ICCAD 2024 CAD Contest
Problem D: Chip Level Global Router

聯發科技（Mediatek）

1 Introduction

隨著製程技術進步，當代電路中可能會包含上億個電晶體，為了能有效降低複雜度，現今的實體設計流程通常會採取階層式的設計風格。當平面規劃完成後，晶片將會被區分成 chip level 和 block level。一旦 block 的形狀和位置決定後，剩下的區域即為 top channel 可繞線的部分，其中 chip level 的繞線區域的規劃是相當重要，因為各個 block 之間必須透過 channel 內的繞線進行溝通，為了保證在給定的時程內完成整體佈局圖(layout)，必須調整 channel 的形狀以減少繞線 detour，使得繞線擁擠不會發生，同時為了能減少整體晶片面積， 必須盡量縮小 channel 的面積。 Chip level 可以由 1) channel, 2) feedthroughable block, 3) non-feedthroughable block, 4) region, 和 5) tile 五種型態的元件構成，它們分別的定義如下:

1. Channel: 整個 chip 扣除 block 和 tile 剩下的區域即為 channel。如圖 1 紅色虛線框。
2. Feedthroughable block: 允許 net 穿過的 block。如圖 1 深藍色虛線框。
3. Non-feedthroughable block: 除了 hard macro feedthrough (HMFT)外，不允許一般 net 穿過此 block。其目的是為了能進一步縮減 channel 面積，因此允許某些特定的 net 穿過此類的 block。如圖 1 綠色虛線框。
4. Region: 除了 block 之外, chip level 中還有些四處散落的 standard cell，它們會依據 function 被分配到不同的 region 中。如圖 1 黃色框。
5. Tile: 由 region 和 block 構成，允許 net 穿過。由於 block 內部的實體設計和 chip level 的實體設計能夠平行進行，為了能夠進一步縮減 chip level 的執行時間，可能會將 chip 再進一步切割出許多 tile。如圖 1 所示，原本 chip level 是整個 die 扣除 block 和 non-throughable block 後剩下的部分，為了減省 chip level 實體設計的執行時間，會再進一步切出一些 tile，如圖 1 淺紅色的虛線框。

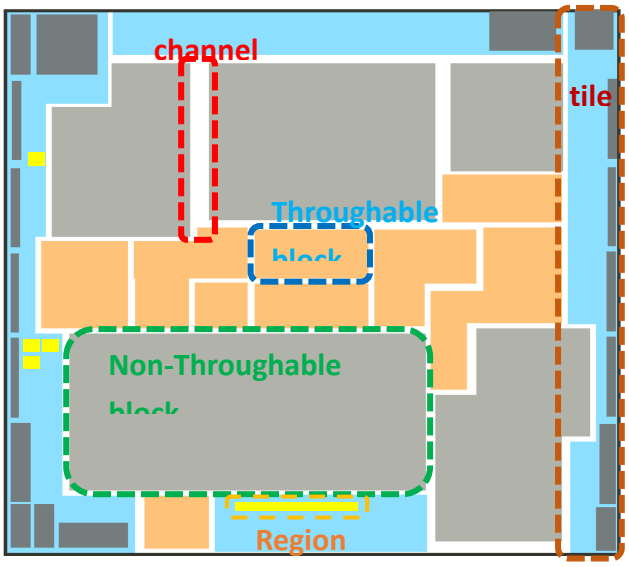


圖 1.

在晶片設計中設計一個輕巧快速且能最佳化繞線線長的 channel global router 是相當重要的工作，然而由於 chip level 的繞線區域和 block 內部的繞線區域差異很大，block 內通常能保留完整的繞線區域，然而 chip level 則只能使用 block 佔據後剩餘的面積進行繞線，因此 chip level 的繞線區域可能會由許多不規則形狀的 channel 繞線區域所構成，這使得 chip level 的繞線相對於 block 具有更大的挑戰性。

2 Problem Statement

在 chip level 進行全域繞線規劃時，必須考量四個重要的成本，分別是 1) channel overflow, 2) wirelength, 3) Edge Pin Density. 4) turn cost。這四項成本的定義如下:

(1) Channel overflow:一個 channel 的 overflow 定義如下:

$$(\text{number of occupied routing tracks})/(\text{number of availble routing tracks})$$，其中

i. number of available routing tracks: 單位面積中包含的 routing tack 數量，在實際設計中，此資訊可由 LEF 檔案中取得，本競賽將直接定義在每一組測試資料的輸入中。舉例來說，20 tracks/um 代表每一 um 中包含 20 條 routing tracks.

ii. number of occupied routing tracks: 每一種型態(type)的繞線會使用不同數量的 routing track,此資訊會定義在每一組測試資料的輸入中。舉例來說,一條訊號線(signal net)會占用一條 routing track, 而時鐘線(clock net)則會占用 3 個 routing track。

繞線時必須盡量減少 channel 的 overflow，因為越高的 overflow 會導致細化繞線 (detailed route)時 DRCs 的違反數量增加，使得晶片無法順利 tape out。

(2) Wirelength: 一條 net 中包含的所有 segment 長度的總和。越短的 wirelength 會占用較少的繞線資源，使得整體晶片成本下降。反之，過長的 wirelength 可能會導致 timing violation，使其功能發生錯誤。

(3) Edge Pin Density:一個矩形的 block 包含四個邊(edge)，每一個邊(edge)上的引腳 (pin)所佔據的密度，它的定義如下:

$$(\text{demands of nets in an edge of a block})/(\text{capacity of nets in an edge of a block})$$，其中

i. demands of nets in an edge of a block: 由程式決定。

ii. capacity of nets in an edge of a block: 定義在 benchmark 的輸入中。舉例來說，Block A (x0,y0)->(x1,y1) 1000, 表示在 block A 的(x0,y0)到(x1,y1)的邊上可以容納 1000 條繞線數。

過高的 pin density 可能會產生 DRCs 違反，也可能導致 block 的介面出現 timing violation，因為在 block 的邊外面可能不存在足夠的空間插入 buffer，這可能使得某一些 net 的 transition time 變長。在早期規劃時, 由於 block 的平面規劃尚未完成，因此會藉由控制通過 block 每一個邊上的引腳數量避開可能的風險區域。注意: 由於 region 的形狀可以改變，因此無需顧慮其 edge pin density。

(4) Turn Cost: 每一條 net 轉折的數量。由於 net 每一次轉折時都需要透過 via 連接，然而 via 相對於金屬線具有更高電阻值，因此 net 如果頻繁的轉折，會增加其繞線延遲，使其 timing violation 的機會增高。

3 Input File / Format

每組測試中包含三個檔案，其格式定義如下:

(1) DEF：描述電路實體佈局的狀態。請遵照 LEF/DEF Language Reference [1] 中 DEF 格式（副檔名為.def）之定義。

1. chip_top.def: 給定所有 block 合法擺置，內容中包含四種敘述 (i.e, VERSION、DESIGN、UNITS、及 DIEAREA)與一個 section (i.e., COMPONENTS)。
2. block.def: 描述在 chip_top.def 中使用的 block 形狀，內容中包含四種敘述 (i.e, VERSION、DESIGN、UNITS、及 DIEAREA)與一個 section (i.e., COMPONENTS)。

(2) CFG: 格式為 JSON (副檔名為.json)， 描述每一個 block 的狀態, 包含五種屬性，其定義如下:

1. Block_name: block 的名字，下例中為 AAA。
2. Through_block_net_num: 當扣除從此 block 出發或接收的繞線後，可以 feedthrough 此 block 的繞線數量上限。下例中 through_block_net_num 為 1000。
3. Through_block_edge_net_num:定義在一個 block 從(X0,Y0)到(X1,Y1)這個邊出發或接收以及 feedthrough 的繞線數量上限。下例中 through_block_edg_net_num 為 100。
4. Block_port_region: 定義在一個 block 邊上可以擺放引腳的矩形區域。下例中 block_port_region 的矩形區域，其左下角座標和右上角座標分別是(X0,Y0)和(X1,Y1)。
5. Is_feedthroughable:表示是否可以允許 net feedthrough 這個 block。其中，True 代表允許，False 則代表不允許。

For example，

{
“block_name”: AAA,
“through_block_net_num”: 1000,
“through_block_edge_net_num”: [(X0, Y0),(X1,Y1),100],
“block_port_region”:[(X0,Y0),(X1,Y1)],
“is_feedthroughable:True”
}

(3) Connection matrix: 格式為 JSON (副檔名.json), 描述每一條 net 的八種屬性, 其定義如下:

1. ID: net id。
2. TX: 定義繞線從哪一個 block 出發。下例中繞線的起點為 block A。
3. RX: 定義繞線的終點是哪一個 block。下例中下例中繞線的終點為 block B 和 C。
4. NUM: 定義這條線需要使用的 routing tracks。下例中定義其需要使用 10 個 track
5. MUST_THROUGH: 定義這條線必須經過的區域。下例中定義此繞線必須穿 block D 的 edge(X0,Y0)->(X1,Y1)和(X0’,Y0’)->(X1’,Y1’) 和 block E 的 edge(X2,Y2)->(X3,Y3)和(X2’,Y2’)->(X3’,Y3’)。見圖 2 所示
6. HMFT_MUST_THROUGH: 定義此繞線必須經過”non-feedthroughable”。下例中此繞線必須 block F 的邊 (X4,Y4) ->(X5,Y5)和(X4’,Y4’)->(X5’,Y5’)。見圖 2 所示。

7. TX_COORD: 定義繞線的起始點相對於 TX block 左下角座標的相對位置。舉例來說，block A 左下座標為(100, 200)，如果 TX_COORD 值為[10, 20]時，繞線的起始點絕對值座標為(110, 220) 。
8. RX_COORD:定義繞線終點相對於 RX block 左下角座標的相對位置。舉例來說，block B 左下座標為(500, 700)，當 TX_COORD 的值為[10, 20]時，繞線終點的絕對值座標即是(510, 720) 。

For example，

{

"ID": 0,

"TX": A,

"RX": [B,C],

"NUM": 10,

"MUST_THROUGH": [[D,(X0,Y0,X1,Y1),(X0',Y0',X1',Y1')],[E,(X2,Y2,X3,Y3),(X2',Y2',X3',Y3')]],

"HMFT_MUST_THROUGH":[[F,(X4,Y4,X5,Y5),(X4',Y4',X5',Y5')]],

"TX_COORD": [xa,ya],

"RX_COORD": [(xb,yb),(xc,yc)]

}

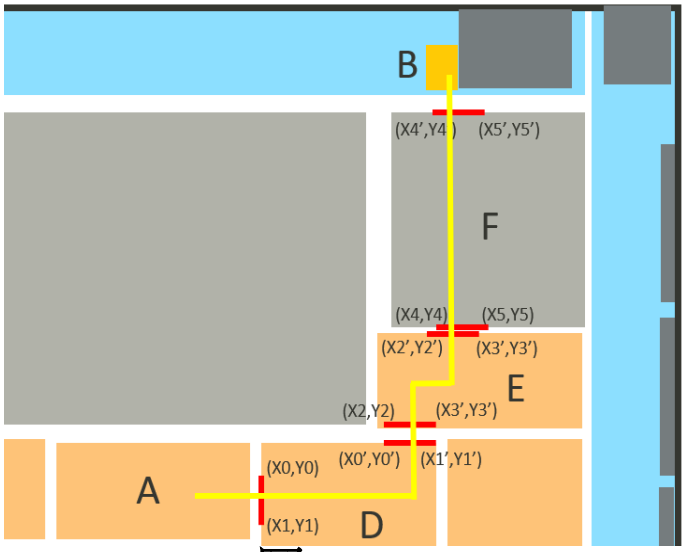
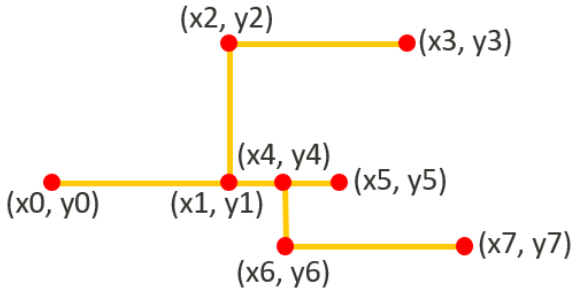


圖 2

4 Output File / Format

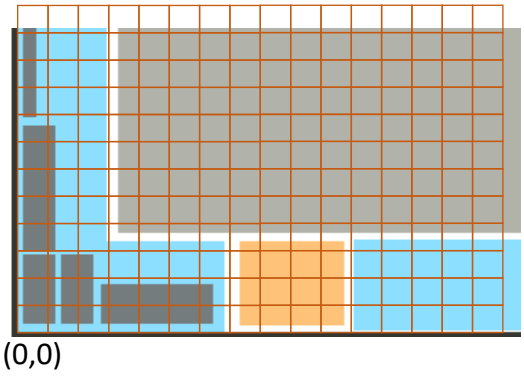
每組測試資料個別輸出一個檔案，檔名為 ***caseOO_net.rpt*** **(OO 為測試檔案編號)**，其檔案格式如下:

[ID]
(x0,y0),(x1,y1)
(x1,y1),(x2,y2)
(x2,y2),(x3,y3)
(x1,y1),(x4,y4)
(x4,y4),(x5,y5)
(x4,y4),(x6,y6)
(x6,y6),(x7,y7)
[ID 2]
(x0,y0),(x1,y1)
(x1,y1),(x2,y2)
.....



其中[ID]為某一條 net 的 id，每一條 net 將有許多的水平或垂直 segment 構成，一個 segment 中只包含兩個端點，舉例來說(x0,y0),(x1,y1)或(x1,y1),(x2,y2)分別代表 net 中某一個 segment 的兩個端點，如上圖所示。

為了評估繞線 overflow，會將整個 chip 從左下角原點(0,0)開始，將整個區域切割成數個相同大小的正方形框，這些框稱為 gcell，如右圖所示。當評估 overflow 時，程式會自動將每一個 segment 的座標平移到 gcell 的中心，其公式定義如下:



$$x_{modified} = \left(\frac{x}{width_{gcell}} + 0.5\right) * width_{gcell}$$

$$y_{modified} = \left(\frac{y}{width_{gcell}} + 0.5\right) * width_{gcell}$$

其中

i. (x, y)為 segment 中某一個端點原始的座標
ii. $(x_{modified}, y_{modified})$ 則為平移後的座標。
iii. $with_{gcell}$ 代表一個 gcell 的寬度， gcell 的寬度會由個別測試資料中單位面積中包含的 routing tack 數量 (i.e., tracks/um)和其包含的 net 形態中需要使用最多的 routing track 數量決定 (i.e., Max {NUM} ∀ net types)。舉例來說，某一個測試資料中單位面積可以繞 20 線 (i.e., 20 tracks/um), 而其所屬的繞線形態中最大的 NUM 值為 100, 則 gcell 的寬度則定義為 5um。

5 Usage Format

請使用 C 或 C++ 實作程式，執行檔取名為 "*CGR*"，並遵照下列的使用格式（*OO* 為測試檔案編號, XX 為整數）：

./CGR XX(tracks/um) caseOO.def caseOO.cfg.json caseOO.connection_matrix.json

6 Platform

OS: Linux

Compiler: gcc/g++

細節依大會公布

7 Testcases

提供 5 組開放測試資料及 3 組隱藏測試資料。

8 Evaluation

(1) 依照參賽者的總分由高至低排序，參賽者的總分為每組測試資料的得分加總。

(2) 每組測試資料的程式執行時間（包含讀寫檔案過程）必須在 2 小時內完成，否則不予計分。

(3) 每組測試資料產生的檔案格式必須符合 output format 定義，否則不予計分。

(4) 每條繞線的起始和終點位置必須正確，若繞線無法連接在給定的位置，則不予計分。

(5) 每條繞線必須符合"MUST_THROUGH"和"HMFT_MUST_THROUGH"的規定，否則不予計分。

(6) 若繞線經過 is_throughable 定義為 False 的 block，不予計分

(7) 若繞線不在 chip_top 內，則不予計分

(8) 若超過*"through_block_net_num"*和*"through_block_edge_net_num"*的限制，會對其結果酌量扣分。

$$score = 0.55 * cost_{overflowLength} + 0.35 * cost_{edgePinDensity} + 0.1 * e^{\frac{time}{2*60*60}} + 0.3 * penalty_{\#pin\_constraint} + 0.01 * penalty_{\#net\_turn}$$

其中

$$cost_{overflowLength} = \sum_{net}^{\#all\ nets} (\sum_{segment}^{\frac{netlength}{gcell_{width}}} \left(1 + \left(\left[\frac{\#used\ track}{capacity_{gcell_{edge}}} \right] > 0.7\ else\ 0 \right) \right) * gcell_{width}) / HPWL_{bbox\_net}$$

$$HPWL_{bbox\_net} = hpwl\ of\ bounded\ box\ of\ the\ net$$

$$cost_{edgePinDensity} = \sum_{block}^{\#all\ blocks} \sum_{edge}^{\#block\ edges} \sum_{segment}^{\frac{edgelength}{gcell\_width}} (\left[\frac{\#used\ track}{capacity_{gcell\_edge}} \right] > 0.6\ else\ 0)$$

$$penalty_{\#pin\_constraints} = \sum_{each\ constraint}^{all\ constraint} e^{(\left[\frac{\#used\ track}{\#pin\_constrains}\right] > 1\ else\ -inf.)}$$

$$penalty_{\#net\_turn} = \sum_{net}^{all\ net} e^{([\#net_{turn}] > 1 else -inf.)}$$

9 Reference

[1] LEF/DEF Language Reference, https://si2.org/oa-tools-utils-libs/