入拉到低電壓時,在+5V到接地端之間一定會產生很大的電流,所以不適用)

輸出電路也能被修改成擁有不同的電子元件特性.所謂的標準輸出端,就如同我們之前已經看到的,可以經由低電阻關閉電晶體,使得輸出電壓提昇到 +5V或是降低成+0V.然而,並不是所有的輸出都必須遵守標準輸出端的定義; 在某些的應用裡,您可能只希望輸出端輸出到+0V,而永遠不要提昇到+5V.這時候就必須用到像下圖(b)所示的開放式消耗性輸出(open-drain output).

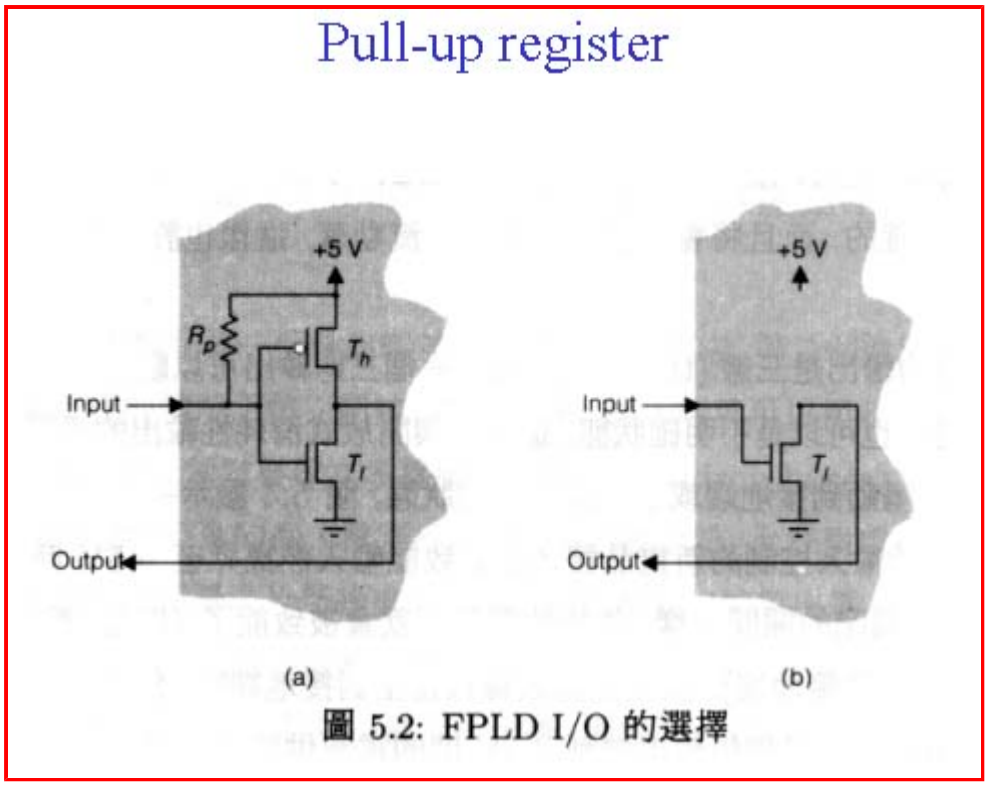


圖 5.2: FPLD I/O 的選擇

開放式消耗性輸出(open-drain output),當輸入是高電壓時,輸出就會經由電晶體T_l拉低至接地端.但是當輸入被拉至低電壓的時候, 輸出端並不會被拉至+5V(雖然+5V屬於標準輸出的電壓值).因為T_h 已經不見了,而且T_l轉變成off,所以輸出端既沒有被連接到+5V ,也沒有接到接地端.所以,此時它就處於不明確的狀態.當然,您也可以在輸出端到+5V之間加一個外部電阻器,如此也能達到與上面同樣的效果.

開放式消耗性輸出的應用是與提昇電阻一起執行接線AND(wired-AND)運算,如下圖所示.如果兩個開放式消耗性輸出直接連接在一起,其中一個的輸出是低電壓的話,則其輸出結果將會是低電壓的輸出.如果兩個輸出都是不明確狀態的話,則線接AND的輸出就會藉由提昇電阻而被提昇至+5V.您不要嘗試著去建構出一個有標準輸出的線接AND,那是因為,一個邏輯閘的輸出也許會試著去藉由它的T_l將線接AND的節點拉至低電壓,

而在此同時,其它的邏輯閘試著要藉由T_h使接線AND的節點提昇成高電壓.如此一來在線接AND節點上的輸出電壓就變成不明確的,而且將會有很大的電流在流動著,這樣也許會燒壞其中的一兩個邏輯閘.

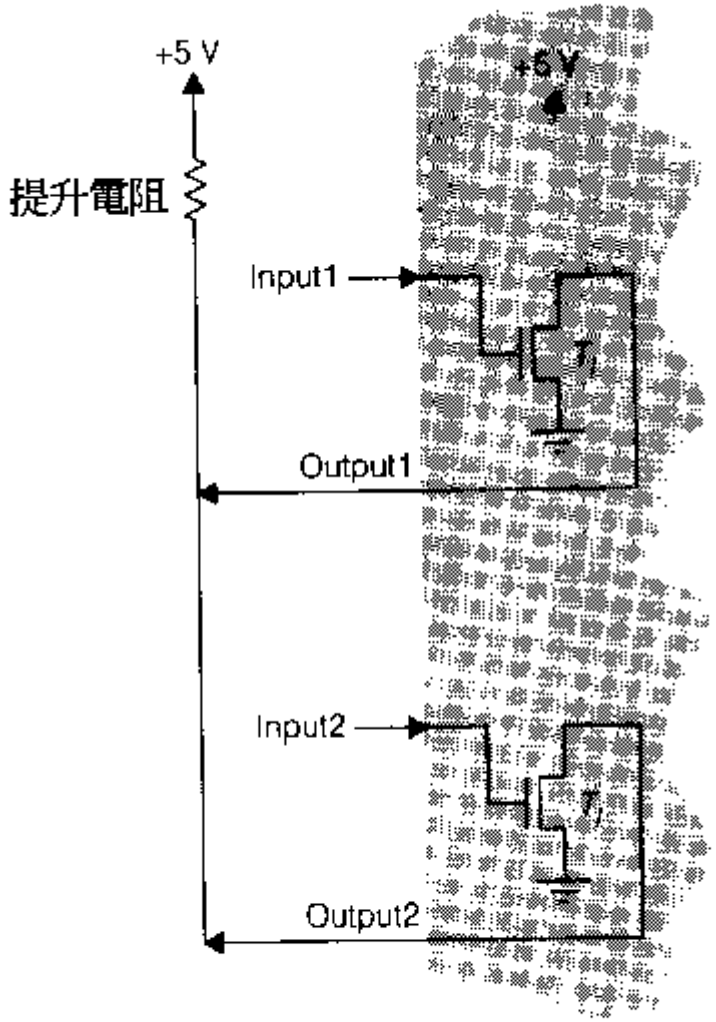


圖 5.3: 使用開放式消耗性輸出的線接 AND

三. I/O之特性-tristate buffer 及其應用

另一種型態的輸出則是三態(tristate)輸出.一個三態輸出可以驅動輸出到+5V或接地端,但是它也可以

是不明確狀態.與開放式消耗性輸出不同的是,開放式消耗性輸出只能驅動到接地端或是不明確狀態,卻不能驅動到+5V.

底下圖示是一個三態輸出的電路.它加入了兩個致能(Enable)輸入控制的新電晶體T_g.當此致能輸入被提昇到+5V時,電晶體T_g 就會表現的像關閉(close)的開關一樣,然後此邏輯閘就被致能了(即是,輸出端可以被驅動成+5V或是接地端).當致能輸入線被拉至到接地端時,然而,此時兩個電晶體T_g都是open,而且使得輸出端到+5V間的電源供應端或接地端完全被切斷了,如此一來,輸出就被抑制,所以輸出端變成了不明確狀態.

三態輸出器有啥功能呢?多個三態多工器可以被連接到另一個邏輯閘的同一個輸入端,而且致能線可以被用來避免輸出端與輸出端之間的干擾.當一個輸出正驅動著輸入端時,其他的輸出端都會被抑制,以至於它們處於不明確狀態而且不能干擾第一個輸出的結果.之後,藉由抑制第一個輸出端並致能第二個輸出端,第二個輸出端可以在不被其他輸出端干擾的情況下傳送訊號.如此一來,輸入端就可以接收到只有一個輸出端所送出的訊號.當您要建立多接腳匯流排(multidrop buses)時,將會使用這樣的技術.舉個例子,個人電腦裡面的記憶體匯流排,允許微處理器去接收從一個大數目的記憶體晶片所送來的資訊.它們都有三態輸出器,所有的記憶體晶片輸出都可以被連接到同一個匯流排,因為這樣就可以會只有特定的晶片可以傳送資訊到微處理機.所以,它們將不會互相干擾,因為同一個時間內,只有一個輸出會被致能.

Tristate buffer 之應用

The output is 0 、 1 or high-impedance

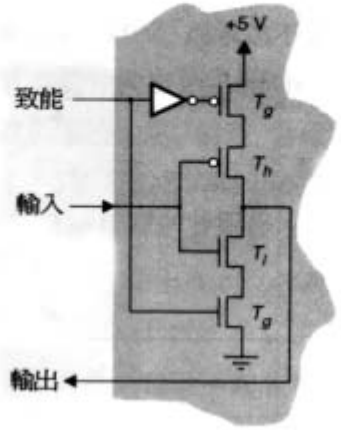
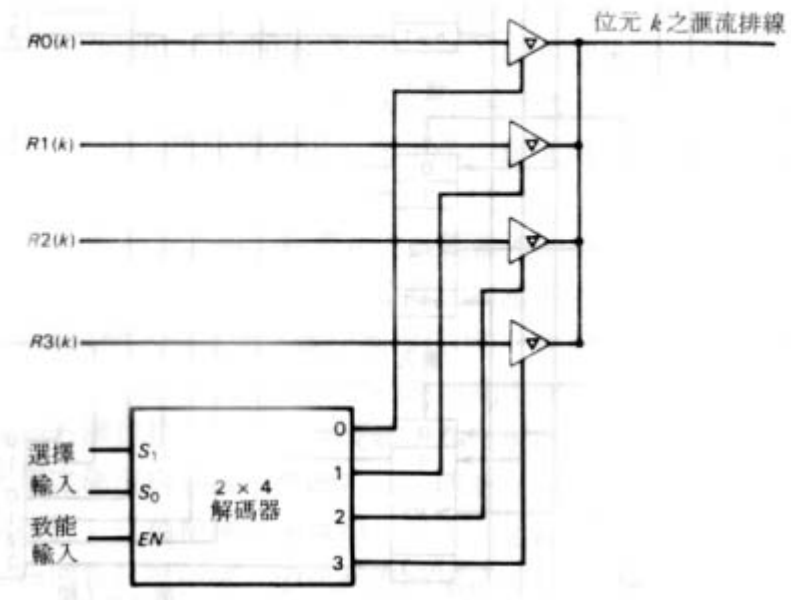


圖 5.4: 三態輸出器的電路

Tristate buffer 應用於匯流排



四. I/O之特性-fanout 之考量

而之前的這些設計,在應用不當時,會帶來什麼樣子的麻煩呢?很明顯地,如果你使用開放式消耗性輸出器,而且不提供一個提昇的電阻器, 你就不能期望輸出會被提昇到+5V.或者是,如果您使用三態輸出器而不去致能它,您將不能期望輸出+5V或是接地值.但是,還有其它更微妙的問題.假設您有一個單一輸出的邏輯閘,此閘被許多其他的邏輯閘當作是輸入端所驅動(如下圖右側).更明確地說,假設這些其他的輸入每個都有10K的提昇電阻.現在,您的輸出可以藉由T_l而被拉低至接地端,但是T_l本身就具備電阻值(意即,它不是個完全理想的導體).假設,T_l的ON-電阻值為100歐母.這個100歐母的電阻值將會試著將蚤IA之電壓拉至接地端,而此時,提昇電壓也正試著要將節點A之電壓提昇至+5V.所以,就形成了一個電壓分配器 (voltage divider).而問題是,在提昇電阻使得節點A的電壓高於0.8V 之前,有多少個輸入可以被連接到輸出端?(對於TTL在低電壓時的最高限制為0.8V)而這個問題的答案就是解下圖左側的式子.

這個式子解出來N=19.所以,這表示您可以從一個單一的輸出端去驅動另外19個其他邏輯閘的輸入.這是您邏輯閘輸出端所容許最多扇出 (fanout)的數目.如果試著驅動更多的邏輯閘將會導致因為抵觸電子元件的特性而不能正常的運作.(注意,驅動輸入端使其在高電壓式沒問題的,因為提昇電阻會幫助T_h使得輸出端提昇為+5V.)

Fanout之考量

$$0.8\text{V} = 5.0\text{V} \times \frac{100\Omega}{100\Omega + \frac{10{,}000\Omega}{N}}$$

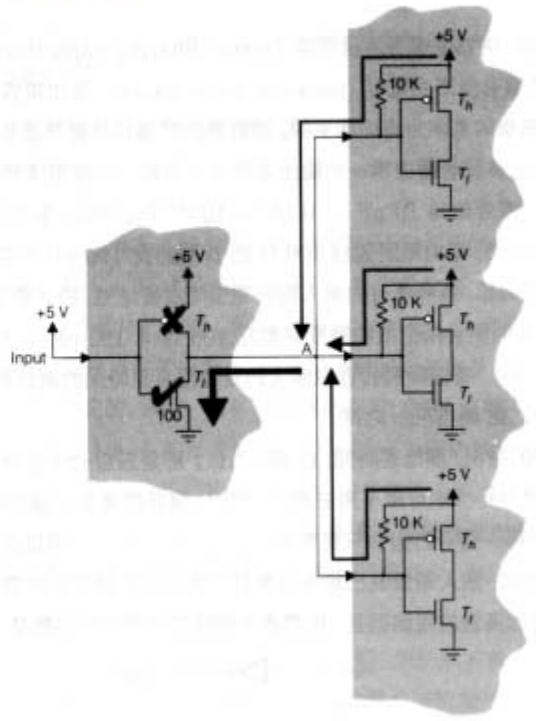
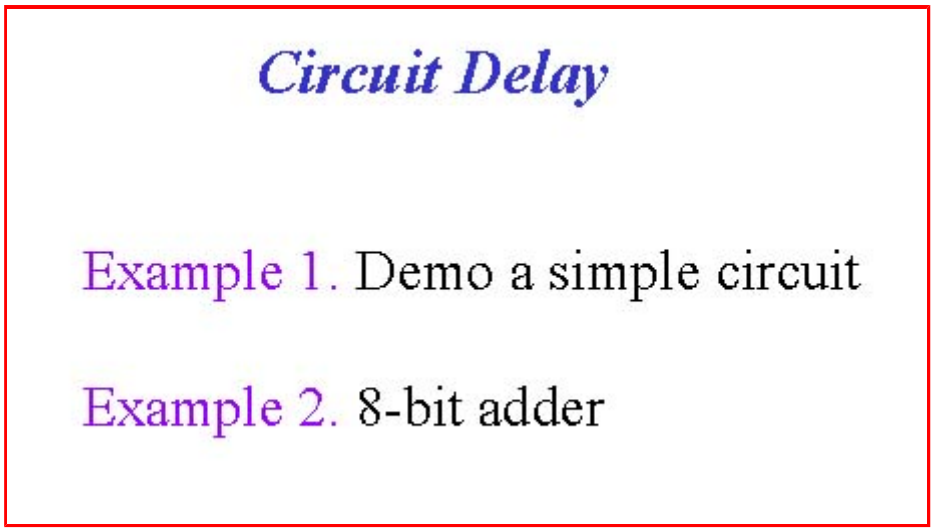


圖 5.5: 邏輯閘的扇出

五. Circuit Delay

驅動其它19個邏輯閘看起來好像是件小事,所以,您也許會覺得扇出(fanout)的數目並不是很重要.然而,仍然還有許多的要素需要去考慮的,其中一項是有關傳送訊號時候的延遲.在電路上的每個節點都有電容(capacitance).在此節點電壓還沒到達+5V或接地值,電容的動作分別地就好像是個即將要被加滿水的一個空水桶,或是要將一個滿水桶慢慢將水倒出來.在XS40或XS95電路板上有您所想要的電容,例如像電源過濾電容(power filtering capacitor).除了電源過濾電容外,當然還有像寄生電容(parasitic capacitance),寄生電容是由電路特性的幾何學與電磁學的基本定理而產生的.而許多的問題往往都是這些寄生電容所導致的.假設,您正嘗試著藉由單一的輸出端將十九個輸入的電壓降到接地值.再假設,每個輸入端的電容值為

10pF(10pF=10^{-13} F).所以,全部就有19X10=190pF的電容值,藉由電阻值為100歐姆的T_l 要將此190pF的電容充電.要將所有的電容充電完畢,並將所有的輸入端的電壓降成接地值,總共要花100歐姆X190pF=19ns(ns:奈秒,相當於10^{-9}.這個時間的近似值大約是電壓衰退時間的兩倍到三倍之間,所以這個值是介於38與57ns之間.



假設最壞的狀況,讓延遲時間,T_d為57ns.即五百七十億分之一秒,也許您可以說這個值小到根本可以不用去理它.然而讓我們去考慮一個200-MHZ的個人電腦.它每計算一個新的指令只需要花 1/(200X10^6 sec=5ns.所以在這短短的57ns的延遲時間裡面 ,個人電腦就已經可以執行大約13個完全的指令運算.這經驗告訴我們: 對於高速的電路而言,我們必須要限制它們的扇出數量.

您不只必須要考慮由扇出數目多寡所造成的延遲時間.而且,邏輯閘本身也具有內建(built-in)的延遲.當邏輯閘的輸入改變時,其相對應的輸出不會立即改變.而這延遲時間主要是由IC內的寄生電容所引起的.從輸入改變開始計算到輸出也改變為止的這段延遲時

間,稱為傳遞延遲 (propagation delay)或是邏輯閘延遲(gate delay).邏輯閘延遲通常有下列兩種型態:
t_{PHL}:從輸入改變開始,到輸出從+5V降到接地值為止的延遲時間(傳遞由高電壓->低電壓)
t_{PLH}:從輸入改變開始,到輸出從接地值升到+5V為止的延遲時間(傳遞由低電壓->高電壓)

通常t_{PHL}與t_{PLH}並沒有太大的不同,所以我們將它們兩個結合起來變成一個單一的邏輯閘延遲參數t_{pd}.在XC95108 CPLD家族晶片裡,t_{pd}的範圍是從7.5ns到20ns, 根據您所使用的晶片價格,一般越快的就越貴.如果是對XC4005XL FPGA 做時脈模擬,就變得比較複雜些,因為延遲時間的長短是依賴在邏輯閘如何地被映射到LUT所決定的.例如,如果有個邏輯閘可以被映射到一個單一的F或G LUT,輸入後只需要1.6ns得延遲時間,就可以在CLB的輸出端得到結果(這是根據XC4005XL FPGA的時脈規格).但是,如果邏輯函數比較複雜,而且需要用到H LUT去結合F與G LUP的輸出,則傳遞延遲就會增加到2.7ns.綜合這兩個情況的延遲時間,都比XC95108的延遲時間還要少, 這是因為它們忽略了從輸入端經由I/O接腳傳遞到輸出端的延遲.

延遲會造成怎樣的後果呢?下圖為一個3位元的漣波進位加法器 (ripple-carry adder)內,指出一條進位訊號的可能路徑.值得注意的是,進位訊號由最低有效位元到最高有效位元一直以漣波方式進位,從這個加法器的每個階段取出兩個邏輯閘延遲來觀察.假設,每個AND-OR 電路是被放置在單一的巨集格裡.因此,對於XC95108 CPLD而言,經過此 3位元加法器的最大延遲為3X20=60ns(但對於XC4005XL FPGA而言為3X2.3=9.6ns).繼而延伸下去,對於32位元的漣波進位加法器,最大的延遲為32X20=640ns(對於XC4005XL FPGA而言為32X4.7=150.4ns).所以 32位元的漣波進位加法器,可以在每秒執行大約1,500,000個加法運算(對於XC4005XL FPGA而言為每秒6,650,000個加法運

算).這筆個人電腦的微處理機慢了許多.那麼為什麼電腦的加法器會那麼快呢?

Full Adder

表 3－5　全加法器的真值表

輸入			輸出	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full Adder

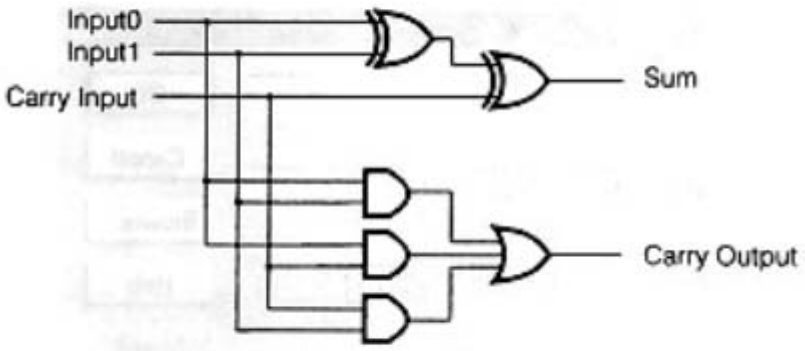
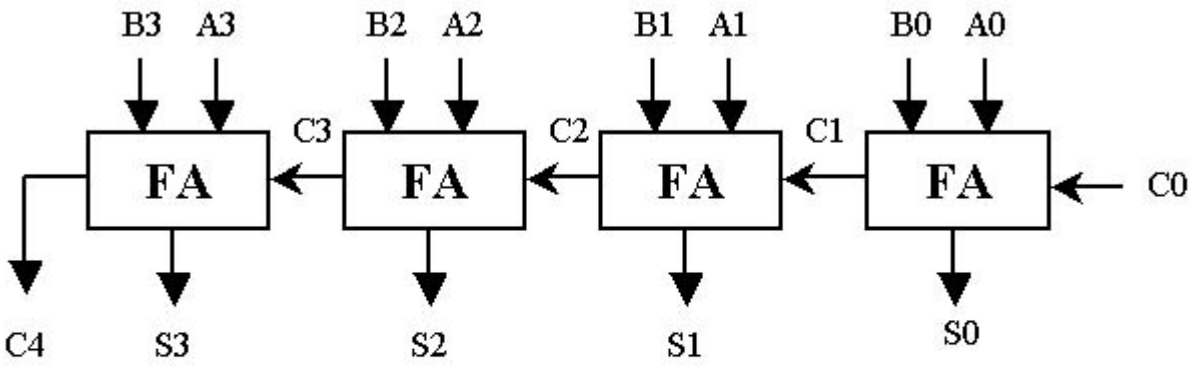
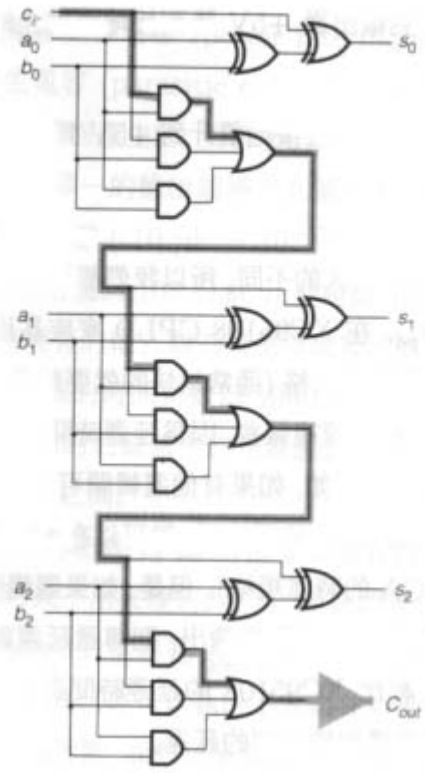


圖 3.11: 1 位元加法器的邏輯電路圖



Construct a 4-bit adder using four 1-adder modules

Fig5.7 ripple adder circuit delay

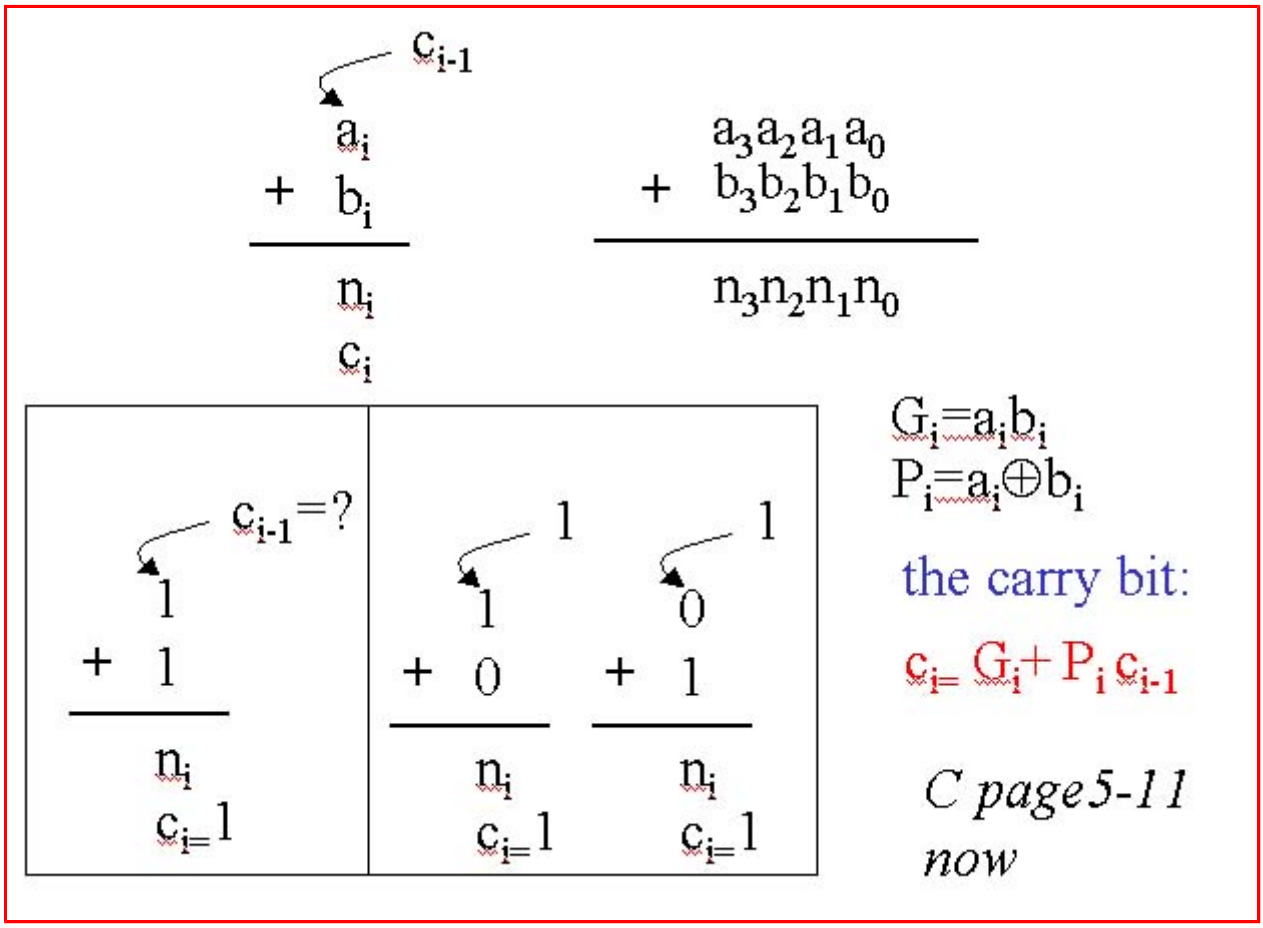


其中的一個答案就是使用進位前瞻電路(carry-lookahead circuit). 進位前瞻電路用平行的方式,同時計算數個進位位元.所以之前在漣波加法器的串聯式

延遲,在此就可以被省略了.它的方法如下所示:

Carry-lookahead circuit

如何找到carry bit 的規則
以減少電路之delay



分析電路

For each bit:

G_i、P_i本身 → (and、xor gate) 1 delay

$c_i = G_i + P_i c_{i-1}$ → (sum of product) 2 delay

加法運算 a_i、b_i、c_{i-1} →(xor) 1 delay

➔*Total 4 delay*

♣Tradeoff:

Waste many logic gates.

C page 5-12 table

下列等式是在 4 位元的進位前瞻加法器裡, 表示該如何去計算出 G_i 與 P_i。

$$c_0 = G_0 + P_0 c_{in}$$

$$c_1 = G_1 + P_1 c_0$$
$$c_1 = G_1 + P_1 (G_0 + P_0 c_{in})$$
$$c_1 = G_1 + P_1 G_0 + P_1 P_0 c_{in}$$

$$c_2 = G_2 + P_2 c_1$$
$$c_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 c_{in})$$
$$c_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_{in}$$

$$c_3 = G_3 + P_3 c_2$$
$$c_3 = G_3 + P_3 (G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_{in})$$
$$c_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_{in}$$

進位前瞻加法器的最大延遲是多少呢?它只需要一

個邏輯閘延遲的時間,就能把所有位元的G_i與P_i同時計算出來. 然後,再兩個邏輯閘延遲的時間,就可以將C_i的積項之和的等式計算出來(AND閘用去一個邏輯閘延遲時間以及OR閘用去另外一個). 所以,經過三個邏輯閘延遲的時間之後,就能將此加法器最後的進位計算出來.然而,要等全部進位位元相加起來,得到每個加法位元S_i 才算是完成了整個加法運算.所以,總共要四個邏輯閘延遲的時間,才能完成整個加法運算.

假設我們持續地重複之前推倒出來的等式,利用前瞻加法器建構一個 32位元的加法器時,此32位元的進位前瞻加法器所需的最大延遲時間一樣還是只需要四個邏輯閘延遲時間而已.而建構這樣的加法氣又需要多少邏輯閘呢?答案是只需要少數即可.(但是比漣波的方式多)

對於進位位元C_i,一個XOR閘必須用來計算P_i ,而在第i+2的AND閘必須與OR閘放在一起.(前題是,您可以找到能夠處理很大數目輸入的AND閘與OR閘.)所以每個進位位元需要i+4個邏輯閘.所以 ,用來處理N位元加法器,所使用到的全部邏輯閘數目是:

N(N-1)/2 + 4N

$$\sum_{i=0}^{N-1} i + 4 = \frac{N(N-1)}{2} + 4N$$

底下列出32位元漣波進位加法器和32位元進位前瞻加法器的比較.

	邏輯閘數目	邏輯閘延遲時間
32 位元漣波進位加法器	138	64
32 位元進位前瞻加法器	624	4

可以發現,若要速度快,付出的代價就是需要更多更複雜的邏輯閘來換取減少延遲時間.32位元進位前瞻全加器使用了大量的邏輯閘電路,所以進位前瞻全加器通常是使用4或8位元的邏輯單元來完成.每個區塊輸出的進位值再載入到下一個區塊中,然後再將進位前瞻加法器與漣波加法器的設計方法結合起來.

如果8位元的進位前瞻加法器被用在32位元的加法器中,則
此加法器最大的邏輯閘延遲為:
(32bits / 8bits) X 3 + 1 = 13 gate delays
以及在這加法電路所用到的邏輯閘數目為:
(32 bits / 8bits) X [8 X (8 - 1) / 2 + 4 X 8] gates = 240 gates

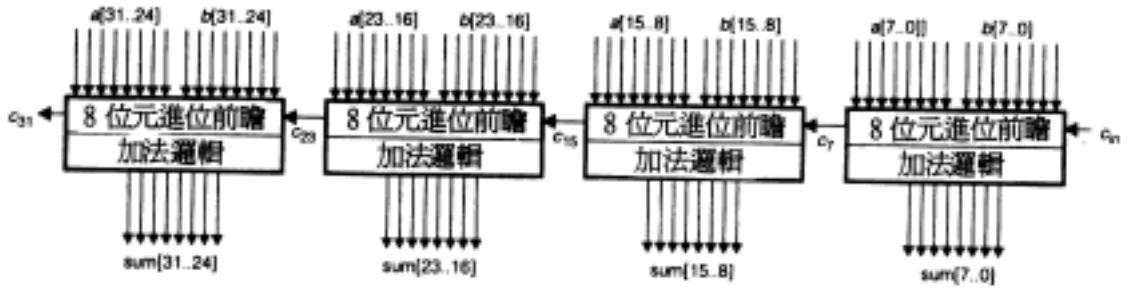


圖 5.8: 將 8 位元進位前瞻加法器, 使用在 32 位元的加法器