3

第三章

組合邏輯

數位邏輯電路主要分為組合邏輯(Combinational Logic)與順序邏輯(Sequential Logic)。「組合邏輯」的輸出只與其當時的輸入有關，「順序邏輯」的輸出除了與其當時的輸入有關外，也與過去的輸入及輸出有關。組合邏輯電路主要是由基本邏輯閘及輸出入變數所組成，以下各節將分別介紹組合邏輯之基本及應用電路。

3-1 基本邏輯閘

數位電路可由各種邏輯閘所組成，瞭解了各種基本邏輯閘的功能後，便可藉由基本閘來設計各種電路，以下我們將分別介紹的基本邏輯閘有及閘(AND gate)、或閘(OR gate)、反閘(NOT gate)、反及閘(NAND gate)、反或閘(NOR gate)、互斥或閘(XOR gate)、互斥反或閘(XNOR gate)。

3-1-1 二輸入 AND 閘

由真值表可知輸入端有兩隻腳 A、B，輸出端有一隻腳 Y，當 AB 輸入為 00,01,10,11 時，輸出 Y 分別為 0,0,0,1。及閘的意義為所有輸入端皆為 1 時，其輸出端方為 1，否則為 0。

布林函數：

Y=AB

符號：

真值表

輸入		輸出
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

圖 3-1-1

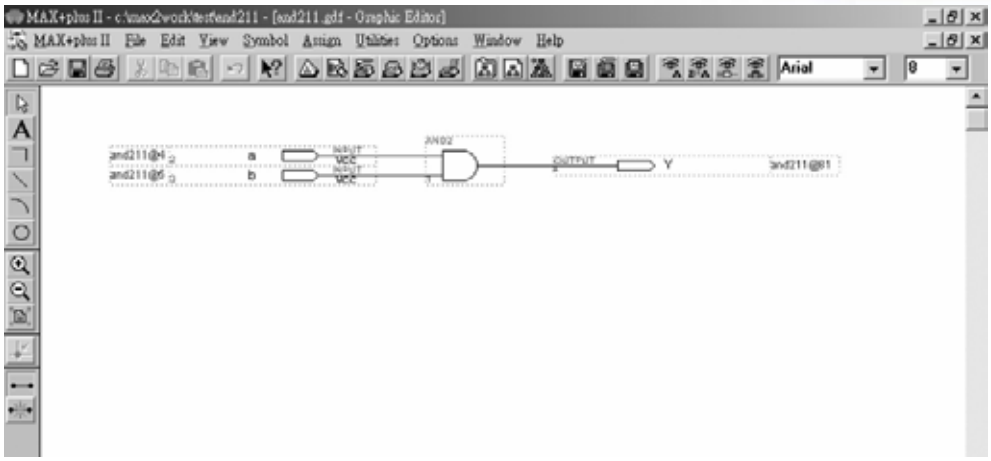

圖 3-1-2　**AND 閘**

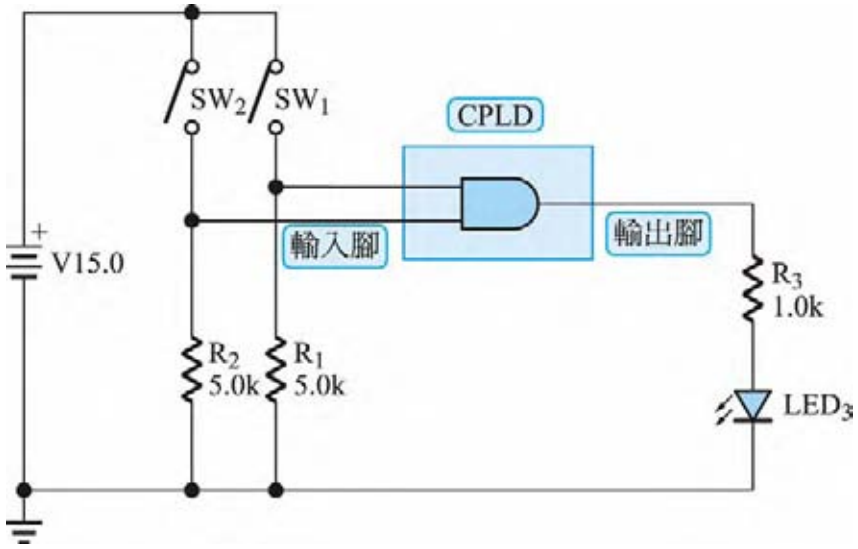

圖 3-1-3　AND 閘實驗電路圖

3-1-2 二輸入 OR 閘

由真值表可知輸入端有兩隻腳 A、B，輸出端有一隻腳 Y 當 AB 輸入為 00,01,10,11 時，輸出 Y 分別為 0,1,1,1。或閘的意義為只要有一輸入端為 1 時，其輸出端即為 1。

布林函數：

$Y=A+B$

符號：

真值表

輸入		輸出
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

圖 3-1-4

3-1-3 NOT 閘

由真值表可知輸入端有一隻腳 A，輸出端有一隻腳 Y，當 A 輸入為 0,1 時，輸出 Y 分別為 1,0。反閘的意義為輸出端的值與輸入端相反。

布林函數：

$Y=\overline{A}$

符號：

真值表

輸入	輸出
A	Y
0	1
1	0

圖 3-1-5

3-1-4 二輸入 NAND 閘

由真值表可知輸入端有兩隻腳 A、B，輸出端有一隻腳 Y，當 AB 輸入為 00,01,10,11 時，輸出 Y 分別為 1,1,1,0。反及閘的意義為所有輸入端皆為 1 時，其輸出端方為 0，否則為 1。亦可說只要有一輸入為 0，輸出即為 1。

布林函數：

$Y=\overline{AB}$

符號：

真值表

輸入		輸出
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

圖 3-1-6

3-1-5 二輸入 NOR 閘

由真值表可知輸入端有兩隻腳 A、B，輸出端有一隻腳 Y，當 AB 輸入為 00,01,10,11 時，輸出 Y 分別為 1,0,0,0。反或閘的意義為只要有一輸入為 1，輸出即為 0，否則為 1。亦可說所有輸入端皆為 0 時，其輸出端方為 1。

布林函數：

$Y=\overline{A+B}$

符號：

真值表

輸入		輸出
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

圖 3-1-7

3-1-6 二輸入 XOR 閘

由真值表可知輸入端有兩隻腳 A、B，輸出端有一隻腳 Y，當 AB 輸入為 00,01,10,11 時，輸出 Y 分別為 0,1,1,0。互斥或閘的意義為輸入端的接腳中 1 的個數為奇數時，其輸出端為 1，反之為 0。

布林函數：

$Y=A\oplus B=\overline{A}B+A\overline{B}$

符號：

真值表

輸入		輸出
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

圖 3-1-8

3-1-7 二輸入 XNOR 閘

由真值表可知輸入端有兩隻腳 A、B，輸出端有一隻腳 Y，當 AB 輸入為 00,01,10,11 時，輸出 Y 分別為 0,0,0,1。互斥反或閘的意義為輸入端的接腳中 1 的個數為奇數時，其輸出端為 0，反之為 1。

布林函數：

$Y=A\odot B=\overline{AB}+AB$

符號：

真值表

輸入		輸出
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

圖 3-1-9

3-2 加法器

我們要設計數位電路來幫我們解決問題，可由算數運算開始，其中加法運算可謂最基本計算。二進制的加法運算為 0+0=0、0+1=1、1+0=1、1+1=10；其中 1+1=10 中的 10 等於十進制中的 2，因二進制只有 0、1 兩種運算元，所以要表示 2 時需有兩個位元表示，較高位元為進位(Carry)，較低位元為和(Sum)。兩個 1 位元的變數的相加，輸入端有兩個變數，產生一個和與進位輸出的電路結

構，稱為半加器(Half Adder; HA)。若在兩個變數相加時，即考慮到上一個位元進位的問題，則輸入端將有三個變數相加(即相加的兩數和上一位元的進位)，同樣的產生一個和與一個進位的輸出，此種電路結構稱為全加器(Full Adder, FA)。

3-2-1 半加器

兩個變數的相加，用半加器即可完成，真值表如下所示，輸入數值有 A、B 值，分別代表相加的兩數，S 表示和，C 表示進位。

真值表

輸入		輸出	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

電路圖

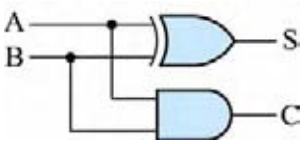

圖 3-2-1

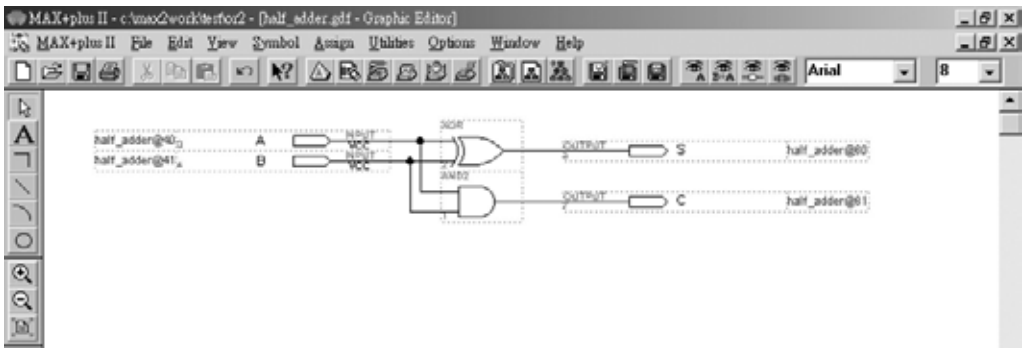

圖 3-2-2 半加器

如圖 3-2-2 所示，經由步驟：畫電路圖→存檔→設為工作專案→指定 CPLD 晶片→編譯→規劃腳位→編譯→下載燒錄(若是非 ALTERA 的晶片下載前需轉檔) 後，即完成電路設計製作，此時可配合外部電路(或實驗器模組，需配合腳位規劃)來驗證。

表 3-2-1　電路圖輸出入腳位、CPLD 腳位及實驗器模組腳位對應表

腳位對應關係	輸入		輸出	
名　　稱	被加數(A)	加數(B)	和(S)	進位(C)
CPLD 晶片腳位	PIN 9	PIN 10	PIN 80	PIN 81
實驗器模組對應腳位	DIPB1	DIPB2	DG6(LED)	DG7(LED)

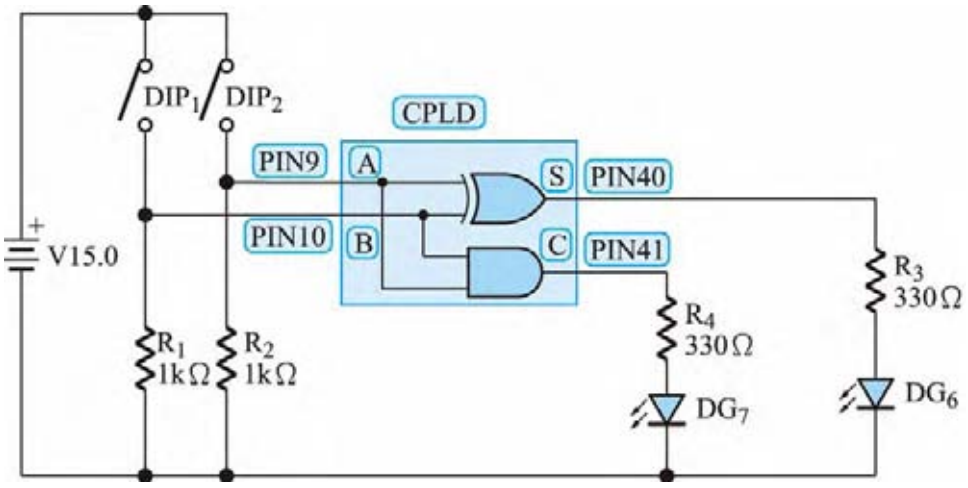

圖 3-2-3　半加器實驗電路圖

3-2-2　全加器(一)

三個變數的相加，需用全加器才可完成，一般是指相加的兩數之外，尚有上一個位元的進位值。真值表如表 3-2-2 所示，輸入方面有 A、B 分別表示相加的兩數，Ci 表示上一個位元的進位數值，輸出方面有 S 表示和，Co 表示進位。

表 3-2-2　真值表

輸入			輸出	
A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

由真值表可知 S 與 Co 之布林函數分別如下：

$$S = AB\overline{Ci} + A\overline{B}Ci + \overline{A}BCi + \overline{A}\overline{B}\overline{Ci}$$

$$Co = \overline{A}BCi + A\overline{B}Ci + AB\overline{Ci} + ABCi$$

上述是指未化簡前由真值表寫成的布林函數，在 MAX+PLUS Ⅱ 中，圖形編輯時不管有無化簡，將其畫成電路圖後皆可正常編繹，只要布林式正確，電路即可正常工作，如圖 3-2-4 所示，全加器之 S 不化簡，Co 化簡，完成之電路仍然可以正常動作。(MAX+PLUS II 會主動化簡電路，所以繪圖時不用化簡電路，反而可以增加可讀性)

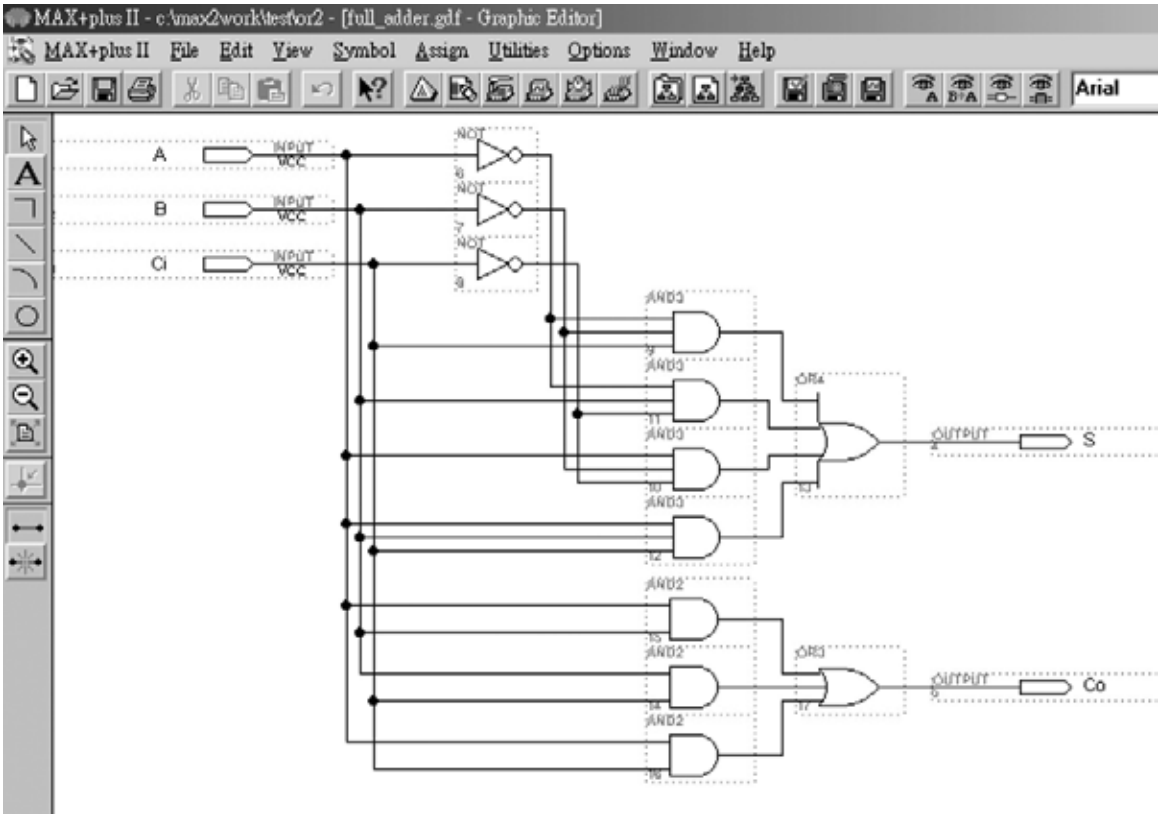

圖 3-2-4　全加器電路圖(一)

3-2-3　全加器(二)(以製作元件方式完成)

三個變數的相加並非一定要用上述做法的全加器來完成，也有另外一種做法，就是先將兩數相加之後，其結果再與另一個變數相加。即我們可以以半加器的組合來完成全加器的功能。

如圖 3-2-5 所示，規劃全加器時，以兩個半加器的組合來完成全加器功能，由真值表的驗證可知此電路完全符合，輸出結果與上圖全加器(一)完全相同。

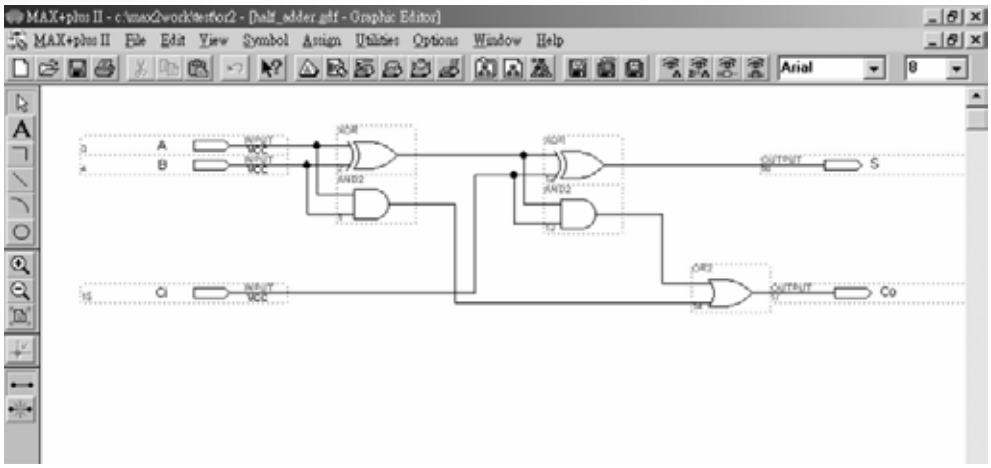

圖 3-2-5　全加器電路圖(二)

若要用半加器完成全加器之功能，而電路圖的畫法是直接畫兩個半加器，則這種以半加器完成全加器可謂意義不大。在 MAX+PLUS Ⅱ中，提供一種製造元件(symbol)的方法，可將一些我們平時會重覆用到的電路，變成單一個元件(symbol)，則我們在使用之時，只要叫出該元件(symbol)，即可直接使用，以下將以半加器為例，介紹這種方法。

製造半加器元件(symbol)：

1 開啟一個新圖形編輯檔，繪製一個半加器的電路圖，如圖 3-2-6 所示。

2. 存檔，取檔名為 half_adder.gdf。

3. 編譯。

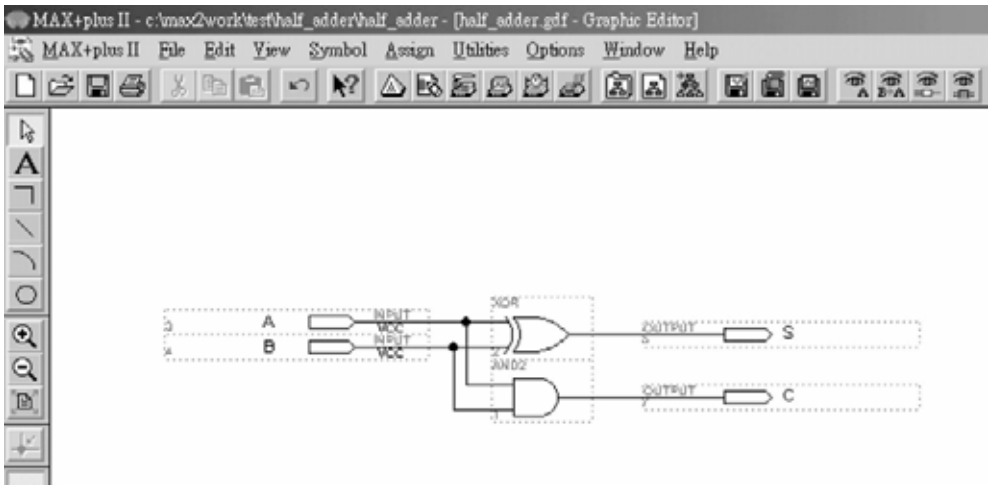

圖 3-2-6　半加器

4. 產生符號檔，點選 File → Create Default Symbol，即可產生新的元件 IC(半加器，名稱 half_adder)。

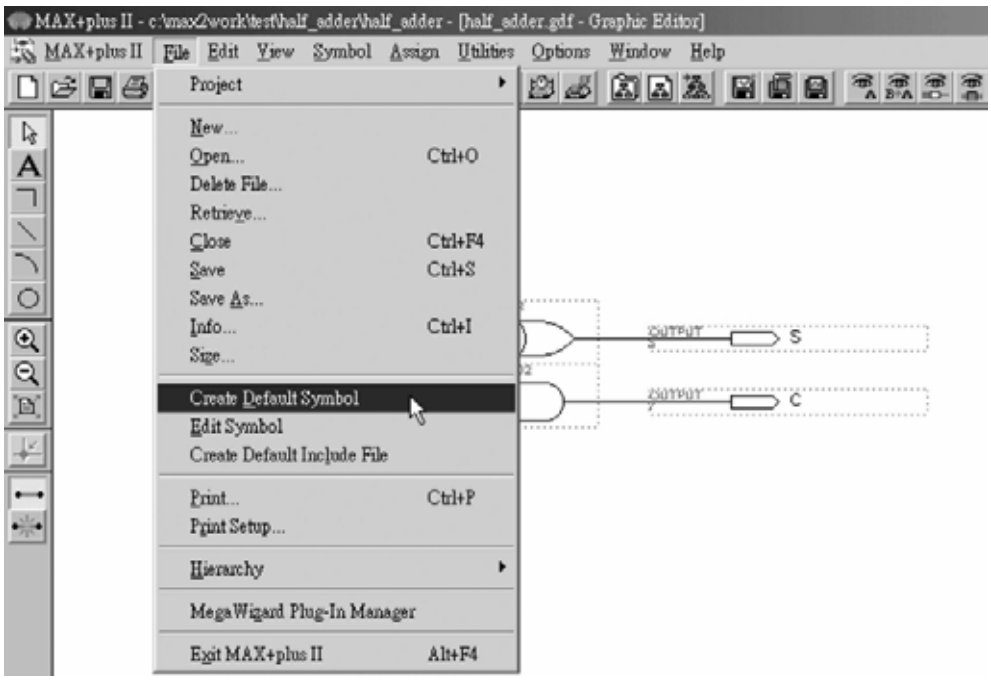

圖 3-2-7　執行製作自製元件視窗

5. 開啟新的圖形編輯檔，此時即可取用所製作的元件(half_adder.sym)，點選 Symbol→Enter Symbol (或在編輯視窗按兩下 Double Click)，若出現如圖 3-2-9 之錯誤訊息時不要理它，它只是說尚定義零件插入點而已，按確定後，即出現圖 3-2-10 零件取用視窗。

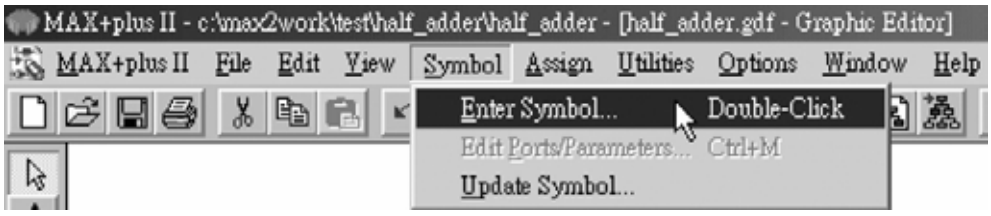

圖 3-2-8　插入元件視窗

圖 3-2-9　錯誤訊息

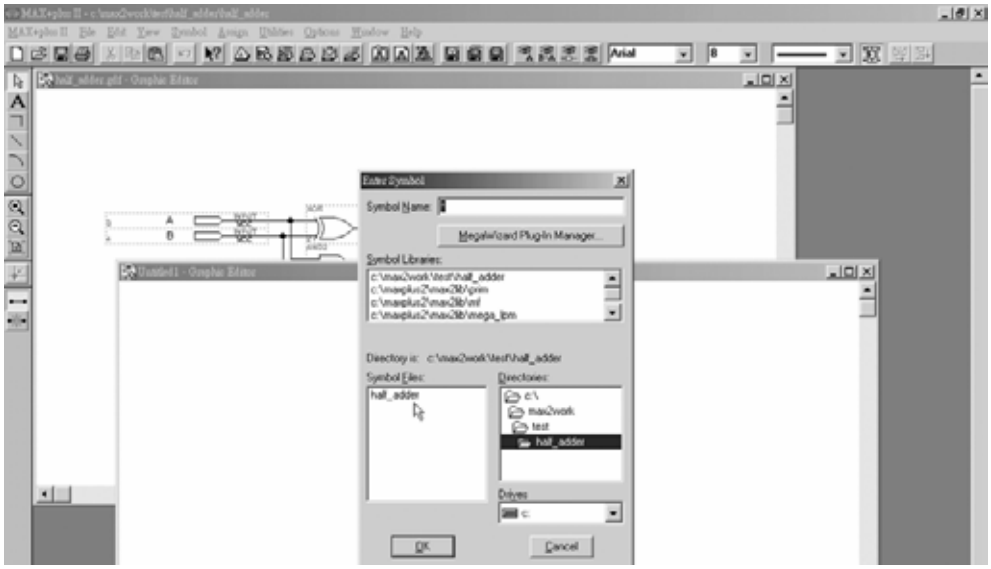

圖 3-2-10　元件取用視窗

6. 取用半加器元件(symbol)：進入剛才存放半加器的資料夾，點選符號元件檔(Symbol File) half_adder，按 OK，在圖形編輯視窗中，即可看到該元件(包裝後的半加器)。可在半加器元件上按兩下，即可開啟其內部電路視窗。

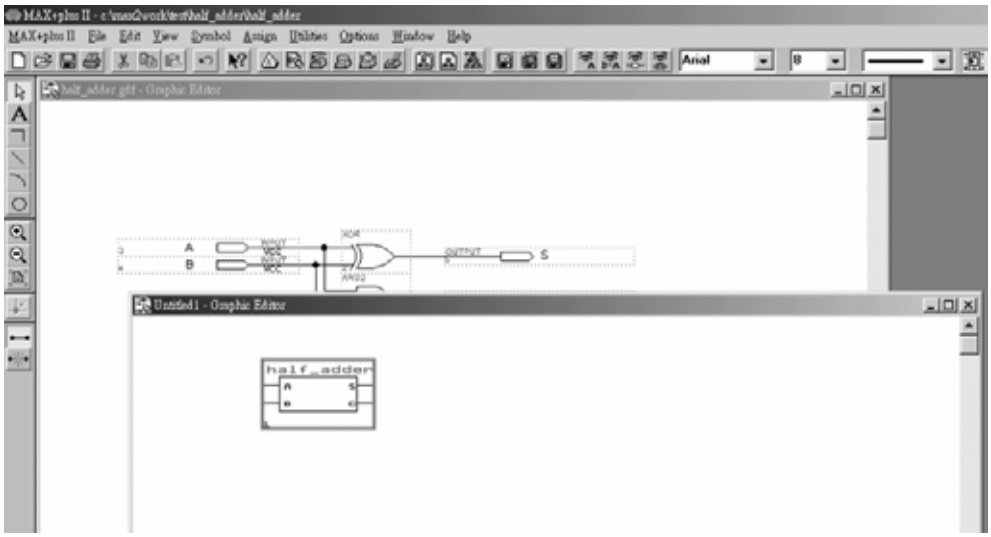

圖 3-2-11　自製半加器元件

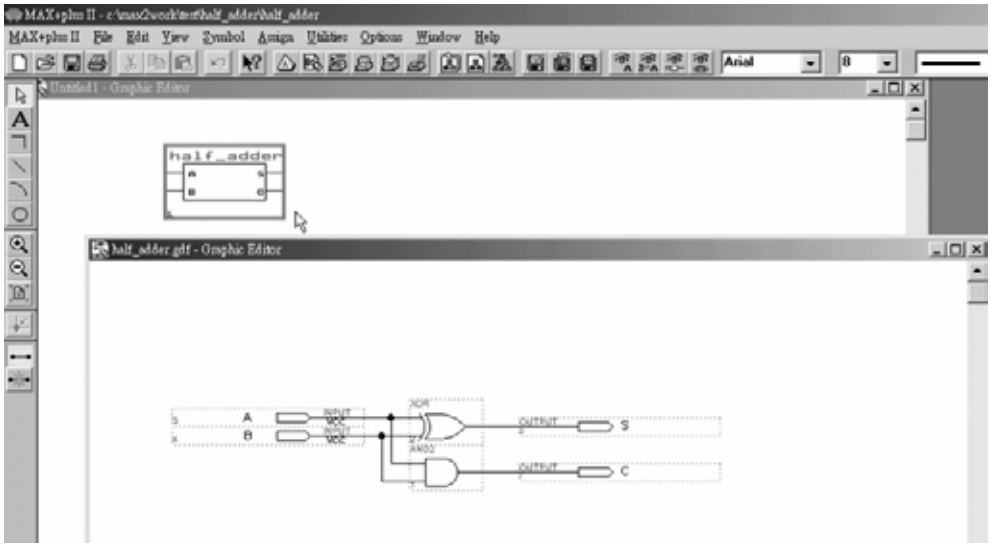

圖 3-2-12　半加器自製元件及內部電路

設計半加器元件(symbol)完成後，雖可叫出包裝起來的半加器(half_adder)，但卻不知其功能是否正常如一般之半加器電路，此時可用模擬功能測試一下。

模擬步驟如下：

1. 在新的編輯檔案中，叫出該元件，並叫出輸入輸出接腳且連接好線路後存檔(Save As)，設為指定工作專案(File→Project→Set Project to Current File)，指定 CPLD 晶片(Assign→Device)，並編譯 (MAX+PLUS Ⅱ→ Compiler)。

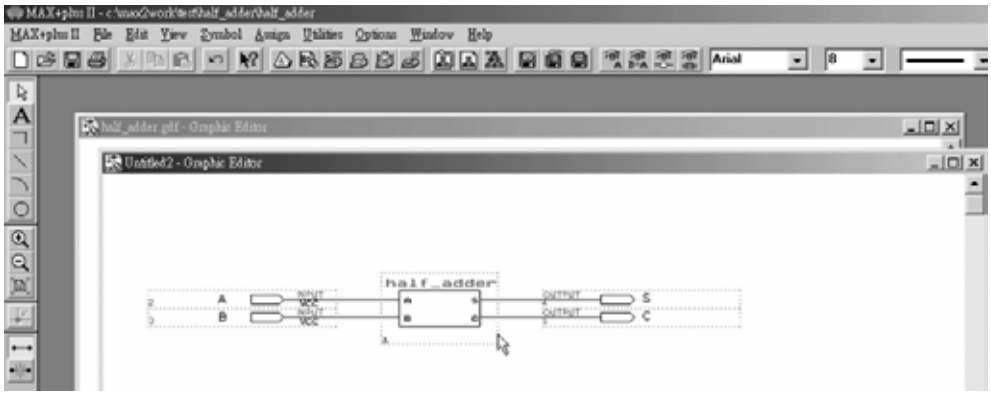

圖 3-2-13　半加器自製元件電路

2. 開啟新的波形編輯檔案，設定功能模擬結束時間(File→End Time)，設定格線間距(Options→Grid Size)，顯示在視窗中適當大小格線(View→Fit in Window)。

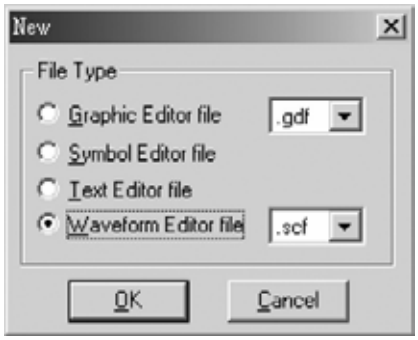

圖 3-2-14　開啟新檔視窗

圖 3-2-15　模擬結束時間設定視窗

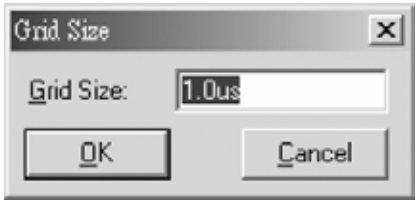

圖 3-2-16　模擬單位時間設定視窗

圖 3-2-17　視窗調整

3. 儲存檔案(Save As)，輸入節點(Node→Enter Nodes from SNF，按 List 及 =>，OK)，編輯輸入時脈( )。

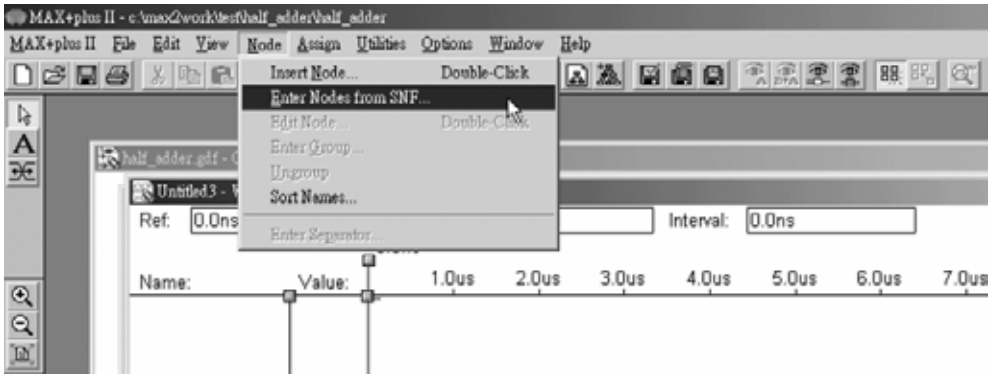

圖 3-2-18　執行輸出入節點選擇視窗

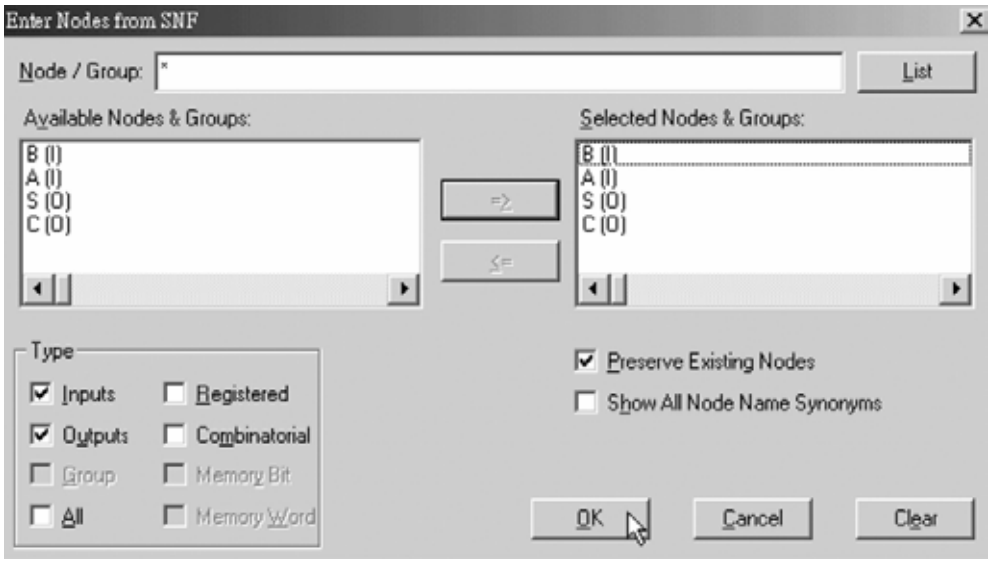

圖 3-2-19　輸出入節點選擇視窗

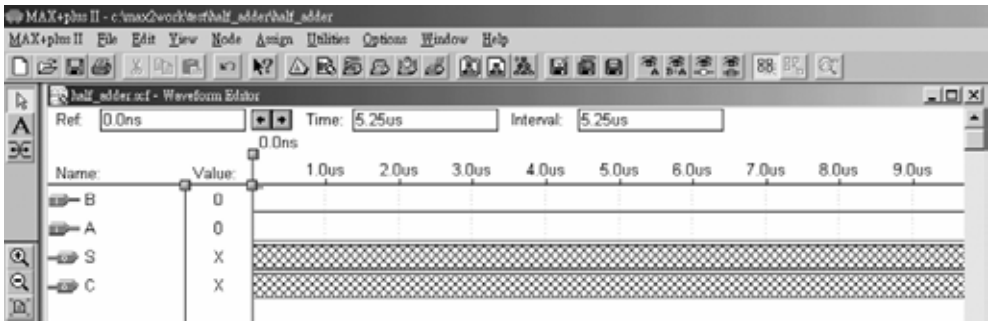

圖 3-2-20　模擬波形設定

4. 點選 B，再點選，設定波形經每 1 個 grid size 轉態一次(Multiplied By = 1)；
點選 A，再點選，設定波形經每 2 個 grid size 轉態一次(Multiplied By = 2)。

圖 3-2-21　時脈信號設定視窗

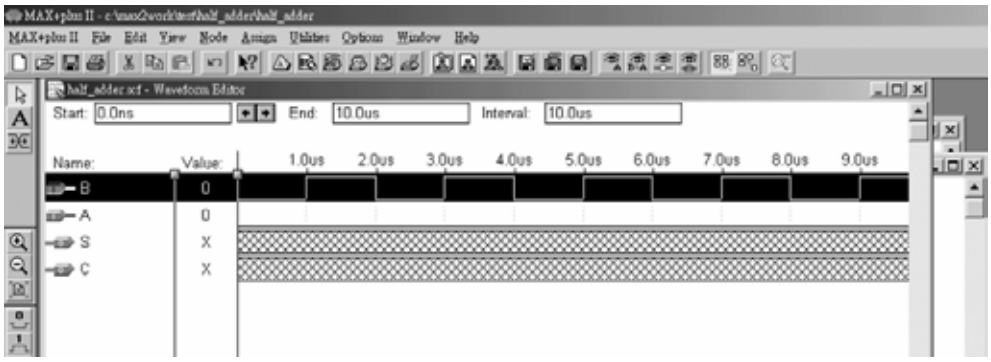

圖 3-2-22　模擬波形設定

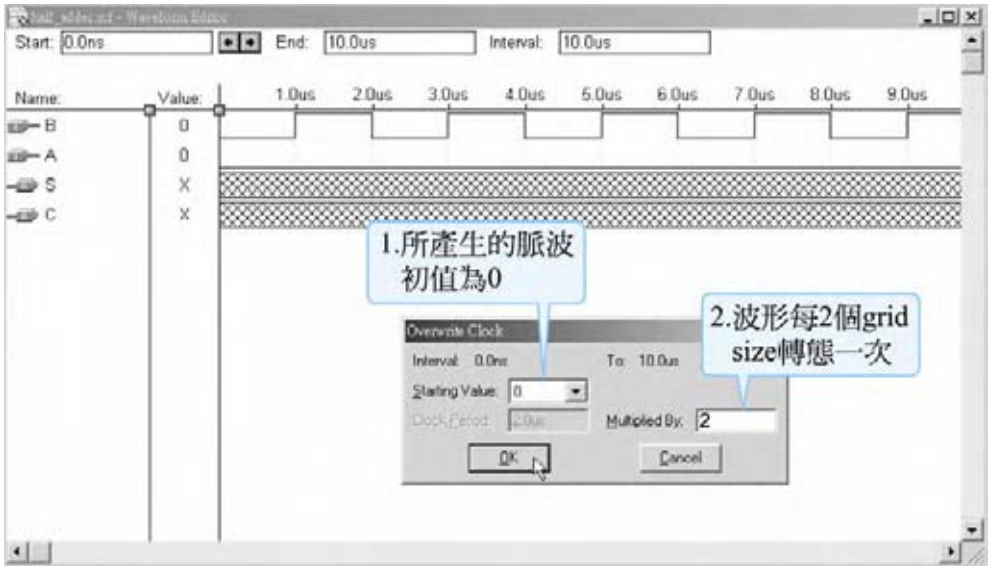

圖 3-2-23　時脈信號設定視窗

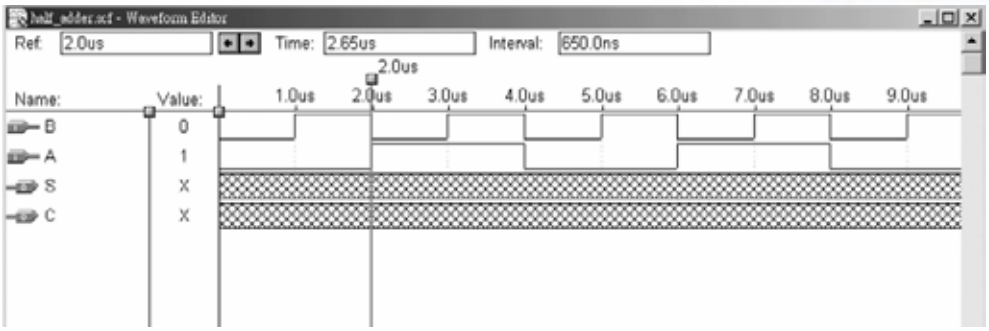

圖 3-2-24　模擬波形設定

5. 執行模擬 MAX+PLUS Ⅱ→Simulator，Start 如圖 3-2-26 所示，模擬沒有錯誤及警告，所得波形模擬結果符合半加器元件，代表我們製作的自製元件是正確可用的。

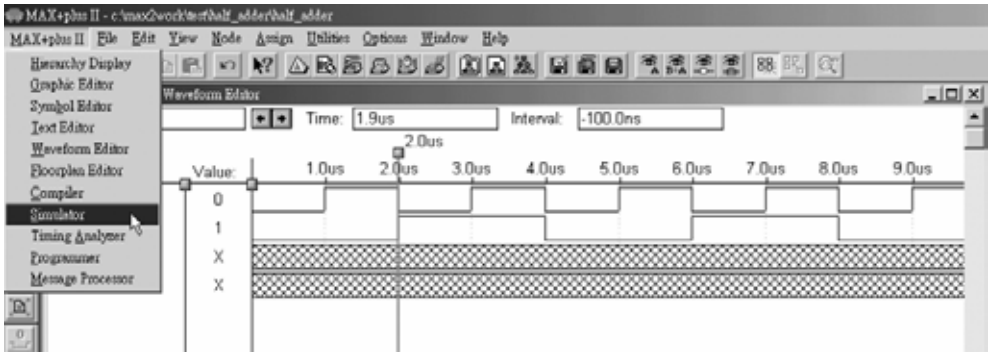

圖 3-2-25　執行模擬

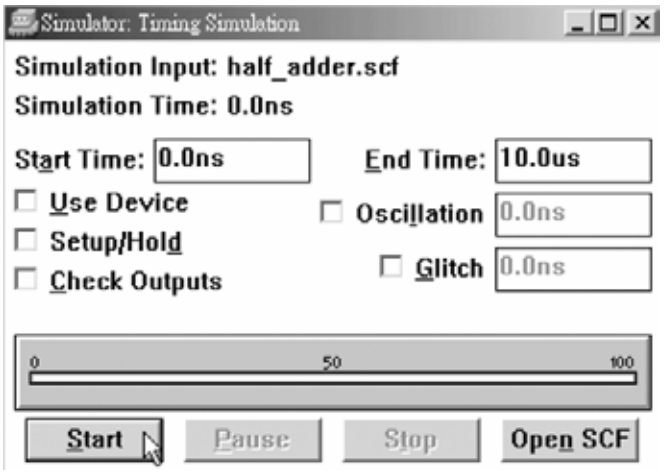

圖 3-2-26　模擬起始視窗

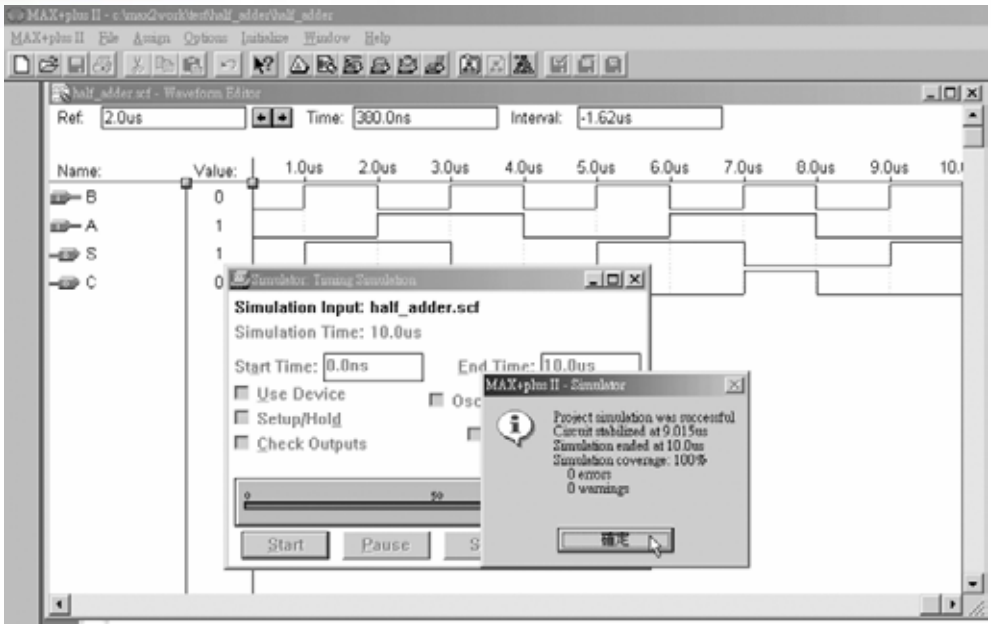

圖 3-2-27　模擬訊息視窗

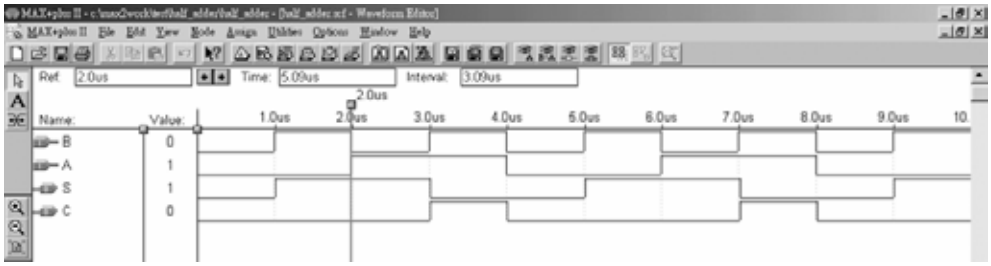

圖 3-2-28　模擬結果

做完半加器元件(symbol)，再以半加器元件的組合來完成全加器功能，如圖 3-2-29 所示，讀者可自行繪圖及模擬功能是否正常。

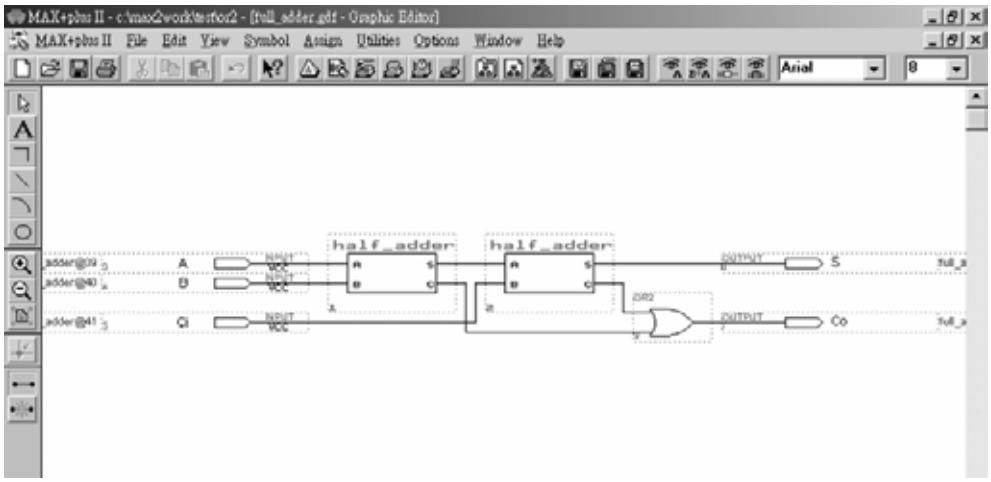

圖 3-2-29　以半加器元件完成全加器電路圖

3-2-4 全加器三(以半加器自製元件組成全加器自製元件)

完成了用半加器元件做成全加器後，可以再將這樣的電路完成另外一個全加器的元件，這樣以後需要用到全加器時，只需再將這樣的元件叫出來，即可使用。因為在用 MAX+PLUS Ⅱ 設計電路時，其編譯過程會產生很多檔案，檔案與檔案之間有很多連結關係，所以如果要用既有的自製元件完成另一個自製元件時，要特別注意一件重要的事，就是要將那些既有的自製元件的圖形檔(graphic editor file,*.gdf)以及元件檔(symbol editor file, *.sym)複製到一個新的資料夾，然後在該資料夾上儲存新的自製元件，如此既可簡單的分類自製元件檔案，且可避免那些舊有自製元件的檔案在產生新檔案時，造成連結上的錯誤。

要以自製的半加器元件(symbol)做出全加器元件時，步驟如下：

1. 建立一個新資料夾，取名 full_adder3，複製事先製妥半加器元件的圖形檔(half_adder.gdf)以及元件檔(half_adder.sym)至該資料夾。
2. 執行 MAX+PLUS Ⅱ 軟體，開啟新的圖形編輯檔，叫出儲存 full_adder3 資料夾的半加器元件檔 half_adder.sym，利用它配合邏輯閘完成全加器繪圖，存檔於 full_adder3 資料夾內。如圖所示，在此取名全加器元件圖形檔案名稱為 full_adder。

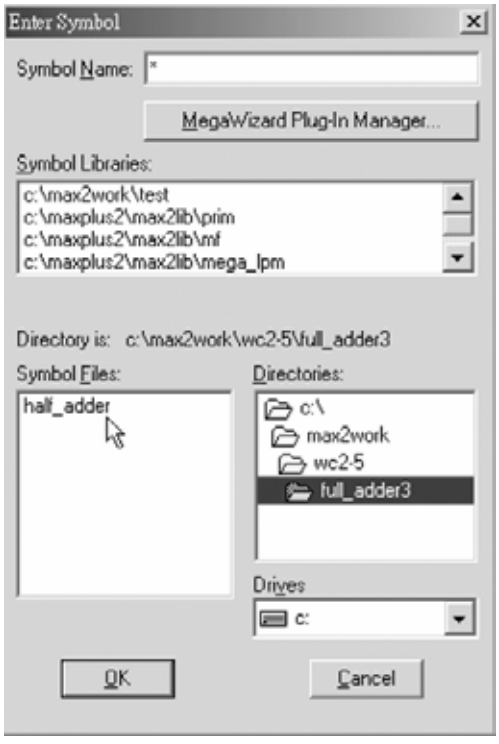

圖 3-2-30 零件取用視窗

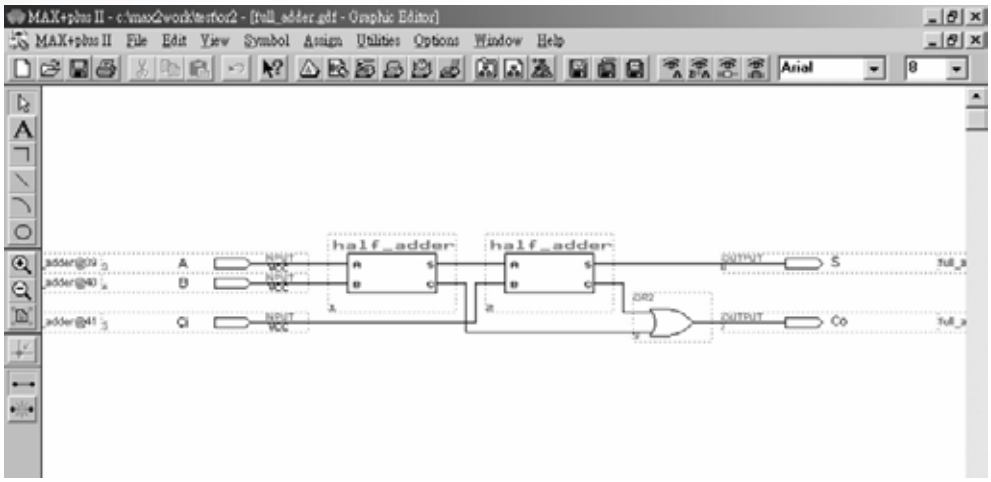

圖 3-2-31　半加器自製元件組成全加器電路圖

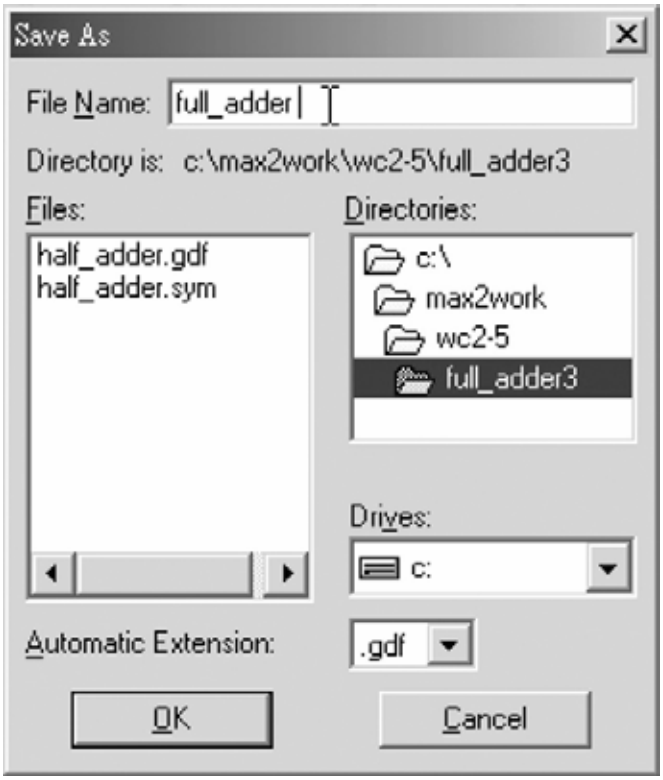

圖 3-2-32　儲存檔案視窗

3. File→Project→Set Project to Current File。

4. 編譯。

5. 產生符號檔，點選 File → Create Default Symbol，即可產生新的元件(symbol)(全加器，名稱 full_adder)(檔名：full_adder.sym)。

6. 開啟新的圖形編輯檔，此時即可取用所製作的元件(symbol) (half_adder)，點選 Symbol → Enter Symbol(或在編輯視窗按兩下 Double Click)，按確定後，

即出現零件取用視窗，選擇剛才所建立之新資料夾 full_adder3 即可發現已產生一個新的元件檔 full_adder.sym，選擇此元件即可在圖形編輯視窗中使用它。

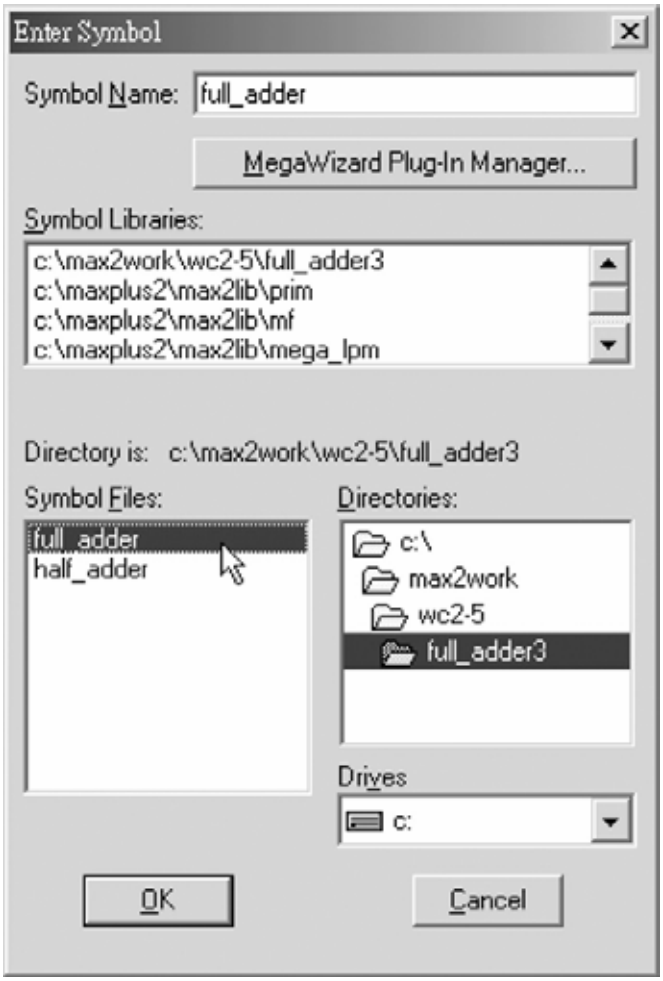

圖 3-2-33　取用元件視窗

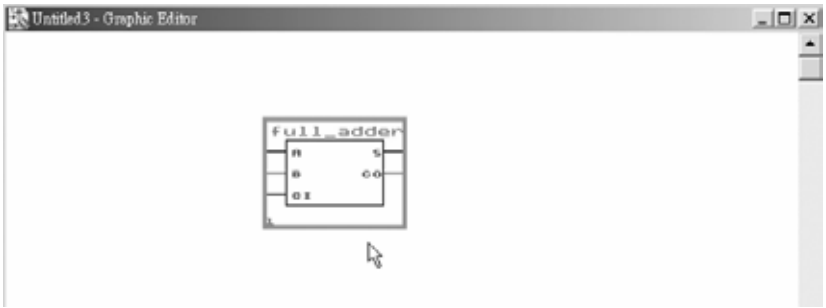

圖 3-2-34　全加器自製元件

設計全加器元件(symbol)完成後，即可叫出包裝起來的全加器(half_adder)，要知道其是否可正常執行，可執行模擬功能測試。

模擬步驟如下：

1. 在新的編輯檔案中，叫出該元件(symbol)，並叫出輸入輸出接腳且連接好線路後存檔(Save As)，如圖 3-2-36 儲存在 full_adder3 資料夾，檔名 full_adder3.gdf。設為指定工作專案(File→Project→Set Project to Current File)，指定 CPLD 晶片(Assign→Device)，並編譯(MAX+PLUS Ⅱ→ Compiler)。

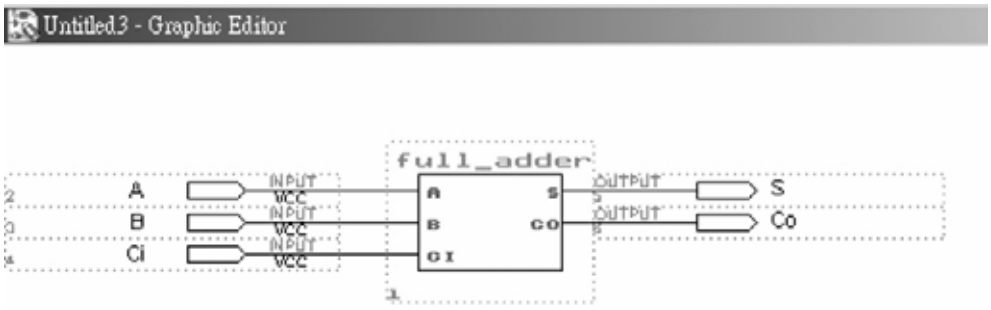

圖 3-2-35 全加器自製元件電路圖

圖 3-2-36 另存新檔視窗

2. 開啟新的波形編輯檔案，設定功能模擬結束時間(File→End Time)，設定格線間距(Options → Grid Size)，顯示在視窗中適當大小格線(View→Fit in Window)。

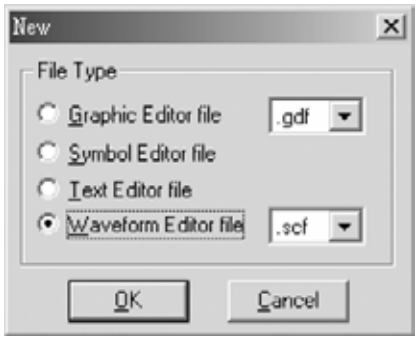

圖 3-2-37　開啟新檔視窗

圖 3-2-38　模擬結束時間設定視窗

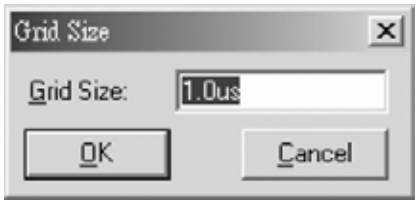

圖 3-2-39　模擬單位時間設定視窗

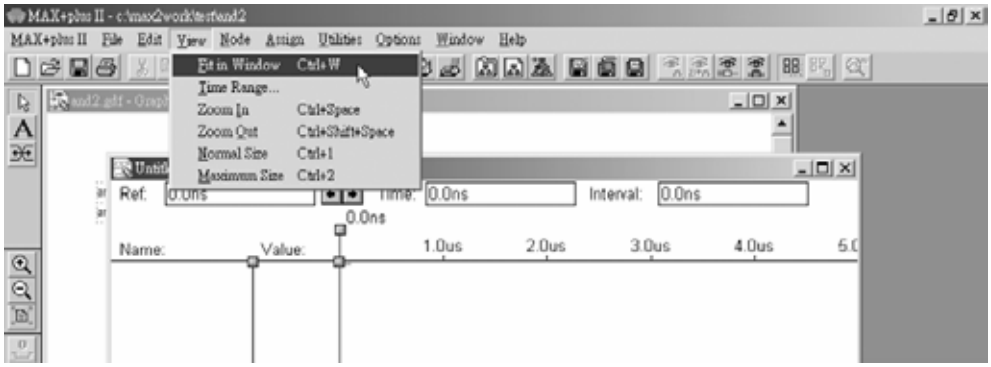

圖 3-2-40　視窗調整

1. 儲存檔案(Save As)，檔名 full_adder3.scf。輸入節點(Node → Enter Nodes from SNF，按 List 及 =>，OK)，編輯輸入時脈( )。

圖 3-2-41　儲存檔案視窗

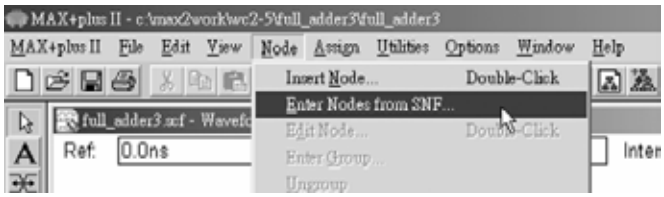

圖 3-2-42　執行輸出入節點選擇視窗

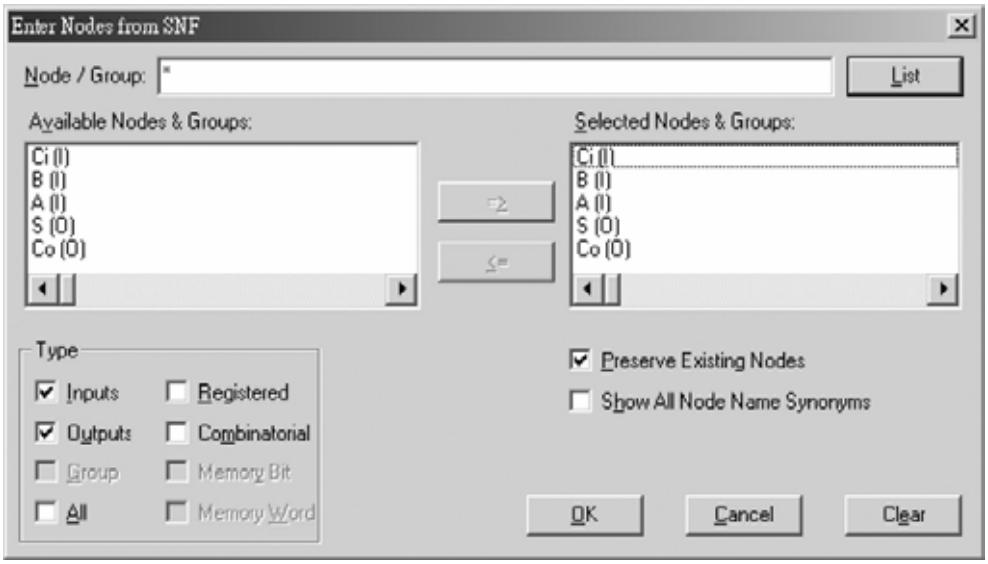

圖 3-2-43　輸出入節點選擇視窗

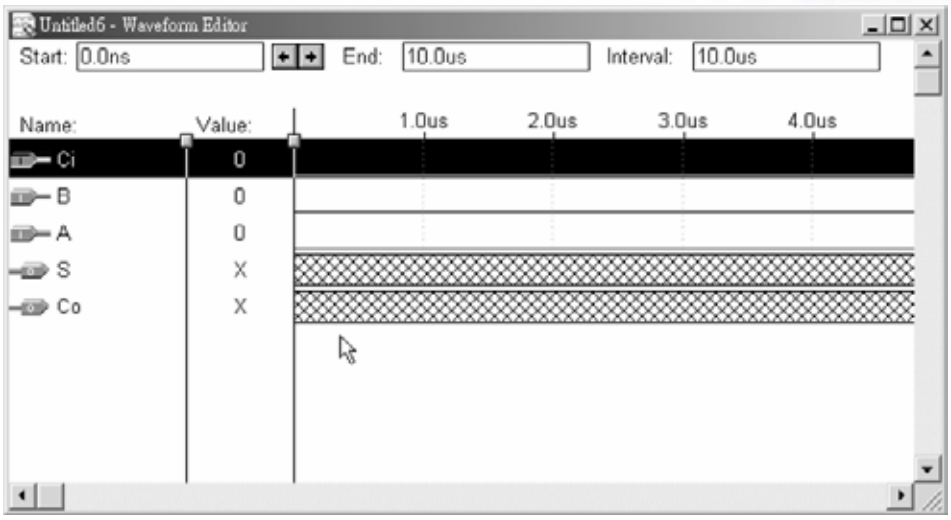

圖 3-2-44　模擬波形設定

4. 點選 Ci，再點選，設定波形經每 1 個 grid size 轉態一次(Multiplied By = 1)；
點選 B，再點選，設定波形經每 2 個 grid size 轉態一次(Multiplied By = 2)；
點選 A，再點選，設定波形經每 4 個 grid size 轉態一次(Multiplied By = 4)

圖 3-2-45　時脈信號設定視窗

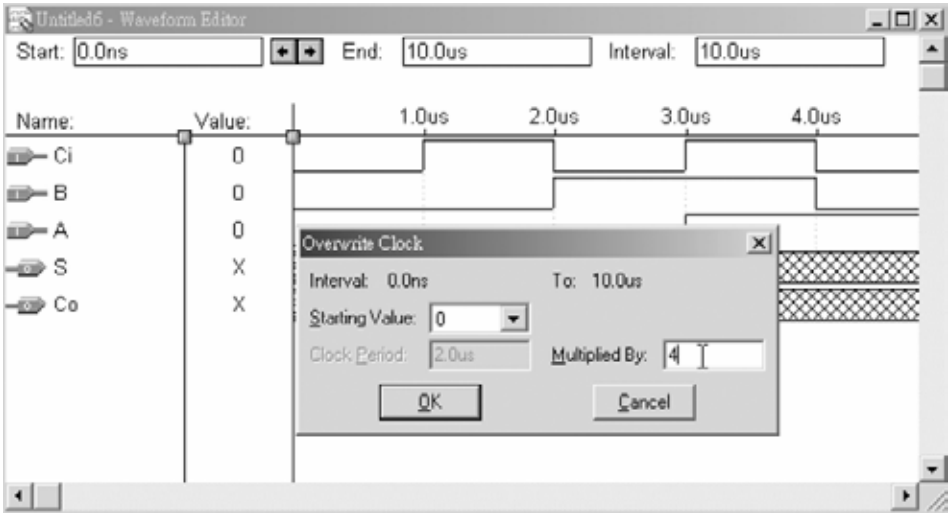

圖 3-2-46　時脈信號設定視窗

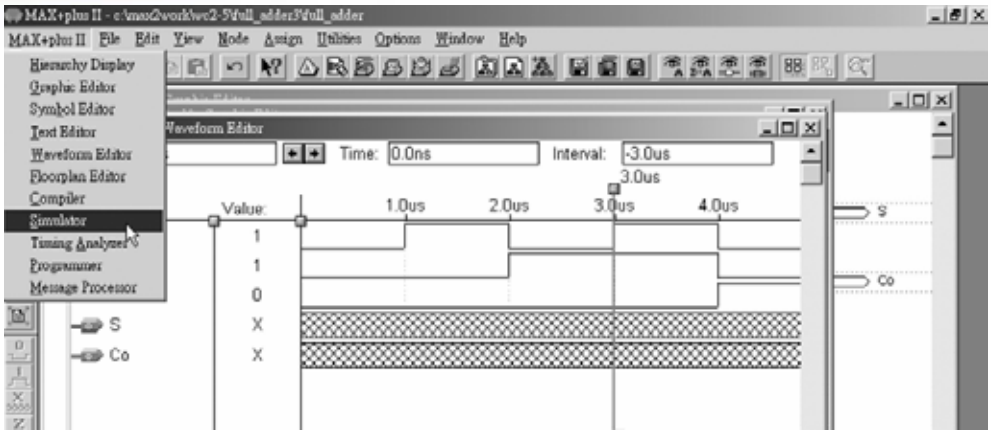

圖 3-2-47　執行模擬

5. 執行模擬(MAX+PLUS II → Simulator，按下 Start)如圖 3-2-48 所示，模擬沒有錯誤及警告，所得波形模擬結果符合全加器元件，代表我們製作的元件(symbol)是正確可用的。

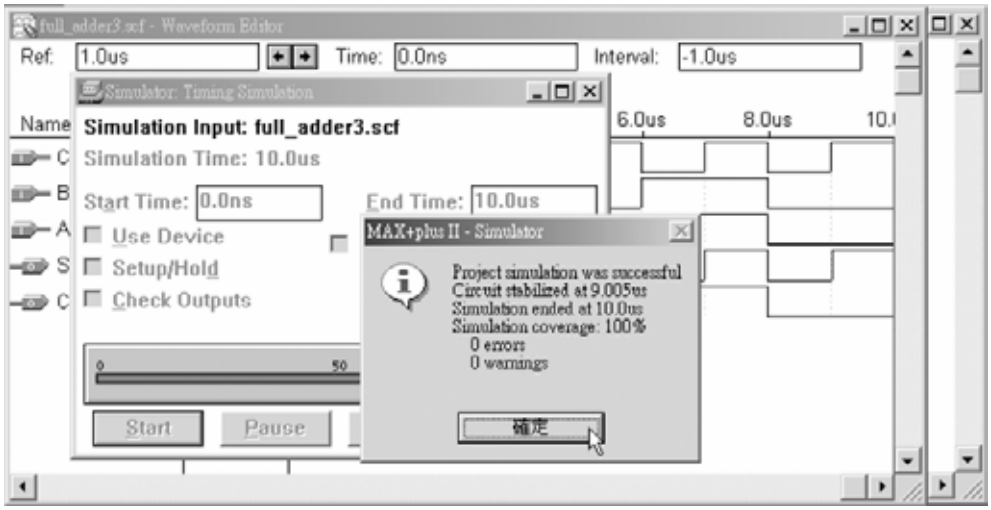

圖 3-2-48　模擬訊息視窗

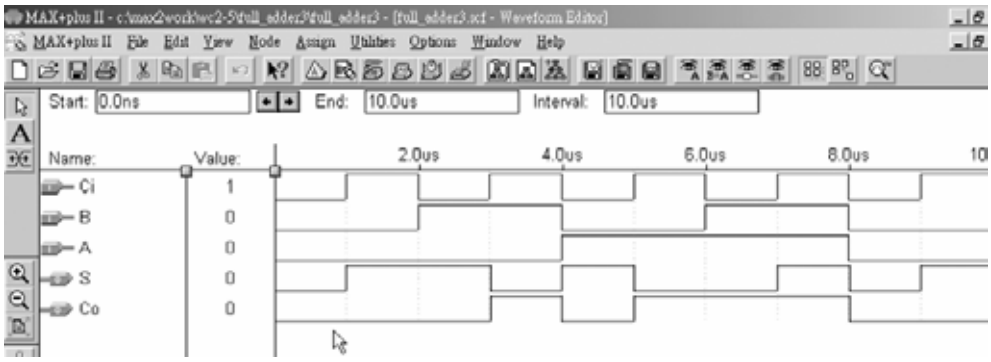

圖 3-2-49　模擬結果

3-2-5 四位元加法器

一般做加法工作時，通常是多個位元相加，若是單純用邏輯閘電路要設計這樣的電路，那麼完成的電路圖肯定相當大，也許用 CPLD 來完成電路可減少焊接的工作，但是光要繪圖就相當複雜了。所以像這種有重覆性質的電路功能，要先做單一的元件，再重覆叫出這些元件來組成電路，這樣才簡單且迅速。在此我們將以上一單元做成的全加器元件來完成四位元全加器的製作。

$$\begin{array}{r} C_3C_2C_1C_0 \\ A_3A_2A_1A_0 \\ +\quad B_3B_2B_1B_0 \\ \hline C_3S_3S_2S_1S_0 \end{array}$$

四位元加法器運算如上所示，A_0~A_3 表示被加數四個位元，B_0~B_3 代表加數的四個位元，其相加產生的進位 C_0~C_3 分別加入其次一位元的運算中，產生的結果和為 S_0~S_3 以及最後的進位 C_3。依照如此原則，要以全加器元件製作四位元加法器，第一個全加器的被加數及加數分別接 A_0 及 B_0，其進位輸入端接地(因為第一位元並無上一級的進位，所以其值為 0)。第一個全加器的 S(和)輸出端直接是第一位元 S_0(和)的輸出值，進位輸出端 C_0 則接到第二個加法器的進位輸入端。第二個全加器則接被加數 A_1 及加數 B_1，產生和(S_1)輸出以及進位(C_1)接到第三個加法器，以此類推接完四個加法器。

要使用既有的自製元件來完成新電路，建議讀者建立新資料夾來儲存，製作步驟如下：

1. 建立新資料夾，取名 4BITS_ADDER，並複製上一單元製作全加器元件所完成繪圖檔(full_adder.gdf)及元件檔(full_adder.sym)儲存於該資料夾。因為製作全加器元件時是使用半加器元件來製作，所以連帶要複製半加器元件的繪圖檔(half_adder.gdf)及元件檔(half_adder.sym)儲存於該資料夾內。
2. 執行 MAX+PLUS Ⅱ 軟體，開啟新的圖形編輯檔，叫出儲存 4BITS_ADDER 資料夾的全加器元件檔 full_adder.sym，利用它配合邏輯閘完成四位元全加器繪圖，如圖 3-2-58 所示。

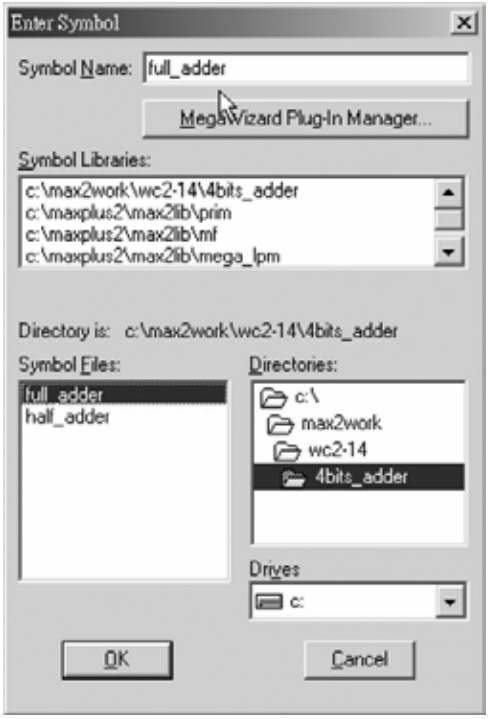

圖 3-2-50　零件取用視窗

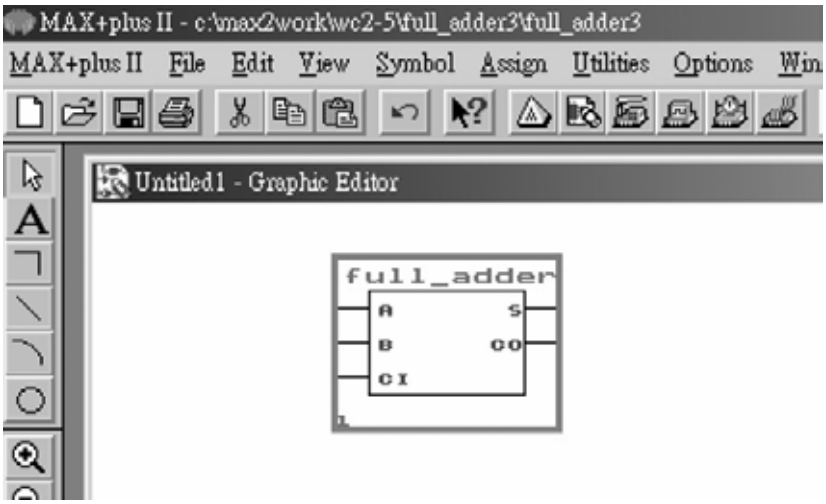

圖 3-2-51　全加器自製元件

如圖 3-2-51 叫出四個全加器元件，依四位元全加器電路繪圖，在此注意一點，因為同樣類型的線路不少，可利用匯流排(Bus)的畫法以減少線路複雜度，且較為簡單。匯流排畫法是用較粗的線表示匯流排，並需以匯流排方式命名，匯流排分支的每單一導線需用細線，且需配合匯流排命名。如圖 3-2-52 所示，匯流排命名需用中括號，四個全加器皆有 A 輸入，用一條四線的匯流排來取代分別畫四條線，在此有 0,1,2,3 四條線，取名匯流排名稱 A[3..0]，點選線路，該線會變成紅色，按下滑鼠右鍵，選擇 Line Style 中較粗的線來當匯流排，選擇線

路時也可使用 Options→Line Style 來更換線的型態。

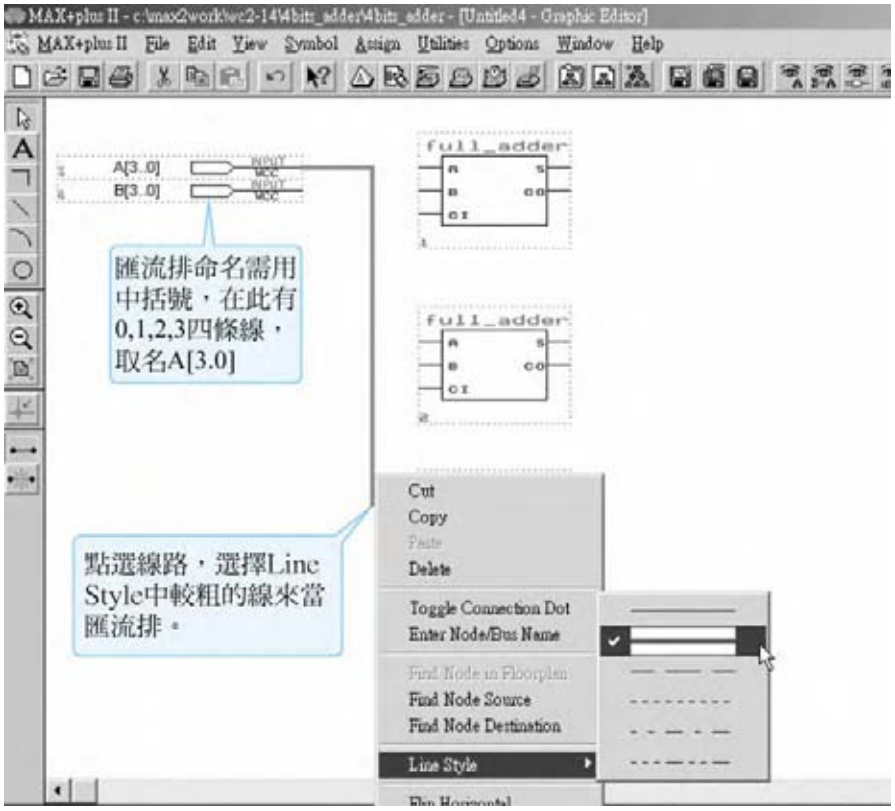

圖 3-2-52　四位元加法器電路

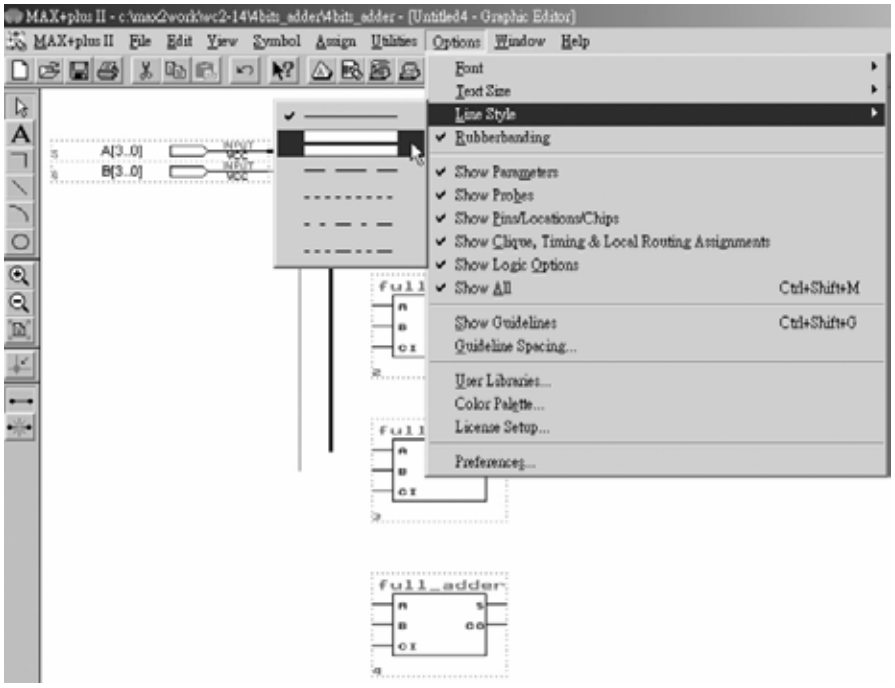

圖 3-2-53　四位元加法器電路

3. 畫完匯流排主線仍需畫連接到各接點的細線，每一條細線都需配合主線命名，可用滑鼠點選該線，其即變成紅色，此時可直接輸入名稱，在此要特別注意一點，當輸入名字後，點選該線時，該線以及名字都會變成紅色，表示有被選取到，此時命名才算成功。假若點選該線而名字沒有被選取到，則表示該線未被命名，線旁的文字只是文字而非該線的名稱。

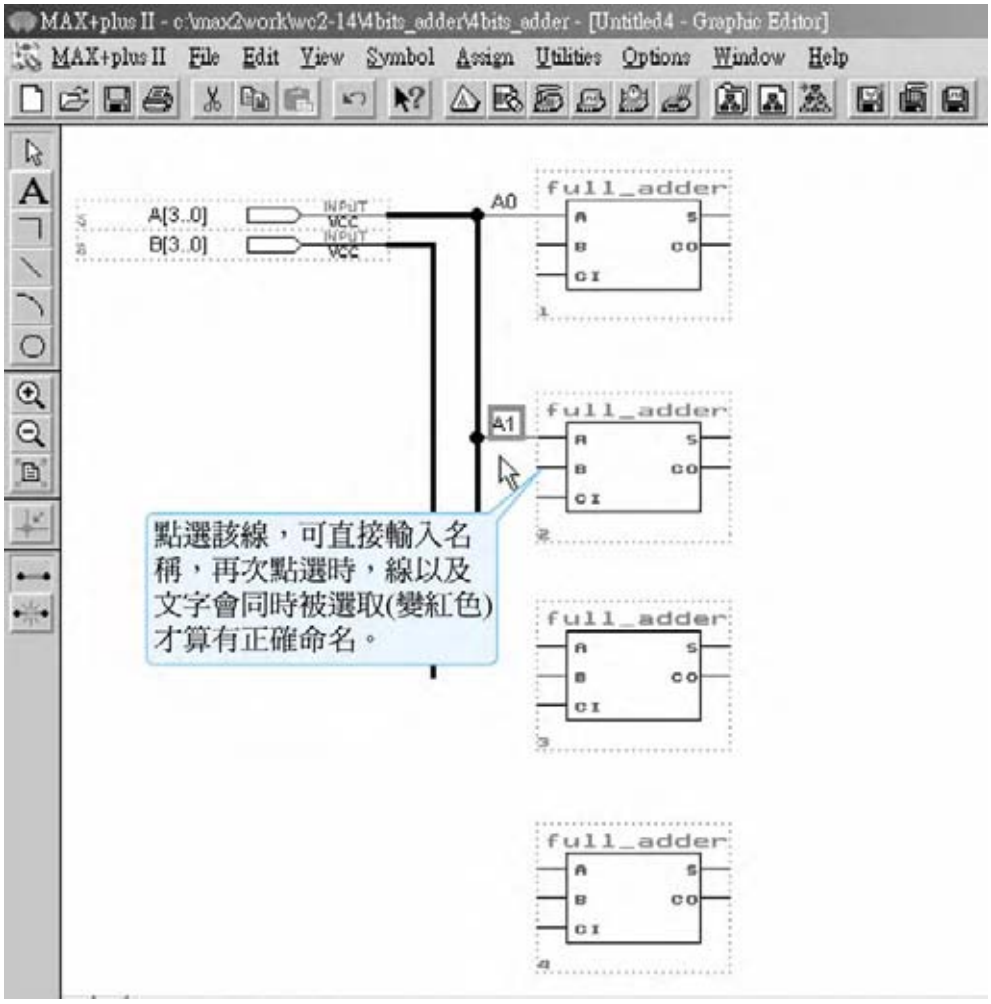

圖 3-2-54　四位元加法器電路

4. 畫匯流排的最末端線路時，當其變成單一條線時，仍需用細線表示，如圖 3-2-55 所示。

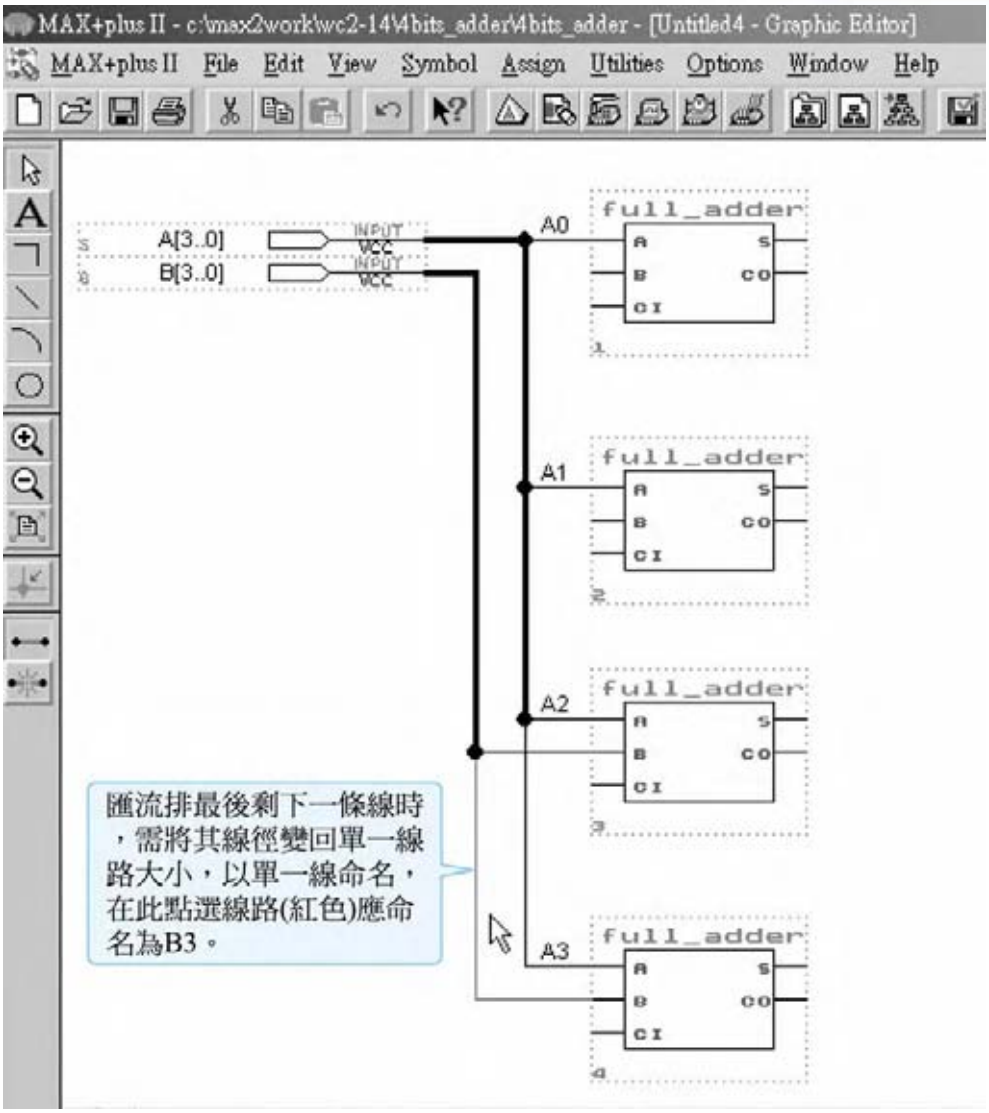

圖 3-2-55　四位元加法器電路

5. 點選 Symbol→Enter Symbol，輸入 GND 即可得到接地符號，用來連接線路接地用。如圖 3-2-57 所示，完成四位元全加器之繪圖。

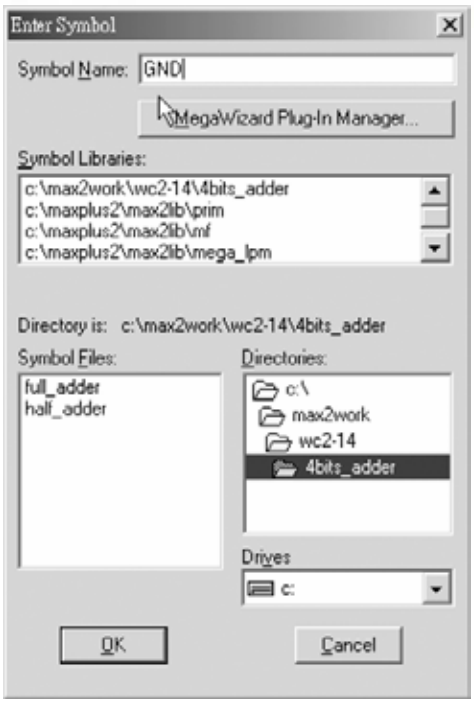

圖 3-2-56　元件取用視窗

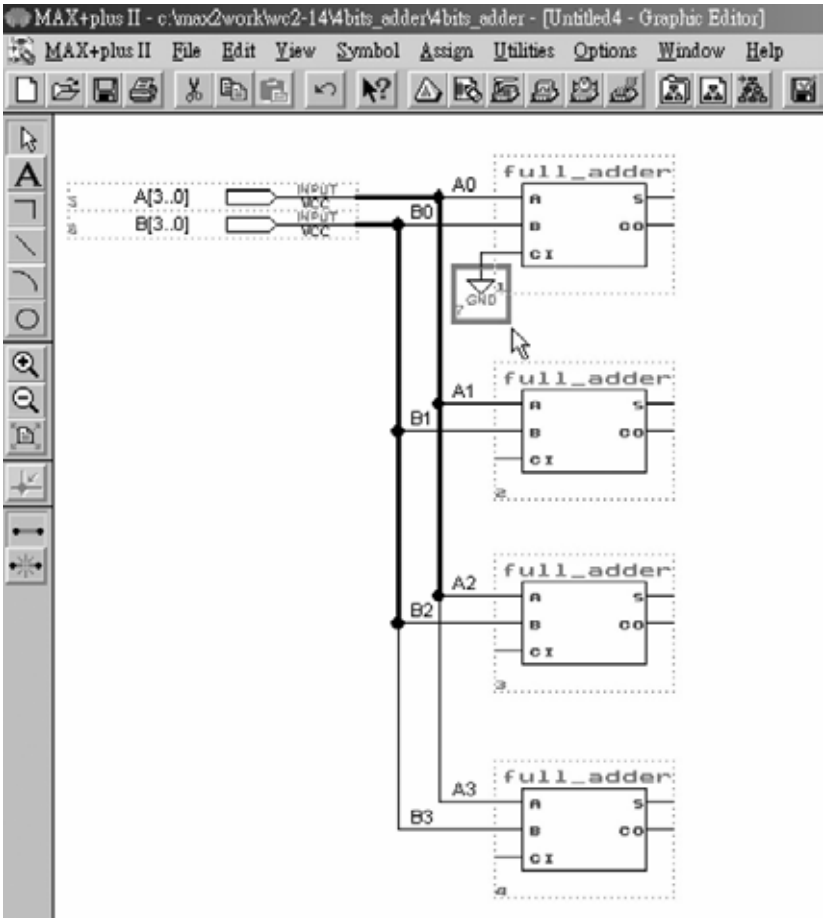

圖 3-2-57　四位元加法器電路

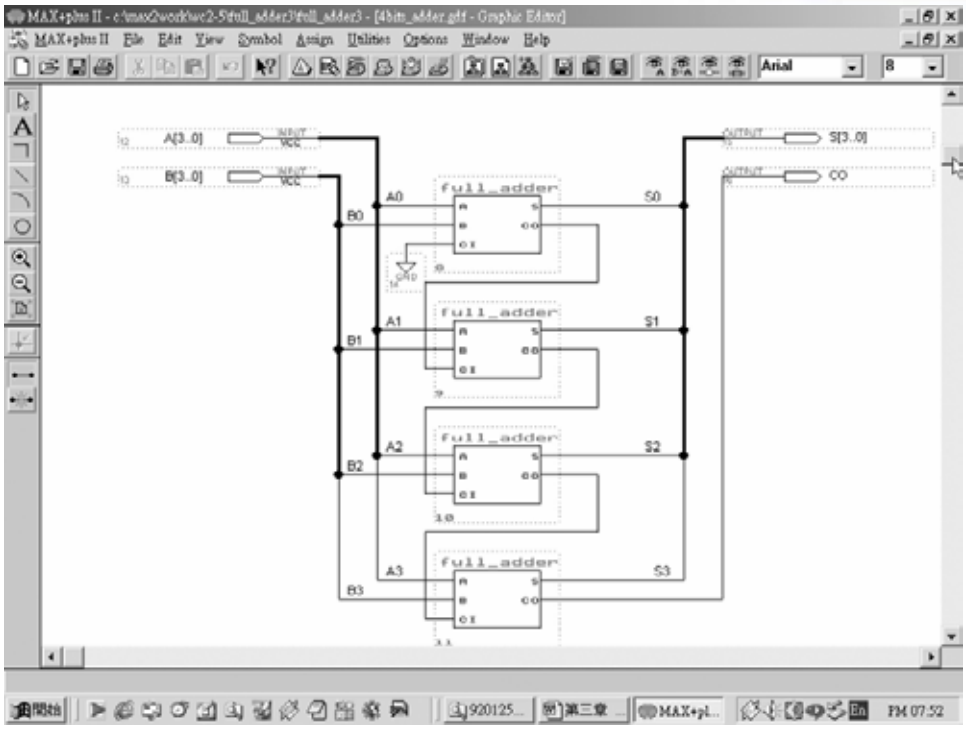

圖 3-2-58　四位元加法器電路

6. 點選 File→Save As，存檔於 4BITS_ADDER 資料夾內。在此取名四位元加法器圖形檔案名稱為 4bits_adder。

7. File→Project→Set Project to Current File

8. 指定 CPLD 晶片(Assign→Device)

9. 編譯(MAX+PLUS Ⅱ→ Compiler)

設計完四位元全加器後，要知道其是否可正常執行，可執行模擬功能測試。

模擬步驟如下：

1. 開啟新的波形編輯檔案，設定功能模擬結束時間(File→End Time)，在此設 256 μs；設定格線間距(Options → Grid Size)，在此設 1 μs；顯示在視窗中適當大小格線(View→Fit in Window)。

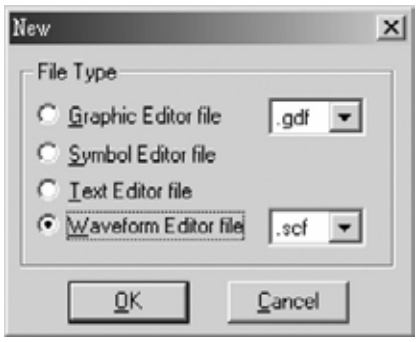

圖 3-2-59　開啟新檔視窗

圖 3-2-60　模擬結束時間設定視窗

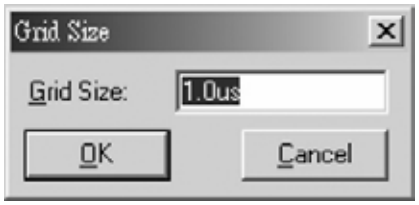

圖 3-2-61　模擬單位時間設定視窗

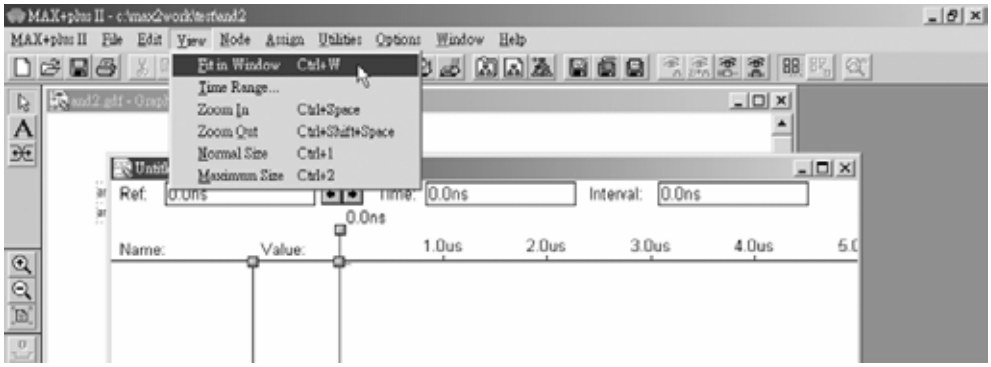

圖 3-2-62　視窗調整

2. 儲存檔案(Save As)，檔名 4bits_adder.scf。輸入節點(Node → Enter Nodes from SNF，按 List 及 =≥ ，OK)，編輯輸入計數器 (XC)。

圖 3-2-63　儲存檔案視窗

圖 3-2-64　執行輸出入節點選擇視窗

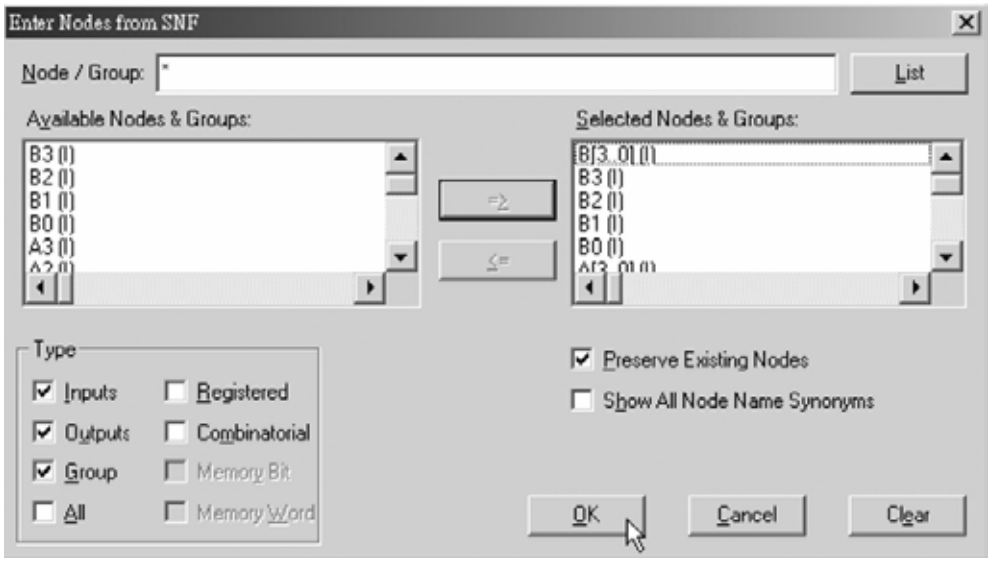

圖 3-2-65　輸出入節點選擇視窗

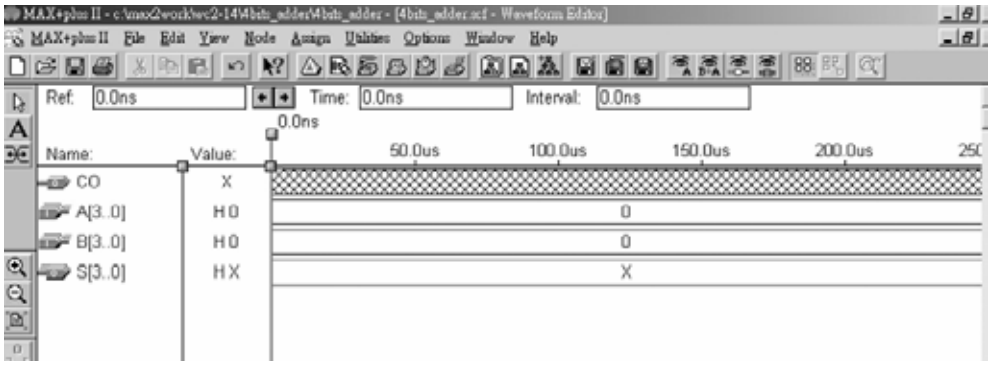

圖 3-2-66　模擬波形設定

3. 點選 A[3..0]，再點選 XC，設定波形經每 1 個 grid size 轉態一次(Multiplied By = 1)，且每次轉態時計數器數值增加 1 (Increment By = 1)，以二進制(Binary)表示；點選 B[3..0]，再點選 XC，設定波形經每 16 個 grid size 轉態一次(Multiplied By = 16)，且每次轉態時計數器數值增加 1 (Increment By = 1)，以二進制(Binary)表示。

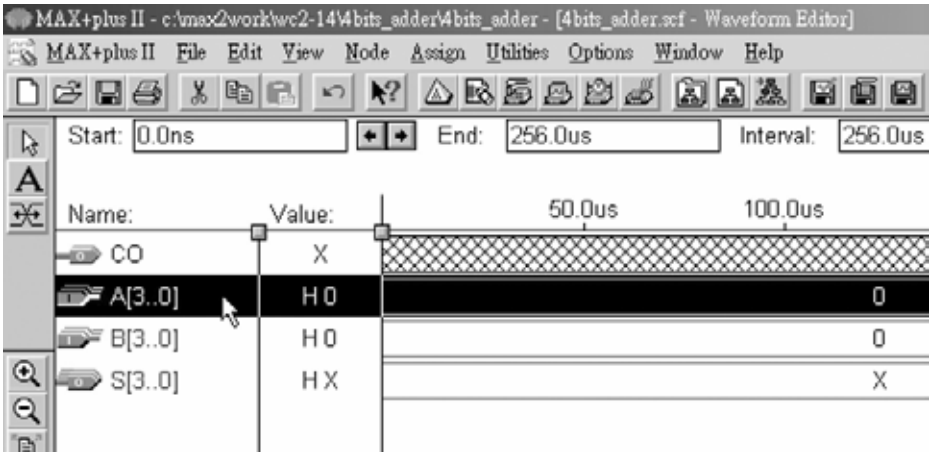

圖 3-2-67　時脈信號設定

圖 3-2-68　時脈信號設定視窗

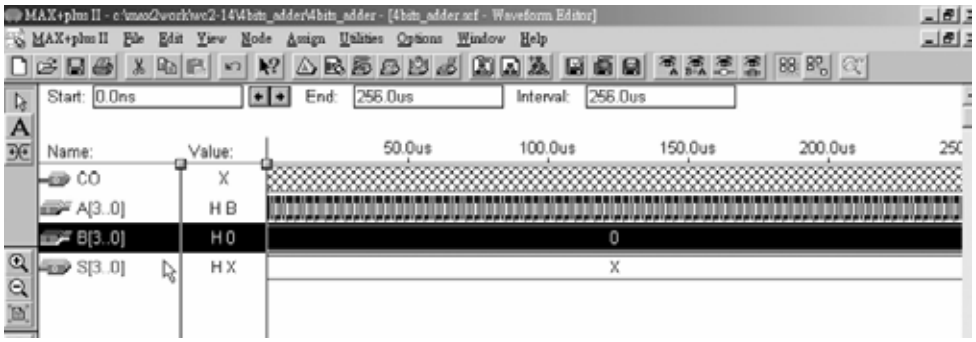

圖 3-2-69　模擬波形設定

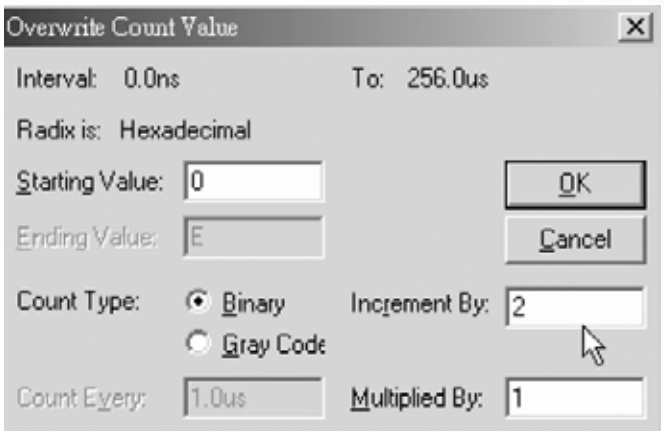

圖 3-2-70　計數信號設定視窗

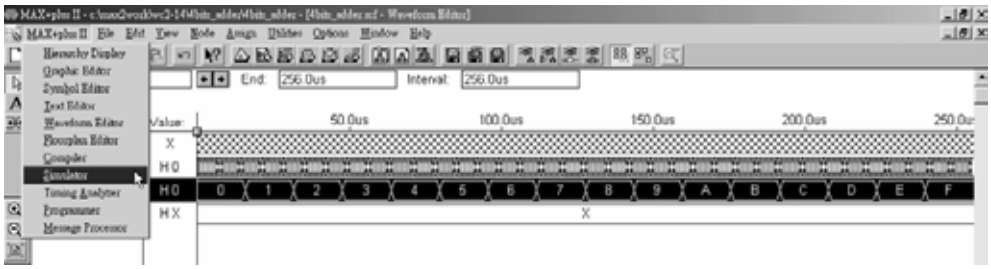

圖 3-2-71　執行模擬

4. 執行模擬(MAX+PLUS Ⅱ→ Simulator，按下 Start)如圖 3-2-73 所示，模擬沒有錯誤及警告，所得波形模擬結果符合四位元全加器電路之結果，代表我們製作的電路是正確可用的。

在此說明模擬結果，因為是四位元加法器，所以被加數 A[3..0]有四位元，16 種數值組合(0000~1111 代表 0~15)，加數 B[3..0]亦有四位元，同樣 16 種數值組合(0000~1111 代表 0~15)。兩數相加則有 16×16=256 種組合，所以模擬之時間設定從 0~256 μs，Grid Size=1 μs，可模擬 256 種狀態。且被加數設定每 1 μs 轉態一次(從 0 開始)，因為被加數有 16 種組合，所以加數設定每 16 μs 狀態一次(從 0 開始)，剛好也可將加數的 16 種狀態表示出來(256 μs /16 μs =16)，如此便可顯示所有狀態的模擬結果。

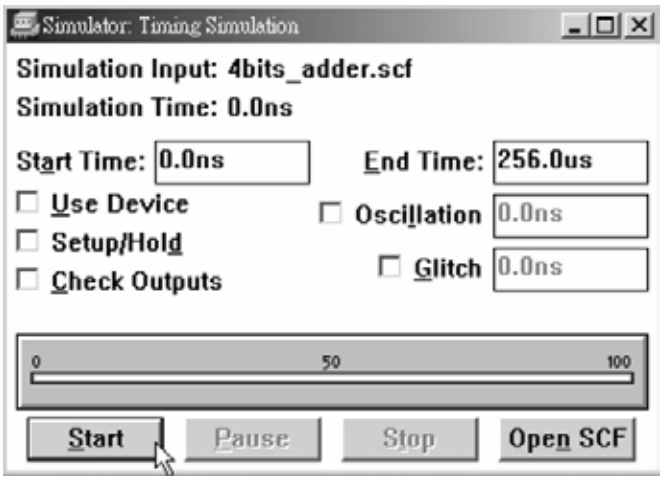

圖 3-2-72　模擬起始視窗

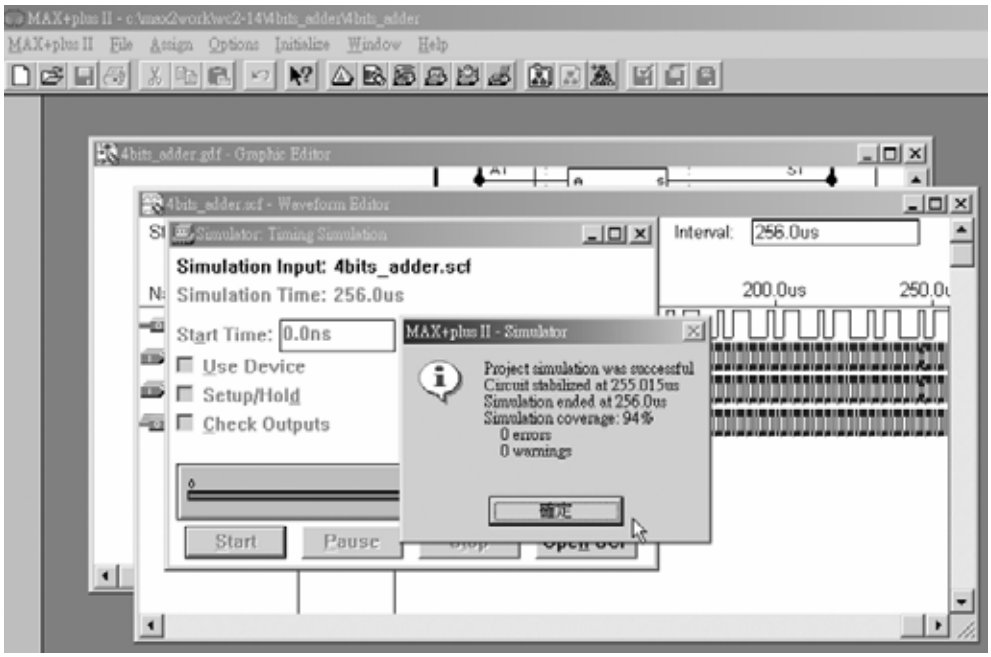

圖 3-2-73　模擬訊息視窗

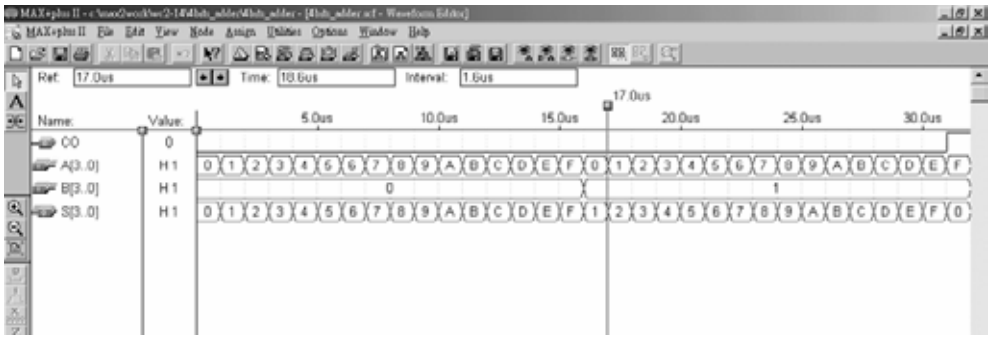

圖 3-2-74　模擬結果

5. 可用 來調整適當格子大小，觀測各時間點之波形是否正確。例如在第 17~18 μs 時，A[3..0]輸入 1；B[3..0]輸入 1；兩數相加可得和 S[3..0] = 2；進位 Co = 0；表示沒有進位，符合十六進制加法運算結果。又例如在第 79~80 μs 時，A[3..0]輸入 F(即十進制 15)，B[3..0]輸入 4，兩數相加可得和 S[3..0] = 3，進位 Co = 1，表示有進位，符合二進制四位元加法器運算結果，可知模擬結果正確。

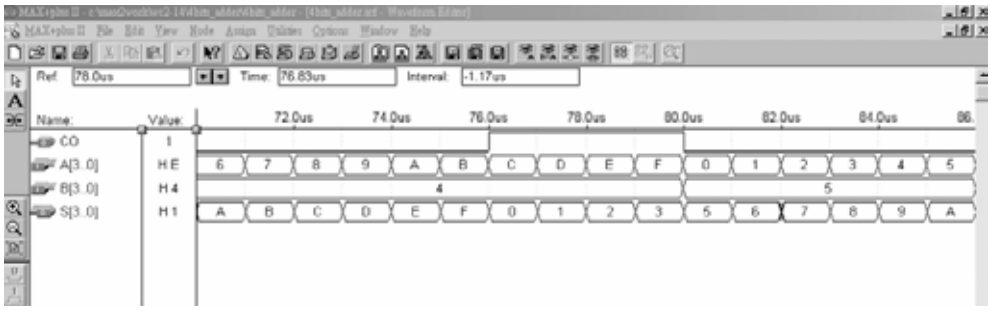

圖 3-2-75　模擬結果

模擬成功之後，可下載(燒錄)到實驗板做實際電路測試，以下為接到尼德公司實驗板的接腳表格，讀者可以按表中接腳設定以完成硬體電路測試。

表 3-2-2　電路圖輸出入腳位、CPLD 腳位及實驗器模組腳位對應表

腳位對應關係	輸入		輸出	
名稱	被加數 A[3..0]	加數 B[3..0]	和 S[3..0]	進位 Co
CPLD 晶片腳位	PIN 24,22,21,20	PIN 12,11,10,9	PIN 76,75,74,73	PIN 77
實驗器模組對應腳位	DIPA 4,3,2,1	DIPB 4,3,2,1	DG 3,2,1,0	DG4

實驗器輸入接腳有被加數 A[3..0]四支腳，可用指撥開關 DIPA4~DIPA1 來表示，加數 B[3..0]四支腳，可用指撥開關 DIPB4~DIPB1 來表示。輸出端有和 S[3..0]四支腳及進位 C0 一支腳可用綠色發光二極體 DG3~DG0 以及 DG4 來表示。

3-2-6 八位元加法器

完成四個位元相加，單純用邏輯閘電路設計與用自製全加器元件來設計的差別，相信讀者已看出來，使用自製全加器使電路有相當多的簡化。除了四位元加法器可簡單完成，對於八位元加法器，一樣可用同樣的方式來製作，在此我們將以上一單元做成的全加器元件來完成八位元全加器的製作。

	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
		A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
+		B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
	C_7	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

八位元加法器運算如上圖所示，A_0~A_7 表示被加數八個位元，B_0~B_7 代表加數的八個位元，其相加產生的進位 C_0~C_7 分別加入其次一位元的運算中，產生的結果和為 S_0~S_7 以及最後的進位 C_7。依照如此原則，要以全加器元件製作八位元加法器，第一個全加器的被加數及加數分別接 A_0 及 B_0，其進位輸入端接地(因為第一位元並無上一級的進位，所以其值為 0)。第一個全加器的 S(和)輸出端直接是第一位元 S_0(和)的輸出值，進位輸出端 C_0 則接到第二個加法器的進位輸入端。第二個全加器則接被加數 A_1 及加數 B_1，產生和(S_1)輸出以及進位(C_1)接到第三個加法器，以此類推接完八個加法器，與四位元加法器極為類似。

與四位元加法器相同方式，只要是使用既有的自製元件來完成新電路，皆建議使用者建立新資料夾來儲存，製作步驟如下：

1. 建立新資料夾，取名 8bits_adder，並複製前面單元製作全加器元件所完成繪圖檔(full_adder.gdf)及元件檔(full_adder.sym)儲存於該資料夾。因為製作全加器元件時是使用半加器元件來製作，所以連帶要複製半加器元件的繪圖檔(half_adder.gdf)及元件檔(half_adder.sym) 儲存於該資料夾內。

圖 3-2-76　元件取用視窗

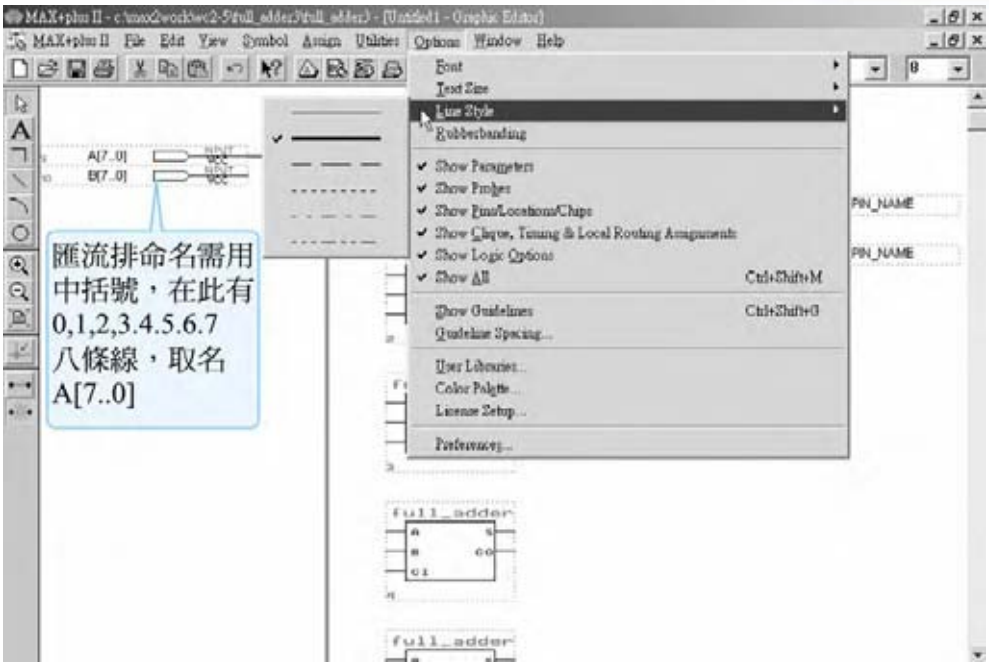

圖 3-2-77　八位元加法器電路

2. 執行 MAX+PLUS Ⅱ 軟體，開啟新的圖形編輯檔，叫出儲存 8bits_adder 資料夾的全加器元件檔 full_adder.sym，利用它配合邏輯閘完成八位元全加器繪圖，如圖 3-2-79 所示。(因為電路圖甚大，讀者若在書本上看不清楚，可在隨書附贈光碟的 8bits_adder 資料夾內找到此檔。)

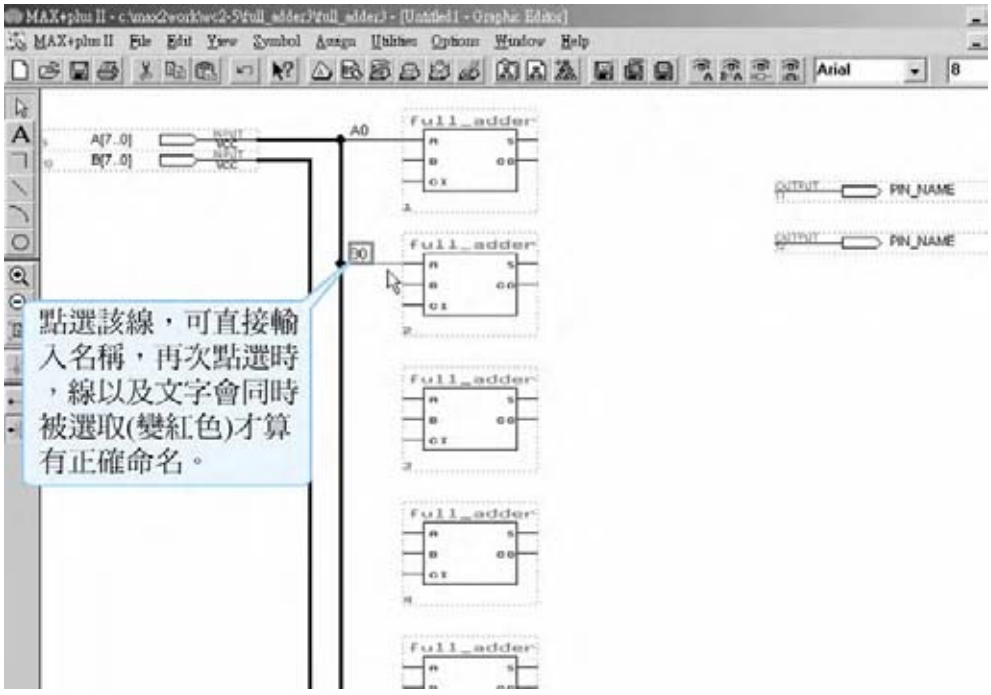

圖 3-2-78　八位元加法器電路

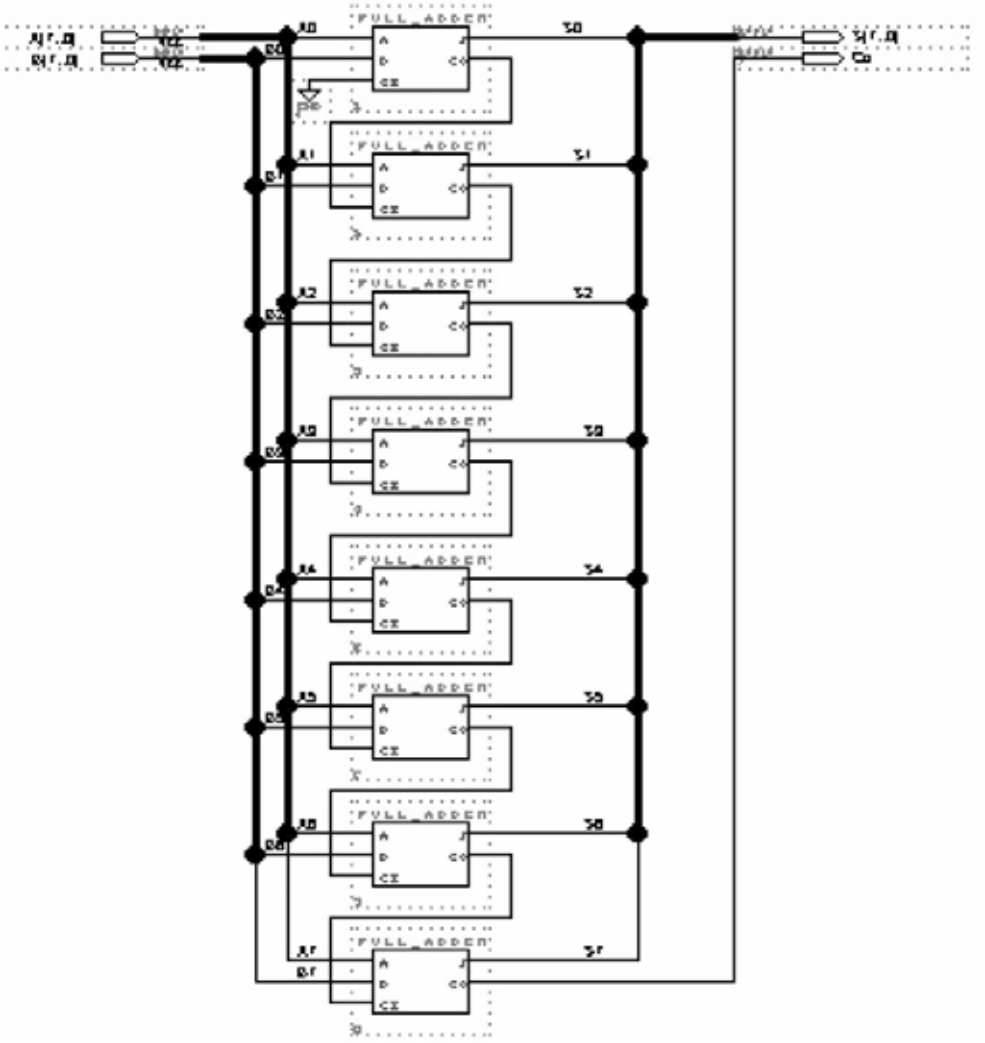

圖 3-2-79　八位元加法器電路

3. 點選 File→Save As，存檔於 8bits_adder 資料夾內。在此取名四位元加法器圖形檔案名稱為 8bits_adder。
4. File→Project→Set Project to Current File。
5. 指定 CPLD 晶片(Assign→Device)。

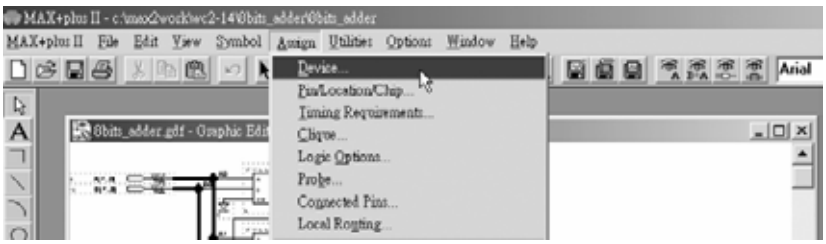

圖 3-2-80　執行指定 CPLD 元件視窗

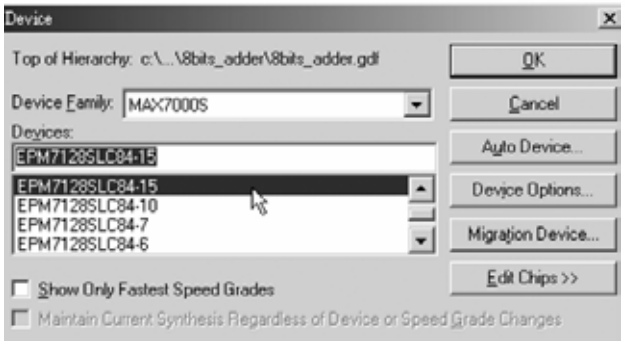

圖 3-2-81　指定 CPLD 元件視窗

6. 編譯(MAX+PLUS Ⅱ→ Compiler)

設計完成八位元全加器後，要知道其是否可正常執行，可執行模擬功能測試。

模擬步驟如下：

1. 開啟新的波形編輯檔案，設定功能模擬結束時間(File→End Time)，在此設 256 μs；設定格線間距(Options → Grid Size)，在此設 1 μs；顯示在視窗中適當大小格線(View→Fit in Window)。

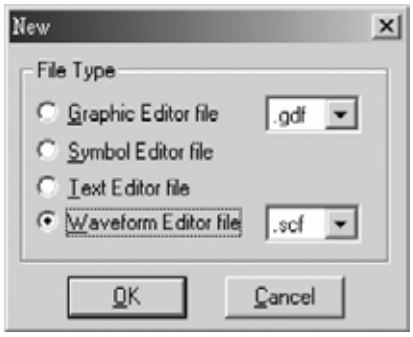

圖 3-2-82　開啟新檔視窗

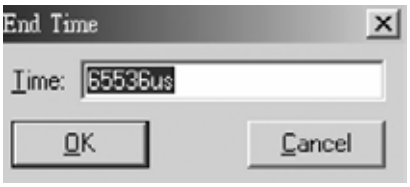

圖 3-2-83　模擬結束時間設定視窗

圖 3-2-84　模擬單位時間設定視窗

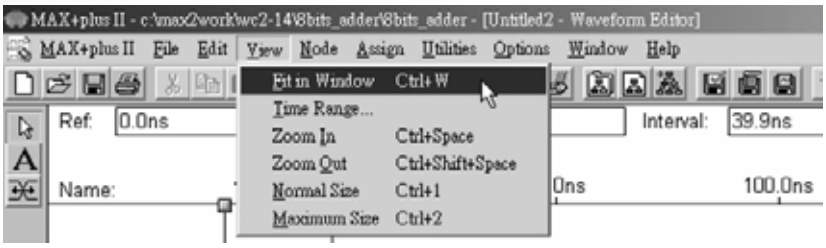

圖 3-2-85　視窗調整

2. 儲存檔案(Save As)，檔名 8bits_adder.scf。輸入節點(Node → Enter Nodes from SNF，按 List 及 =≥ ，OK)，編輯輸入計數器 (XC)。

圖 3-2-86　儲存檔案視窗

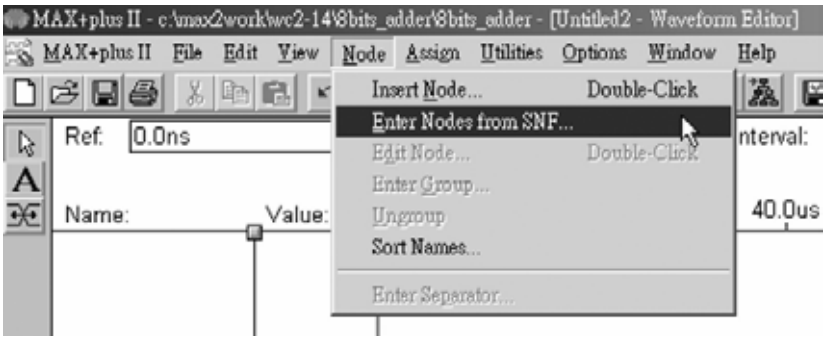

圖 3-2-87　執行輸出入節點選擇視窗

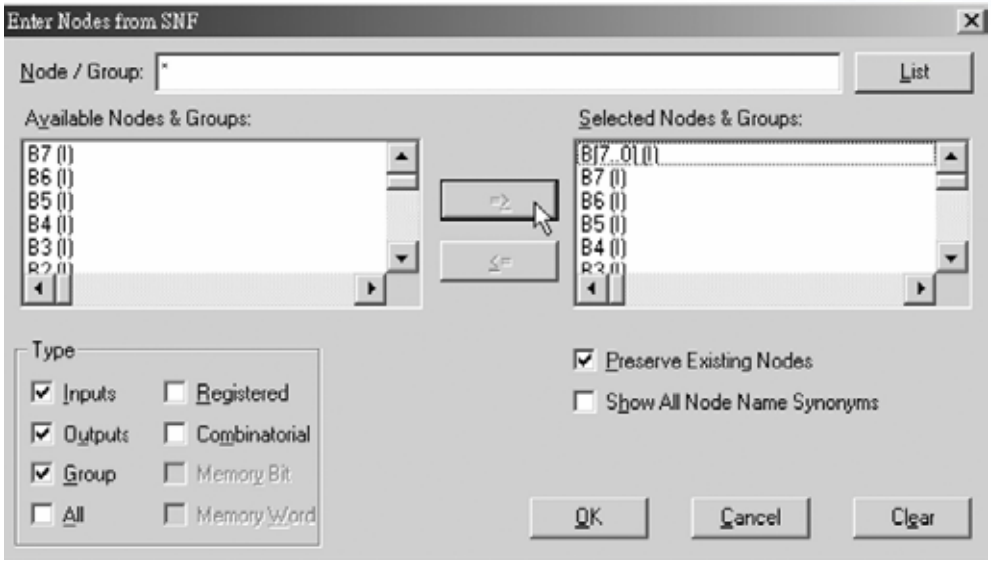

圖 3-2-88　輸出入節點選擇視窗

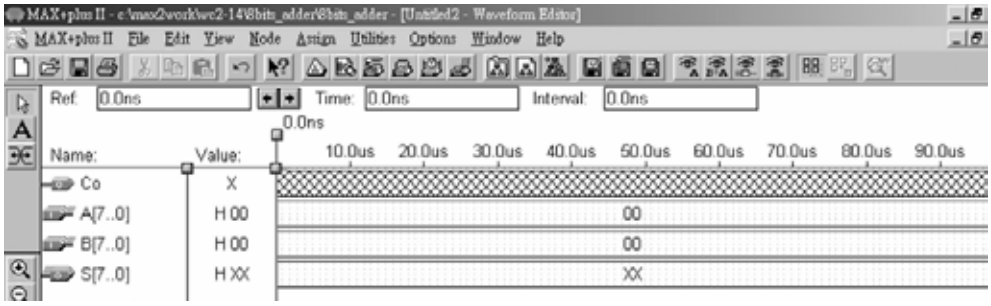

圖 3-2-89　模擬波形設定

3. 點選 A[7..0]，再點選 XC，設定波形經每 1 個 grid size 轉態一次(Multiplied By = 1)，且每次轉態時計數器數值增加 1 (Increment By = 1)，以二進制(Binary)表示；點選 B[7..0]，再點選 XC，設定波形經每 256 個 grid size 轉態一次(Multiplied By = 256)，且每次轉態時計數器數值增加 1 (Increment By = 1)，以二進制(Binary)表示。

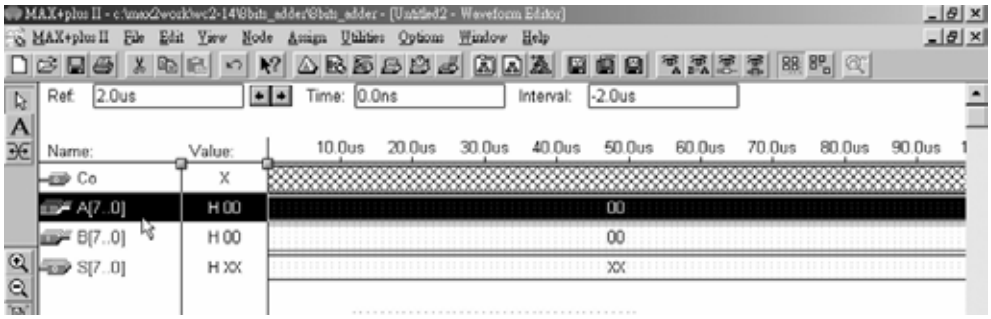

圖 3-2-90　模擬波形設定

圖 3-2-91　計數信號設定視窗

圖 3-2-92　計數信號設定視窗

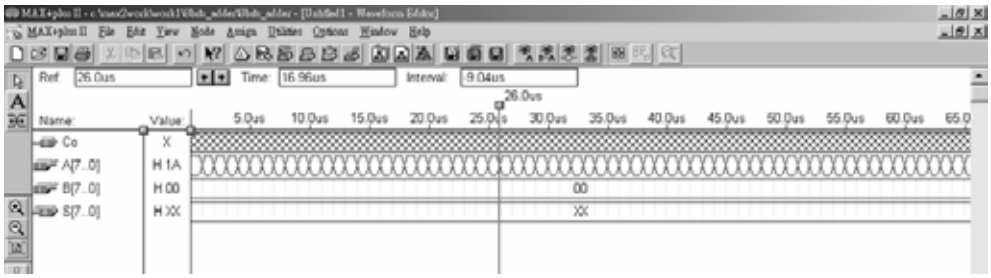

圖 3-2-93　模擬波形設定

4. 執行模擬(MAX+PLUS II → Simulator，按下 Start)如圖 3-2-95 所示，模擬沒有錯誤及警告，所得波形模擬結果符合八位元全加器電路之結果，代表我們製作的電路是正確可用的。

在此說明模擬結果，因為是八位元加法器，所以被加數 A[7..0]有八位元，256 種數值組合(00000000~11111111 代表 0~255)，加數 B[7..0]亦有八位元，同樣 256 種數值組合(00000000~11111111 代表 0~255)。兩數相加則有 256*256 = 65536 種組合，所以模擬之時間設定從 0~65536 μs，Grid Size=1 μs，可模擬 65536 種狀態。且被加數設定每 1 μs 轉態一次(從 0 開始)，因為被加數有 256 種組合，所以加數設定每 256 μs 狀態一次(從 0 開始)，剛好也可將加數的 256 種狀態表示出來(65536 μs /256 μs =256)，如此便可顯示所有狀態的模擬結果。

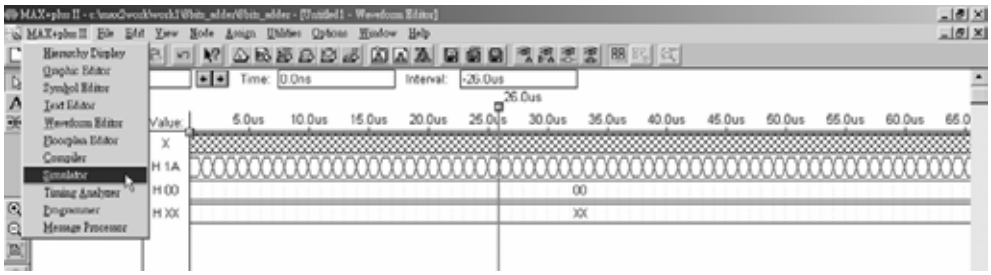

圖 3-2-94　執行模擬

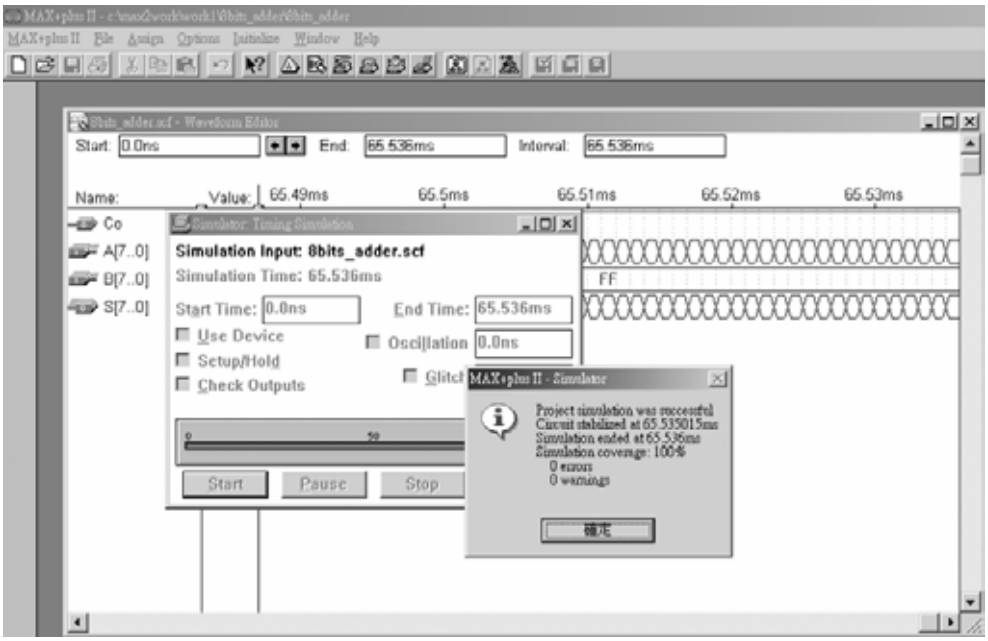

圖 3-2-95　模擬訊息視窗

5. 可用 🔍 🔍 來調整適當格子大小，觀測各時間點之波形是否正確。例如在第 1.278ms 時，A[7..0]輸入 FEH；B[7..0]輸入 04H；兩數相加可得和 S[7..0] = 2；進位 Co = 1；表示有進位，符合二進制八位元加法器運算結果，可知模擬結果正確。

圖 3-2-96　模擬結果

模擬成功之後，可下載(燒錄)到實驗板做實際電路測試，以下為接到尼德公司實驗板的接腳表格，讀者可以按表中接腳設定以完成硬體電路測試。

表 3-2-3　電路圖輸出入腳位、CPLD 腳位及實驗器模組腳位對應表

腳位對應關係	輸入		輸出	
名稱	被加數 A[7..0]	加數 B[7..0]	和 S[7..0]	進位 Co
CPLD 晶片腳位	PIN 29,28,27,25, 24,22,21,20	PIN 18,17,16,15, 12,11,10,9	PIN 81,80,79,77, 76,75,74,73	PIN 61
實驗器模組對應腳位	DIPA 8,7,6,5,4,3,2,1	DIPB 8,7,6,5,4,3,2,1	DG 7,6,5,4,3,2,1,0	DR0

實驗器輸入接腳有被加數 A[7..0]八支腳，可用指撥開關 $DIPA_8$~$DIPA_1$ 來表示，加數 B[7..0]八支腳，可用指撥開關 $DIPB_8$~$DIPB_1$ 來表示。輸出端有和 S[7..0]八支腳及進位 C_o 支腳可用綠色發光二極體 DG_7~DG_0 及紅色發光二極體 DR_0 來表示。