含可靠性及繞線性之元件擺置的研究

Jing Lee

Department of Electronic Engineering

Southern Taiwan University of Technology

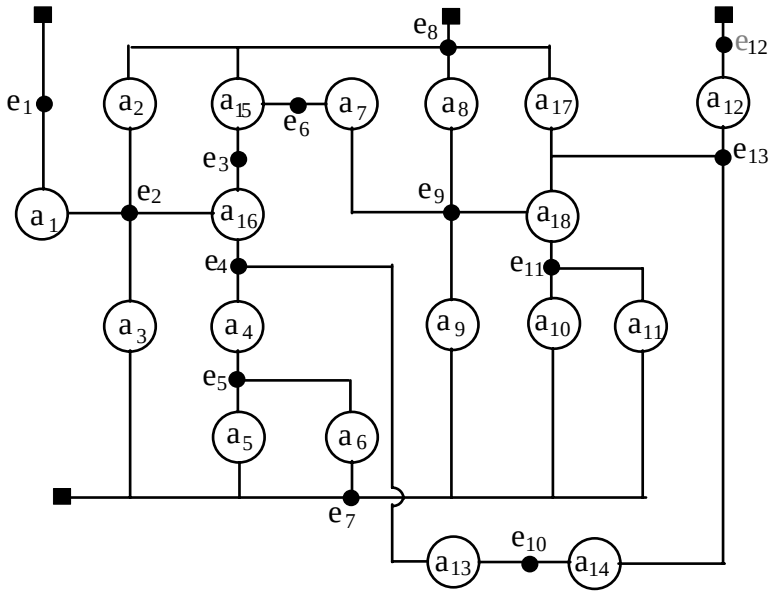
Tainan, Taiwan 701, R.O.C.

Email: leejing@mail.stut.edu.tw

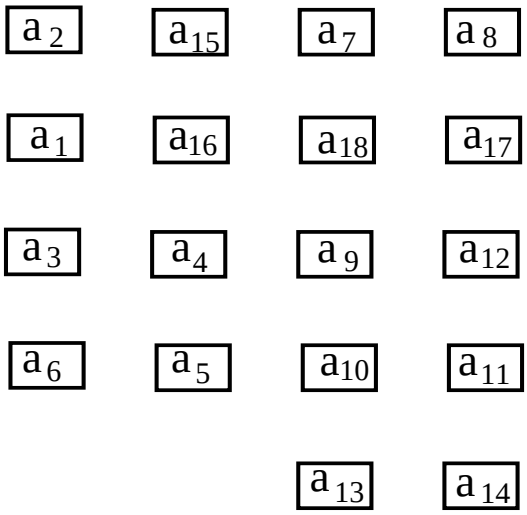
第一章　導 論

由於微電子技術的進步，使得電子電路積集化的程度愈來愈高，電路佈局的複雜度也隨之增加；另方面電子產品更新的速度愈來愈快，使得允許佈局設計的時間大為縮短，也因此對自動化佈局的需求日益增加。

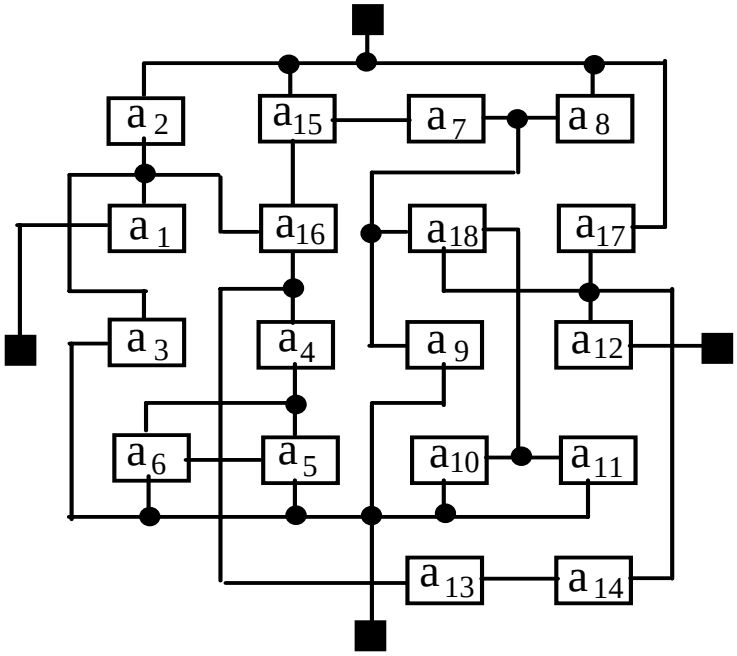
受限於佈局的複雜度過高，通常將佈局分成元件擺置(placement)及繞線(routing)兩階段分別處理。由於元件擺置的好壞不但直接影響隨之而來的佈線的困難度及最後佈局面積的大小，也決定晶片或基板上的溫度分佈進而影響系統的可靠性，故元件的擺置在整個佈局階段居於關鍵地位。



(a) 含有18個元件的電路 (繼續)



(b) 元件的擺置



(c) 繞線

圖1.1：一個電路及其佈局

為了使元件擺置的問題可以得到適當的簡化，通常將此問題分成系統階層、印刷板階層、及晶片階層等各別處理。

1.1、系統階層

系統階層的擺置如圖1.2所示，擺置目標在於要求在最小的體積內將PCB板組合，在此同時模組產生的熱量必須被有效的冷卻。

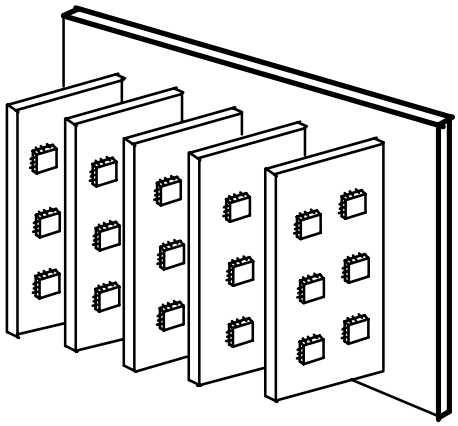


圖1.2：系統階層的元件擺置

1.2、印刷板階層

印刷板階層的擺置如圖1.3所示，元件擺置的目標在於妥善的將IC擺置在PCB板上，使得隨後之佈線的困難度降至最低且所需的佈線層必須為最少，與此同時也必須將散熱的好壞加入考慮，以確保電路的高可靠性。

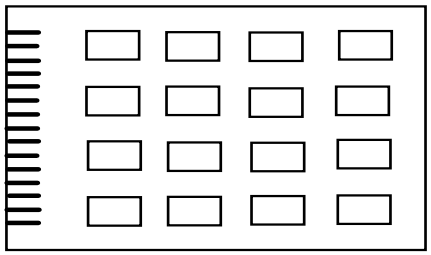


圖1.3：板子階層的元件擺置

1.3、晶片階層

晶片階層的元件擺置如圖1.4、1.5及1.6所示。在晶片階層，元件擺置的目標要求在最小的面積內完成元件擺置及隨後的佈線。

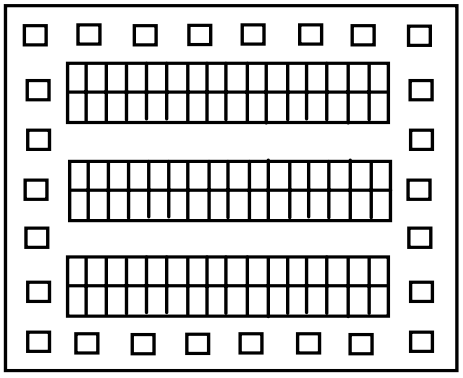


圖1.4：晶片階層的元件擺置(閘陣列)

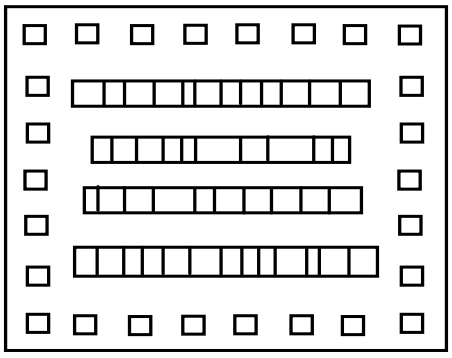


圖1.5：晶片階層的元件擺置(標準元)

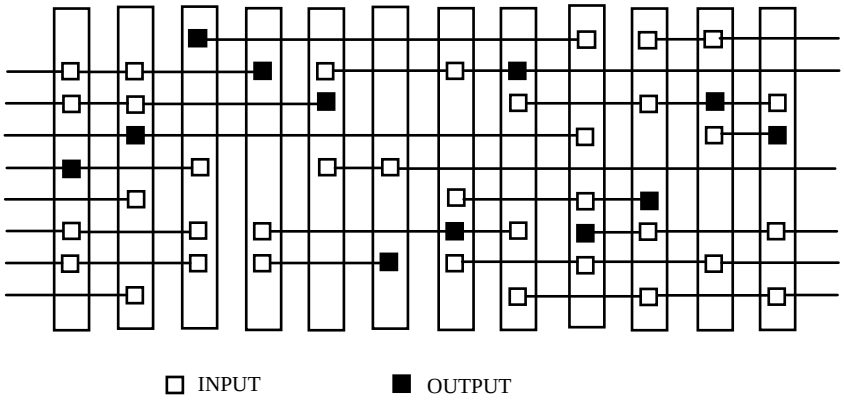


圖1.6：晶片階層的元件擺置(一維閘列)

第二章　電路的抽象模型

為了能夠有效的處理元件擺置的問題，以抽象的資料結構取代實際的電路是很有必要的，因為它能使演算法專注於接線的關係，而去除元件的形狀、大小、及方位等次要的細節，這些次要的細節可以在後處理時再加以考慮。**圖形**常被用來表示電路上的接線關係。在圖形結構中，元件以**節點**表示，任兩元件間若以導線相連，則圖形中其對應的兩節點間即以一條**邊**連接起來。由此對應關係可知：若一條導線連接了n 個元件，在圖形中必需使用 n(n-1)/2 條邊表示，此種**邊數擴大效應**增加了在圖形結構中評估元件間關連性的困難度 [Sch72]。為了克服這個困難，通常採用邊加權的方法以降低邊數擴大效應。Charney 和 Plato 提出 2/(n-1) 加權法 [Cha68]；Hanan 和 Kurtzberg 採用 2/n加權法 [Han72]；Donath 則建議 4/(n^2-mod(n,2)) 加權法 [Don88]。這些加權法事實上都祇能部份的修正邊擴大的影響。近年來，愈來愈多的學者採用**超圖形**來表示電路的接線關係，因為超圖結構可以很自然的反映電路的接線關係。在超圖的模型中，元件仍用節點表示，任一導線則使用一個節點集合表示，此集合(又稱為**超邊**)是由此導線所連接的所有元件(節點)所構成的。圖 2.3 即為圖 2.1 的超圖模型。

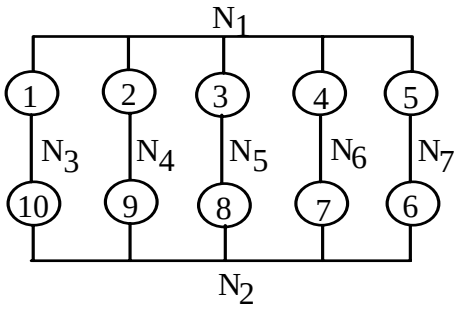


圖2.1：包含10個元件及7條導線的電路

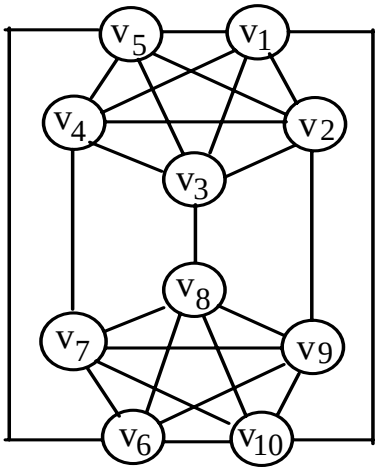


圖2.2：圖2.1的圖模型.

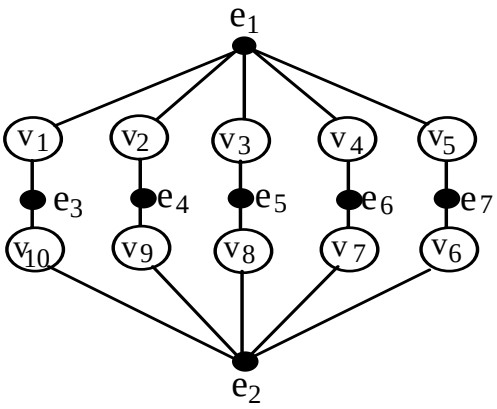


圖2.3：圖2.1的超圖模型

第三章　電路分割

元件的數目愈多，電路的設計與佈局愈加困難。因此複雜的系統必須分割成較小的系統才能有效的處理。所謂**電路分割**問題就是把大的電路分割成數個較小的子電路，並且這些子電路間的接線必需儘量的減少，以便各別處理時相互間的干擾為最低。由於**分割**問題屬於 **NP 完備**的問題之一，因此不存在複雜度為多項式等級的**確定性演算法**，可以確保獲得最佳解。較可行的方法是採用**非確定性的演算法**

(或稱為**明智的(heuristic)演算法**) 得到次佳的解答。這類的演算法往往利用某些目標函數或規則引導求解過程的進行，若過程中出現了數條可能的行進途徑時，即由這些目標函數評估(實際上是猜測)它們的優劣，並選擇其中一條途徑前進，如果每次都能猜到正確的行進途徑，就可以得到最佳的解答；反之，則無法得到最佳解。明智法的缺點在於容易陷入局部最佳解的陷阱。

分割問題的明智演算法，可分為**由上而下**法及**由下而上**法兩大類。由上而下法又稱為**最小割**(min-cut)法。最小割法假設所有的節點都有相同的重要性，而且沒有元素被事先固定它的位置。它開始於兩個已被分割的子集合，而後藉由逐次調換此兩集合的元素以獲得最佳解。由於這個方法忽略了元件間的差異性，因而使得它的實用性受到限制，因為很多實際的元件其面積並不相等，發熱功率也不相同，而且某些熱敏感性高的元件可能必須遠離高功率的元件以免受到影響。此外，這個方法也未限制訊號線最長的長度，因此在高速電路的設計上可能也不適用。不過若排除以上不利的情況，這個方法通常得到很好的解答。它的時間複雜度為$0(n^3)$至$0(n^4)$，跟其它的構成性(constructive)演算法比起來是偏高。

由下而上的方法也稱**叢聚法**(clustering method)，這個方法由某些事先指定位置的節點(稱為**種子**，seed)開始，這些種子可以人為指定，或選擇某些特殊的元件如I/O接點，或電路中的重要元件如 CPU等。這些種子逐次吞併周遭連接性較強的節點而成為**節點叢**(cluster)。每個節點的價值可以不同以反映元件間的差異，但每個節點叢的總價值都受到限制 (若為均勻分割，則總價值都相等)，所以若一節點叢的總值已達上限，則不可以再吞併其它節點。當所有的節點均已完成合併時，分割即告完成。由於吞併的過程優先吞併與節點叢連接性較強的節點，所以這個方法猜想及期望最後連接這些節點叢的邊數為最少。這個方法的優點在於每個節點均可以加入很多不同的權值以反映真實元件的差異，所以它很適合處理多目標的分割問題，因為藉由節點叢的總值限制，此方法即可達成不同目標間的妥協。此外，比起由上而下法，它的速度也快的多，實際的時間複雜度在$0(n)$ 至 $0(n^2)$之間。它的缺點在於每次的合併都是根據局部的資訊所做的決定，所以很容易落入局部最佳化的陷阱中，特別是當種子選擇不當時此問題更嚴重。

早期的叢聚法多半採用圖或加權圖模擬電路，因為它們比較容易定義節點間的連接性，但其缺點則在於受到邊數擴大的效應，以致無法正確的反應出元件間連接性的真實大小。本章提出超圖的叢聚法以免除邊數擴大的誤差，此外本章也提出一個**種子產生器**以決定種子，此產生器的考慮因素如下：

1)種子必須與最多的節點相連，如此以來在決定被合併的元件時可以做更整體的考量，也就更不容易陷入區部最佳化的陷阱。

2)種子必須均勻分散於超圖中以獲得更佳的分割結果。

在決定了種子後，本章的叢聚法即一次選擇一個與完成分割的節點集 (此即節點叢的集合) 有最大連接性 (以IOC衡量) 的節點，再將之指定至與它有最大關連性 (以CV衡量) 的節點叢中。重覆上述步驟直至完成分割為止。此演算法的時間複雜度為$0(n^2)$。

為了評估這個方法的優劣，此處測試了20個大小不一的問題，並且將其結果與傳統採用圖或加權圖的結果相較於表3.1中。結果顯示採用超圖結構所得到的解答比起採用圖結構平均減少53%的割線數，比起加權圖也可以減少17%至32%的割線數。

表3.1：超圖、圖、及加權圖的比較

例題	圖	加權圖 W = 2/(n-1)		加權圖 W = 2/n		加權圖 W = 2/(n^2-n%2)		超 圖	
	割線	割線	縮減百分比	割線	縮減百分比	割線	縮減百分比	割線	縮減百分比
1	6	3	50	3	50	2	66.6	2	66.6
2	7	6	14.3	6	14.3	5	28.8	4	42.9
3	12	6	50	8	33.3	9	25	5	58.3
4	21	9	57.1	8	61.9	12	42.9	6	71.4
5	10	11	-10	10	0	11	-10	8	20
6	18	8	55.6	8	55.6	8	55.6	4	77.8
7	22	10	54.5	9	59.1	6	72.7	7	68.2
8	25	18	28	16	36	16	36	12	52
9	13	14	-7.7	13	0	10	23.1	5	61.5
10	19	10	47.4	6	68.4	13	31.6	8	57.8
11	19	15	21	15	21	16	15.8	10	47.4
12	32	25	21.9	23	28.1	22	31.3	20	37.5
13	24	20	16.7	21	12.5	19	20.8	9	62.5
14	21	20	4.8	20	4.8	14	33.3	10	52.4
15	14	15	-7.1	16	-14.3	17	-21.4	10	28.6
16	34	33	2.9	25	26.5	18	47.1	14	58.8
17	18	28	-55.5	22	-22.2	25	-38.9	15	16.7
18	28	31	-10.7	28	0	14	50	14	50
19	37	31	16.2	33	10.8	17	54.1	17	54.1
20	37	33	10.8	27	27.0	29	21.6	17	54.1
總和	417	346	17.0	317	24.0	283	32.1	197	52.8

n is the size of a net

第四章　晶片層次的元件佈置法

雖然晶片的設計有各種不同的型式，如閘陣列、標準元、及一元閘列法等，這些不同的設計事實上都可以用線性佈置法完成。

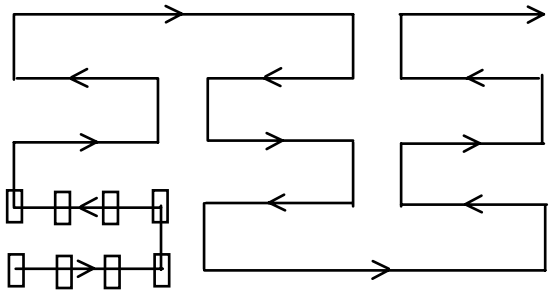


圖4.1：晶片設計的線性佈置法

線性佈置問題可以定義如下:

給定一超圖形，將節點做適當的排列，以使得繞線所需的道路(trucks)數為最少。

這個問題根據對所需道路數的評估方式不同而分成兩個獨立的問題，分別稱為**線性排列**問題及**一維閘列**問題。在線性排列的問題中，道路數等於最大割所通過的超邊數。圖4.2為一典型的例子，在這個例子中導線共佔有7條道路。同樣的接線關係但若為一維閘列的設計，則需要8條道路，如圖4.3所示。由此例可知在一維閘列的設計中，所需道路數通常大於最大割所通過的超邊數。此外在一維閘列的問題中，最左及最右的節點為

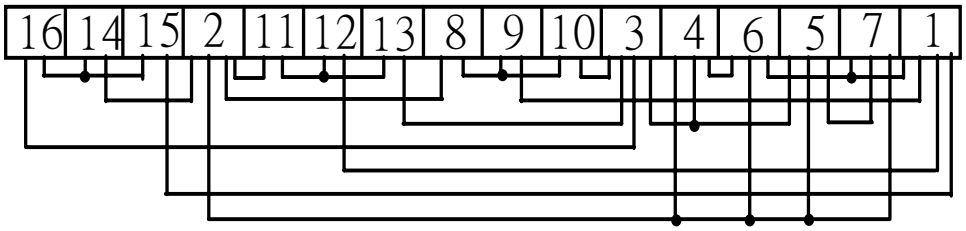


圖4.2：線性排列的例子

I/O節點故必須固定於兩側，但在線性排列的問題則沒有這個限制。

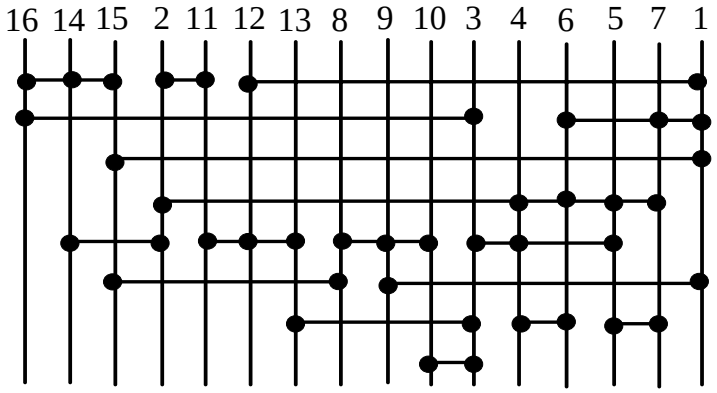


圖4.3：一維閘列的例子

4.1、節點合併演算法

節點合併演算法較適於求解線性排列的問題。本方法包括一連串的節點合併過程。每一次的節點合併都包含三個步驟。首先，比較所有相鄰節點的連接性，並選出連接性最強的一對節點；其次，排列此兩節點間的次序，以使得兩者間所需的道路數(即割線數)為最少，並將此次序儲存於次序樹中；最後，將此兩節點合併使得超圖的節點數目比原先的數目少一個。由於這個方法在選擇合併的節點時考慮超圖的整體接線關係，就如同由上而下的方法一般，故而可以避開局部最佳化的陷阱。此外，每一次合併後超圖的節點數就減少一個，如同由下而上的方法一般，故此方法的效率也非常高。

在表4.1中比較了此方法與其它三個知名方法—cluster growth algorithm [Kan83]、graph partitioning [Che84]、和 global method [Che87]。在所比較的例子中，節點合併演算法都得到最好的解答。

表4.1：節點合併演算法與其它三個方法的比較

演算法	例1: 9 個節點， 16 條導線			例2: 16 個節點， 17 條導線			例3: 31 個節點， 33 條導線		
	道路數	繞線長	CPU(s)	道路數	繞線長	CPU(s)	道路數	繞線長	CPU(s)
Vertex Combination	11	50	< 0.1*	7	77	0.1*	4	90	0.4*
Cluster Growth	11	50	< 0.1*	7	83	< 0.1*	5	96	< 0.1*
Graph Partitioning	11	50	2.1**	8	73	2.7**	5	91	7.3**
Global Method	11	50	1.6**	8	79	1.7**	8	102	4.6**

* Run on a SUN SPARC I work-station.

** Run on a VAX 11/780 machine.

4.2、去叢聚演算法

傳統的線性佈置法是一個最小化的問題，其目的在使道路數為最少。由於此問題屬於NP-hard的問題之一，故實際上沒有一個演算法可以在合理的時間內得到最佳的解答，取而代之的是用循序最佳化的方法求次佳解。但使用循序最佳化的方法求解時，由於每個步驟都必須要求得到最少的道路數，此種貪婪的特性使得求解過程非常容易落入區部最佳化的陷阱，以至於無法得到最佳解。為此，本章重新考慮線性佈置法如下：

給定一超圖形及道路數上限，將節點做最佳化的排列以得到不超過此上限的解答。如果得不到解答，則逐漸放鬆此上限直到得到解答為止。

於是線性佈置的問題由**最小化**的問題變成**限制最佳化**的問題。當採用循序最佳化的步驟求解時，不再受制於最小化道路數的限制，此時的目標變成如何使剩下的問題更容易在此道路限制下完成。於是此道路限制就成了一個宏觀的指引，它可以協助循序最佳化的步驟避開局部最佳化的陷阱。

去叢聚演算法由兩個邊界節點開始，一次選取一個節點，並指定它屬於左邊或右邊的集合。所挑選的節點是與左或右邊的集合有最大連接性且滿足道路限制的節點。如果找不到符合條件的節點，表示求解失敗，必須進一步放寬道路限制，並重新求解。由於去叢聚法必須由兩個邊界節點開始，因此較適於求解一維閘列的問題，但是若線性排列的問題其邊界節點已知，則亦可用此法求解。

表4.2、4.3和4.4比較去叢聚法及其它知名的構成演算法[Fuj87, Che87, Yam89, Hon89]。由表中的結果可知：去叢聚法不但求解的速度很快，而且所得到的解答也優於其它的構成演算法，但略遜於模擬退火法。由於去叢聚法採用貪婪的策略，而且步驟簡單，所以它的速度很快是很容易理解的，但是為什麼它所得的結果可以優於其它更複雜的方法？最主要的原因就在於道路限制發揮了宏觀指引的效果。

表4.2的結果也指出：在較小的電路中(閘數在10左右)去叢聚法可以得到最佳解；但在較大的電路中則仍難避開局部最佳化的陷阱。那麼是否能夠藉由適當的分割將大電路分割成小電路後，再用去叢聚法求解以得到較佳的答案？根據這個想法，

表4.2：去叢聚法與其它方法的比較

例題	閘數	導線數	去叢聚法			其它的方法		最佳解	
			道路數	繞線長	CPU (s)	道路數	繞線長	道路數	繞線長
1	8	8	3	12	0.01	3	12	3	12
2	9	9	4	17	0.012	4	18	4	17
3	9	8	4	24	0.018	4	27	4	24
4	9	7	5	25	0.018	5	26	5	25
5	10	11	4	25	0.017	4	25	4	25
6	10	11	4	25	0.018	4	26	4	25
7	9	21	11	50	0.038	11	51	11	50

表4.3：去叢聚法與Hong等人的方法及模擬退火法的比較

例題	閘數	導線數	去叢聚法			Hong et al.		模擬退 火法	
			道路數	繞線長	CPU (s)	道路數	繞線長	道路數	繞線長
4	9	7	5	25	0.015	5	26		
8	15	18	7	71	0.038	7	71	7	71
9	29	37	13	265	0.153	13	254	13	245
10	48	48	12	371	0.2	13	383	11	357

表4.4：去叢聚法與其它方法的比較 .

例題	閘數	導線數	去叢聚法			Yamata et al.			Fujii et al.			Cheng		
			道路數	繞線長	CPU* (s)	道路數	繞線長	CPU+ (s)	道路數	繞線長	CPU+ (s)	道路數	繞線長	CPU⊗ (s)
3	9	8	4	24	0.018				4	27	N.A.			

11	15	79	20	116	0.125							21	112	N.A.
12	16	17	8	70	0.033	8	71	0.2	8	79	1.5	8	73	2.7
13	31	33	6	92	0.055	6	94	0.7	6	106	3.0	6	91	7.3
14	85	96	22	1379	1.028	23	1470	4.7	32	1968	9.3			

* Run on a SUN SPARC I work-station.

\+ Run on a SONY NEWS 830 work-station.

⊗ Run on a VAX 11/780 machine.

本文提出了**遞迴去叢聚法**。這個方法重覆的使用**比例分割法**(ratio-cut method)將原來的問題分割成兩個較小的問題，再使用去叢聚法求解，若新解優於舊有的解答則取代之。表4.5比較了去叢聚法、遞迴去叢聚法、及模擬退火法所得之結果。由表中發現：遞迴去叢聚法可有效的改善去叢聚法的解答，而且所得到的解與模擬退火法所得之答案[Hon89]有相同的最大道路數。由於遞迴去叢聚法主要應用於改善舊有的答案，所以線性排列及一維閘列的問題均可適用。

表4.5、去叢聚法、遞迴去叢聚法、及模擬退火法的比較

例題	去叢聚法			遞迴去叢聚法			模擬退 火法	
	道路數	繞線長	CPU(s)	道路數	繞線長	CPU (s)	道路數	繞線長
9	13	265	0.153	13	257	0.853	13	245
10	12	371	0.2	11	360	1.47	11	357
13	6	92	0.055	6	90	0.173		
14	22	1379	1.028	20	1314	10.857		

第五章　混成電路的元件擺置法

在晶片設計中，由於元件的發熱量都很低，而且彼此間的發熱功率差異亦不大，所以通常僅需考慮總發熱量而無需考慮元件間發熱量的不同，也因此在元件擺置時可以暫時不考慮元件的發熱量，而僅需考慮繞線性。但在混成電路的設計中，由於元件間發熱量大小的差距極大，所以在元件擺置時必須將此發熱量的差異加入考慮。此外，由於混成電路都是封裝起來的，所以它的發熱量必需依賴熱傳導(conduction)的方式傳送到外界。因此，高發熱元件無可避免的會將部份熱量傳送至周遭發熱量較低的元件。也因此，在混成電路的佈局中，不單要考慮繞線的好壞，溫度分佈的均勻與否也必須一併考慮。然而由於熱傳導的方程式是一個橢圓型的偏微分方程式，因此唯有當所有的元件都完成擺置時，才能正確的求出溫度分佈。這個特性使得循序擺置的演算法幾乎無法應用在混成電路的設計上，因為在擺置的過程中無法預估最後的結果在溫度分佈上的優劣。

理論上，迭次改善的演算法 (iterative improvement algorithms) 似乎可以應用在混成電路的設計上，但實際上由於每次擾動解答都必需重新分析它的溫度分佈，而溫度的求解又必需大量的計算時間，因此迭次改善的演算法也無法應用在實際的問題上。

為了克服混成電路設計上此種先天難纏的特性，本文提出了一個階層性的元件擺置法。此法先擺置發熱量大的元件(稱為熱元件)以達到溫度分佈均勻的要求，再擺置發熱量小的元件 (稱為冷元件) 以滿足易繞線的目標。

5.1、熱元件的擺置

在此階段，每一個熱元件首先與周遭的冷元件合併成熱叢，熱叢中元件的數目與熱元件的發熱量成正比，合併所用的演算法，即第二章所介紹的叢聚法。在完成叢聚後，原來的超圖即形成一個新的、退化的超圖。接著對此超圖實施**四裂分割**(quadrisection)，將它分割成四個發熱量相當的子圖，並將之指定至基板上，每個子圖所佔的面積與其發熱量成正比。接著對每個分割後的子圖遞迴地執行四裂分割，直到每個子圖均僅包含一個熱叢為止。最後再將每個熱元件置於熱叢所擁有的面積之中心處，熱元件的擺置即告完成。

5.2、冷元件的擺置

冷元件的擺置採用**力指向法**(Force-directed algorithm)[Qui79]求得元件間的相對位置，在此階段熱元件的位置均視為固定。

5.3、實體擺置

在完成了元件間的相對擺置後，使用交談式的電腦輔助程式對元件執行實體佈置。在這個階段程式會計算每一個新的元件擺置的溫度分佈、系統可靠度、及繞線長度等特性。

表5.1 針對兩個不同的電路比較了用本文的方法(階層擺置法)及力指向法所得到的擺置結果。結果顯示對此兩電路採用階層擺置法較用力指向法可分別提昇28.5%及38.5%的可靠性，而總繞線長度僅分別增加了9%及10.7%。

表5.1：比較階層擺置法及力指向法所得的結果.

項目	例題 1		例題 2	
	力指向法	階層擺置法	力指向法	階層擺置法
最高溫	126 ℃	118 ℃	135 ℃	126 ℃
最低溫	80 ℃	86 ℃	60 ℃	64 ℃
最大溫差	46 ℃	32 ℃	75 ℃	62 ℃
總繞線長	992	1081	1600	1771
可靠性	133.0 f/M	95.1 f/M	89.2 f/M	54.9 f/M

第六章　印刷電路板的元件擺置

印刷電路板的元件擺置問題本質上與混成電路相似，不但要考慮隨之而來的佈線的困難度，也必須考慮散熱的需要。不同的是印刷電路板通常採用強制對流(forced convection)的冷卻方式。由於印刷電路板的發熱量經由強制對流帶走的熱量通常遠大於經由熱傳導(conduction)帶走的，因此經由熱傳導帶走的熱量常被忽略。由於強制對流的方程式是屬於拋物線型的偏微分方程式，故下游元件的發熱量不會傳遞至上游的元件。因此當須要計算某元件的溫度時，只要上游的元件已完成擺置，即可利用熱尾流函數(thermal wake function) [Mof92, And92a, And92b]求得。為了利用這個特性，本文將二維的元件佈置問題以一維折疊的方式完成，如圖6.1所示。在這個方法中元件的擺置由上游向下游逐次完成。同時，在擺置的過程中部份解必需滿足三個限制條件，祇要有任何一個條件不滿足，此部份解即放棄以減少不必要的求解過程。這三個限制函數一個用來評估繞線的優劣，一個評估可靠性的好壞，另一個評估此部份解將來完成後的解答優於目前最佳解的可能性之大小。

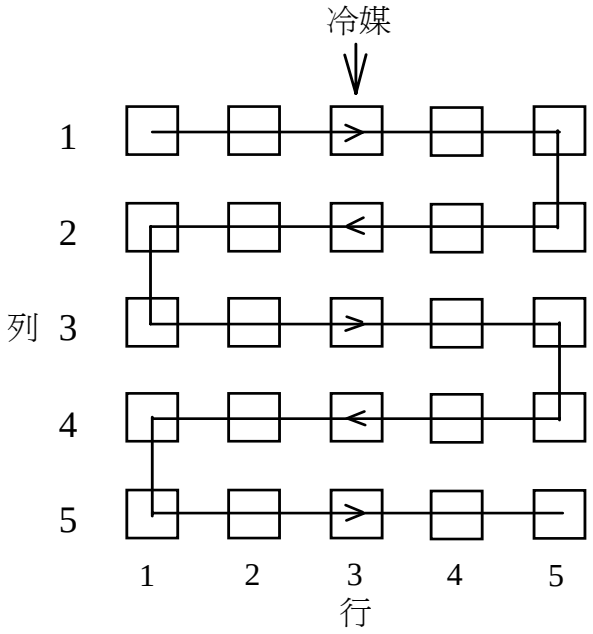


圖6.1：以一維折疊的方式完成二維的元件佈置

由於本章的方法基本上屬於**回溯**(backtracking)演算法，所以非常的耗時，為減少所需的時間，本文使用一個優先序函數評估每個元素放在上游的優先性。表6.1列出根據此規則得到的第一個解答，由表中的結果知道第一個解答的可靠性相當高，因此若不執著於最佳解，用本章的方法可以在很短的時間內得到相當好的解答。

表6.1：第一個解與最佳解的比較

	例題 1	例題 2	例題 3
最佳解的可靠性	2.159 f/M	6.034 f/M	25.750 f/M
第一個解的可靠性	2.171 f/M	6.073 f/M	26.088 f/M

第七章　迭次改善演算法

由於元件擺置屬於NP-hard的問題[Gar79]，因此在實際問題的解決上，常常先用構成演算法得到一個不差的初解，再利用迭次改善演算法獲得更佳的解答。迭次改善法每次均擾動舊有的解答以獲得新的解答，**傳統的迭次改善法**唯有當新解優於舊解時才取代舊解，此種貪婪的特性使得傳統的迭次改善法容易陷入區部最佳化的陷阱，而無法獲得真正的最佳解。**機遇型的迭次改善法**則允許在求解的過程接受較差的解以逃出局部最佳解的陷阱，但付出的代價則是大幅增加了求解所需的時間。然而隨著電腦的速度愈來愈快及演算法的改進，機遇型的迭次改善法已快速的取得了主流的地位。本章介紹了包括simulated annealing、sequence heuristic、stochastic evoluation、genetic algorithm、simulated evoluation、hybrid heuristic 等機遇型的迭次改善演算法。

第八章　散熱設計

無論何種類型的電路，散熱設計都屬於關鍵的一環。半導體裝置的功耗可區分為內部功耗及外部負載所引起的功耗兩部份。而外部功耗又分為直流及交流兩部份。功耗產生的熱量必須有效的傳送至外界，以免晶片因溫度過高而損壞。

近年來，一方面電路一再的積集化以增加及強化電路的功能，另一方面又要求降低晶片的溫度以增加可靠性，這使得散熱設計更加的複雜及困難。

電子裝置的熱阻可粗分為內熱阻及外熱阻兩部份。內熱阻指半導體的pn接面或熱源至封裝外殼的熱阻，當電路封裝完成後此值即告確定。因此欲降低內熱阻唯有循改變封裝方式著手。外熱阻指封裝外殼至外界的熱阻，由於外熱阻視冷媒、散熱面積、及傳熱係數而定，因此可藉由各種不同的熱控制技術降低外熱阻，最常見的方式就是採用散熱片以增加散熱面積，或直接針對熱源使用風扇或致冷器冷卻(如Pentium的冷卻方式)。這些方式不但增加了系統的成本，也增加了系統的體積。由於很多系統散熱不良的原因，只是由於冷媒未能充分混合所致，故只要能夠設法使冷媒充分的混合即可大幅提昇散熱能力，而無需增加散熱片或致能器之類的裝置。本章所探討的**渦流產生器**即為一可以有效混合冷媒的裝置。實驗結果顯示裝有渦流產生器的元件可以增加20%至60%的熱傳係數。事實上渦流產生器不但可以增加區部地區的散熱能力，它也可以增加整體的熱傳係數。

第九章　可靠性評估

正確的評估電子裝置的可靠度是非常重要的。特別是在電子裝置的開發階段，常常存在數個可能的設計，若無法正確的評估出它們的可靠性，往往就無法做出正確的選擇。然而要正確的評估電子裝置的可靠性，特別是在產品尚未製造出來之前，是非常困難的，因為電子裝置的可靠性不但與溫度息息相關，更與製造的品質、電源的電壓、製程的成熟度及電子裝置的工作環境有關。

本章介紹美國軍方評估電子裝置可靠性的方法，即MIL-HDBK-217的規定，以供參考。在MIL-HDBK-217的規範中，元件可靠性的模型(壓力模型)如下：

$$\lambda = \Pi_Q(C_1\Pi_T\Pi_V + C_2\Pi_E)\Pi_L$$

其中Π_Q為品質因數，C_1和C_2為失敗率係數，Π_T為溫度加速係數，Π_V為電源電壓的因數，Π_E為環境因數，Π_L為元件的學習係數。由於此模型考慮了大部份影響可靠性的原因，因此可以得到相當正確的評估結果。

第十章　結論

本論文主要探討電路元件的擺置問題，內容涵蓋印刷電路板的擺置問題，混成電路的擺置問題、及積體電路的擺置問題—如標準元、閘矩陣、及一維閘列等設計法。此外，本文亦探討構裝的冷卻設計及電子裝置的可靠性分析等重要的課題。

第一章說明了三個不同層次的元件擺置問題及設計時必須考慮的因素。第二章介紹了電路的抽象模型及繞線長度的評估法。第三章處理電路分割問題，本章發展了超圖形的叢聚法以解決分割問題，這個方法也與傳統採用圖形或加權圖的方法相比較，結果顯示超圖形的叢聚法優於這些傳統的方法。第四章探討晶片上元件的擺置問題，本章採用線性擺置法設計包括標準元、閘矩陣及一維閘列等不同型式的電路。這類問題根據對繞線軌道數的評估方式不同而分成線性排列問題及一維閘列問題。針對線性排列問題，本文提出一個節點合併演算法，這個方法兼具由上而下法及由下而上法的優點，所以能夠很有效率的得到高品質的答案，實際的電路設計發現這個方法所得到的解答優於其它構成演算法所得到的答案。關於一維閘列的問題，本文不同於傳統的方法將之視為最小化道路數的問題，而將之轉換成在給定道路數的限制下完成元件的擺置問題。對此新問題本文提出去叢聚法及遞迴去叢聚法求解，所得的結果不但優於其它構成演算法的解答，而且與模擬退火法所得到的結果相當。第五章處理混成電路的元件擺置問題，此問題與晶片層次的元件擺置問題的最大不同點在於它的高發熱功率，使得在元件擺置時，不但要考慮繞線性，還須要將高發熱的元件分散以避免基板上的溫度不均。本章發展出一個兼顧易繞線性及高可靠性的元件擺置法，此方法先擺置發熱量大的元件使它們分散，而後再根據繞線最佳化的原則擺置其它元件。在所測試的例題中發現此方法所得的結果優於力指向法所得的結果。第六章探討印刷電路板的元件擺置問題。與混成電路的元件擺置問題一樣，最佳化的目標包括了易繞線性及高可靠性。本章使用回溯演算法處理此雙目標最佳化的問題。為加速求解的速度本文使用一個優先序函數決定元件被置於上游的優先程度，並利用三個限制函數刪除不必要的搜尋空間。第七章介紹傳統的迭次改善演算法與機遇型的迭次改善演算法。機遇型的迭次改善演算法內容包括 Simulated annealing, sequence heuristic, Stochastic evoluation, genetic algorithm, simulated evoluation, hybrid heuristic 等演算法。第八章探討熱控制技術，內容著重於以外在被動的裝置降低外熱阻。本章所探討的裝置為渦流產生器，實驗結果顯示渦流產生器不但能大幅提昇局部區域的冷卻能力，也可以增加整體的散熱能力。第九章介紹如何評估電子裝置的可靠性。本章介紹美國軍方的標準，這個標準不但考慮溫度的效應，也考慮製造技術的品質及使用的環境等因素。