VLSI Physical Design Assignment5_V2.0

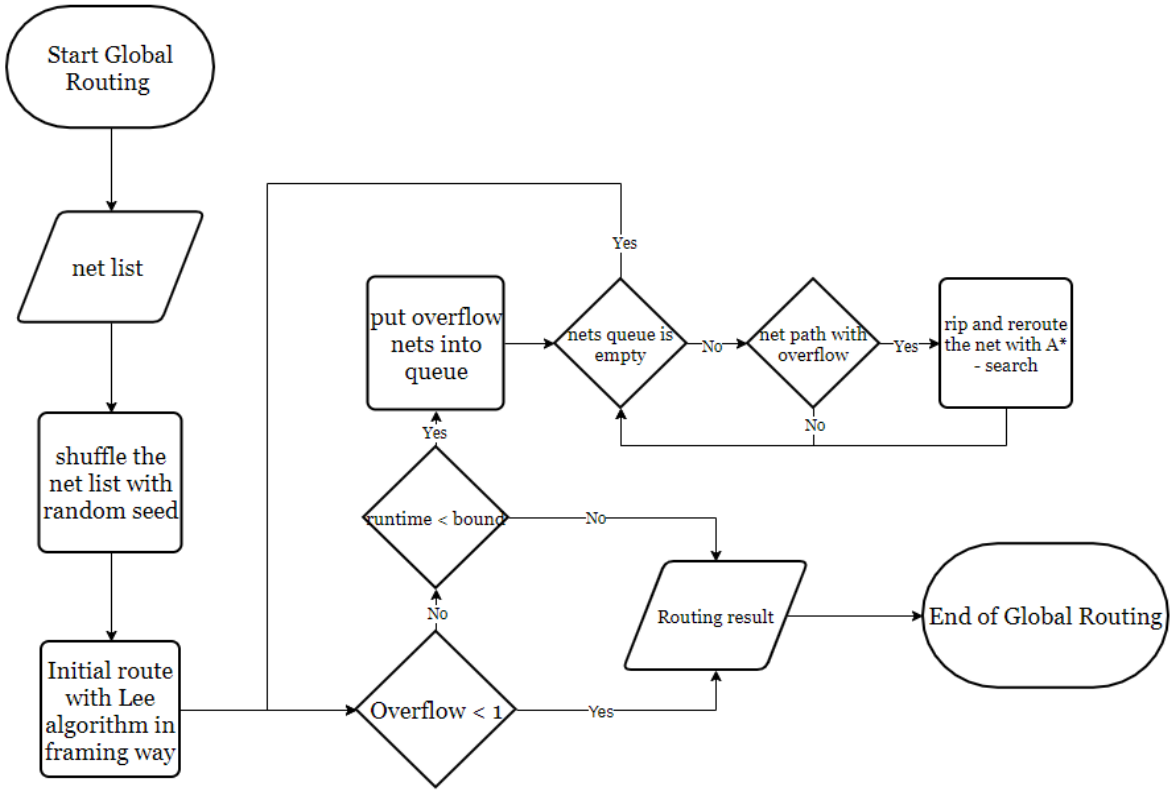
109065526 許立農

- 執行方法及範例參照 README。

- Result Table:

	ibm01	ibm04
Overflow	0	288
Total wirelength	66793	169980
Total Runtime	2.05(s)	61.03(s)

- Details of implementation:



首先，將 net 的資訊，包括起點、終點存入寫好的 class Net，並將所有的 Net 存入 vector<Net>。因為繞線時的順序會影響最終結果，所以將 vector<Net>做一次 random shuffle 才會進入 initial route 的部分。其中 initial route 是以 lee algorithm 並以 framing 的方式進行，這樣能避免 filling 多餘的區塊。完成後，檢查是否存在 overflow，若有 overflow 則進行 second route 的部分。進行 second route 前，首先要找出 channel 的瓶頸，以 overflow 最高之 channel 的最大 HPWL Net 為最優先處理對象排出 second route Net queue。依序進行以 A* search + channel utility 為基礎的 reroute，須注意若當前的 Net 因先前的操作而不再 overflow，則當前的 Net 無須進行拔線重繞。重複 second route，直到不存在 overflow 或到達時間限制。

- Enhancement:

  針對執行時間: 採用 framing 的 Lee algorithm 大幅加快了 Initial Route 的速度。

  針對成果品質: 嘗試過 3 種不同的 second route 模式。分析如下:

  1. Lee algorithm(framing) + utility based retrace
     雖然 Retrace 會考量所有方向的 utility，但是因為 framing 的關係，使得若 Bounding Box 中無可行解，必然會產生 Overflow。速度很快但 overflow 無法顯著下降。
  2. Dummy Point + Lee algorithm(framing)
     為了解決上述的問題，會將 overflow 的 Net 加上虛擬起點及終點，強迫路線先離開擁擠區域，並以虛擬起點及終點為 Bounding Box，進行 Lee algorithm。Overflow 有顯著下降，但因為 Bounding Box 的限制，有些 Net 會沒地方繞。
  3. A* search + Utility Cost Function
     上面兩種方法，都是要在範圍內尋找最短路徑。但有些路線必然要能夠 detour 才可以繞線成功。因此我使用了 A* search。依照課程的討論，只要 h(x) 不要高估都可以在範圍內找到最短路徑。在這邊我們將 h(x) 定義為 current node 到 terminal 的最長邊，加上 Utility Cost。若 Channel 已經 overflow 則會在 h(x)產生極大的 Penalty。依照這樣的原則製作 label map。就可以 Retrace 出一條 detour 的路徑。有很大的機會產出沒有 overflow 的 output。取決於 random shuffle 的 Net ordering。速度是這三種方法中最慢的。

- What I've learned:

  這一次的作業是我認為最難的一次，原因如下：

  1. 低估了 rip up and reroute 的難度，卡關非常久。
  2. 沒有與其他同學討論。
  3. 沒有設立 Checkpoint。
  4. 對於程式及時間的掌握能力不夠好，常常覺得自己修完這個 bug 就會有結果，但其實還有很多問題。

  針對這些問題，其實都有改善的空間。我認為最重要的是對於撰寫這種有規模的程式，利用 Git 來幫助記錄各版本之間的資訊迭代是很重要的。最後一點的話，我想或許我在排定流程時，必須幫自己加入一些緩衝空間，才能避免自己竟然過了 Deadline 還繳不出合格的東西。

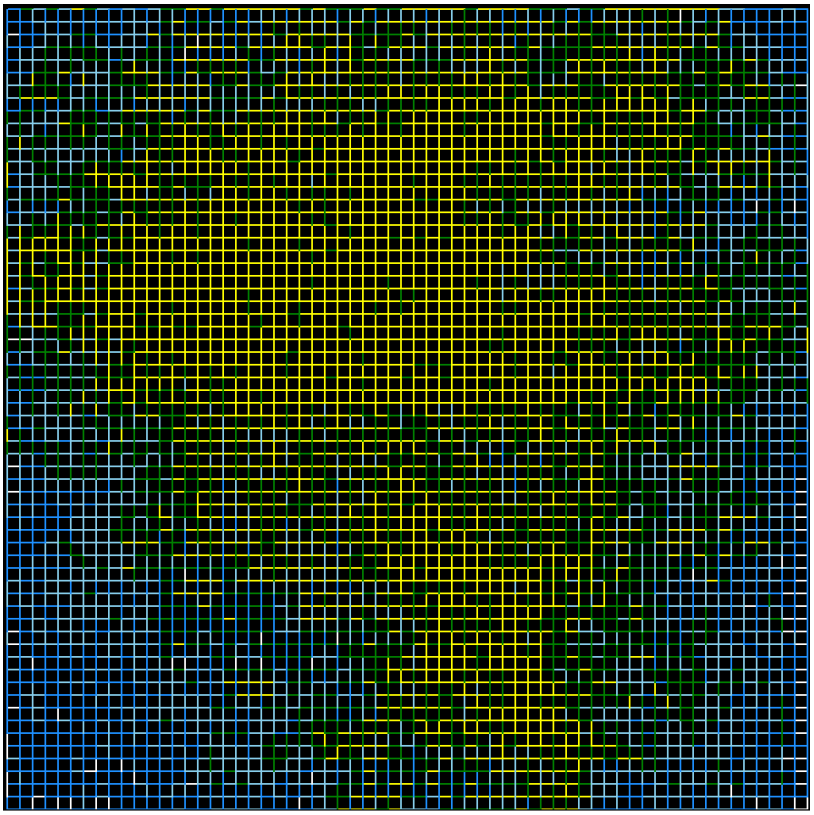
  學期總覽：

  從學期初到現在，經歷了每次作業的洗禮都覺得自己有所成長。漸漸熟悉一個 APR 工程師究竟面臨怎麼樣的問題，也逐漸長出自己的 Coding Style。學到了很多，例如 Makefile 的使用或是 ssh 遠端操作。這些都是

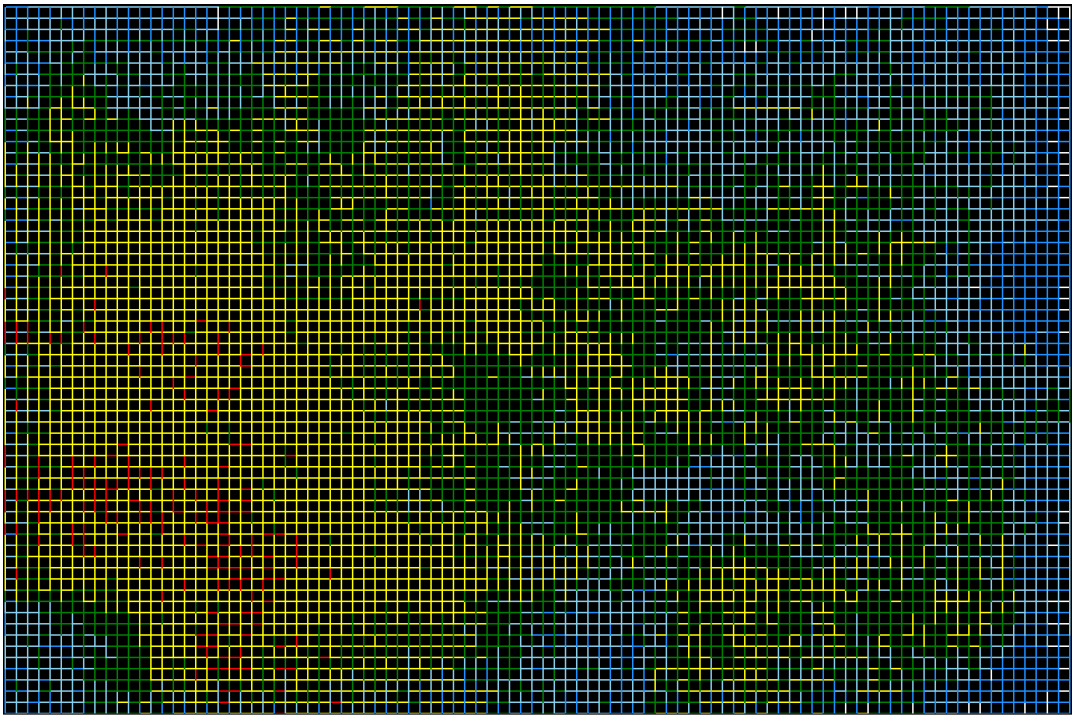
以往從未遇到的。我也期許自己在有空閒的時候，要把 Assignment2~5 修改到最簡潔且有效率的版本。

- Bonus:

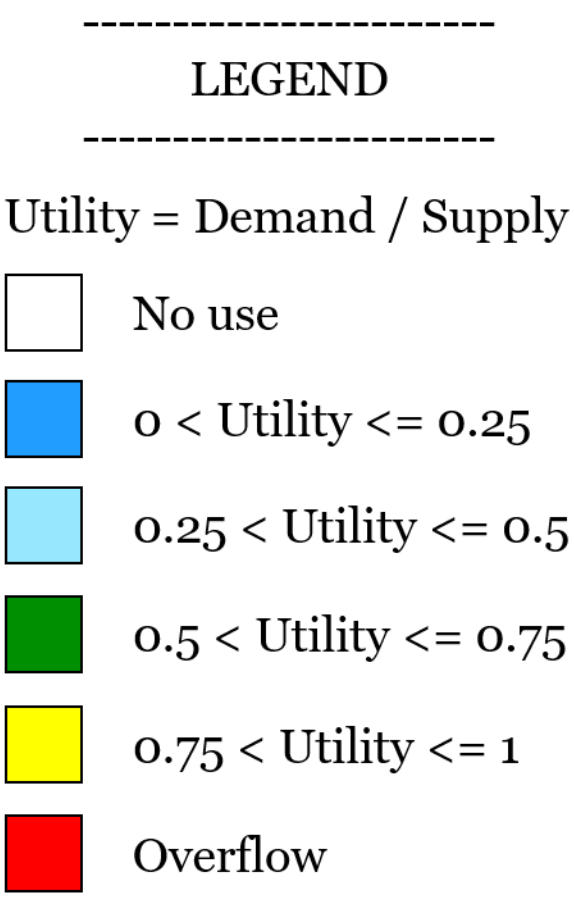
ibm01:



ibm04:



Legend:


LEGEND
Utility = Demand / Supply
No use
0 < Utility <= 0.25
0.25 < Utility <= 0.5
0.5 < Utility <= 0.75
0.75 < Utility <= 1
Overflow