# CEG5270 Assignment 2 (2009 Spring)

XIAO Zigang
zgxiao@cse.cuhk.edu.hk
Department of Computer Science and Engineering
The Chinese University of Hong Kong

March 26, 2009

1. First it is easy to know that for any pin pair, where one is in the top side and one is in the bottom side, they can use different vertical layers for routing and will not cause conflict, if there are two vertical layers. Then for the track assignment, since each net corresponds to a horizontal segment, it turns out to be problem which finds the chromatic number of a segment graph. It is proved in assignment I that greedy algorithm similar to LEA can produce the optimal result. Since we know that the lower bound of the channel width is the max of:

   (a) length of the longest path in VCG

   (b) channel density

   which also equals to the maximum clique in the segment graph. And we know that maximum clique size equals to chromatic number in the segment graph. Then we proved that LEA gives solution of minimum channel width.

2. At most $K = 4$ times of gain update is needed due to the change of a net.

   Since only critical nets will cause gain update, we only need to consider those nets which are critical before or after the move. We observe that once two cells on a net are moved to X and another two cells are moved to Y, they are locked and it is impossible to make the net to be critical anymore. Hence it is the bound of updates. The worst case is at most 4, if we assume every move bring changes before the net reaches the state mentioned above. For example, originally $X = \{A, B\}, Y = \{C, D\}$, by moving them in the order $A, D, B, C$ will produce 4 gain updates, but no further update anymore.

3. Without loss of generality, we assume the routing direction is anti-clockwise, For a multi-pin net i, we will break it into several 2-pin subnet and assign starting terminals to each subnet. For example, for net $b$, it is splited into $b1 \rightarrow b2$, $b2 \rightarrow b3$.

4. There are only three possible cases of exchange.They are:

   - Exchange a corner node with another corner node. This gives a gain of 0.
   - Exchange a corner node with a non-corner node.This gives a gain -1.

**Algorithm 1** Find Largest Routable Net Numer

**Require:** A list of net pins $plist$, the starting terminal of each net is known.

queue $\leftarrow$ plist // put all elements of list into a queue
**while** queue not empty **do**
  $p \leftarrow$ queue.front()
  $x \leftarrow$ pinOfNet(p) // check pin in which net
  **if** $p$ == $x$.startingTerminal **then**
    stack.push($p$)
    mark[$p$] $\leftarrow$ TRUE // mark starting terminal in stack
    queue.remove($p$)
  **else if** mark[$p$] == TRUE **then**
    // $p$ is not starting terminal, and starting terminal is in stack
    $j \leftarrow$ index of stack.top()
    **if** stack[$j$] == $x$.staringTerminal **then**
      // do not intersect with any nets
      stack.pop()
    **else**
      mark al the net between this net as conflict
      **while** stack[$j$] $\neq$ $x$.staringTerminal **do**
        $y \leftarrow$ pinInNet(stack[$j$])
        conflictList[$x$].add(stack[$j$])
        $j \leftarrow j - 1$
      **end while**
    **end if**
    queue.remove($p$)
  **else**
    queue.remove($p$)
    queue.pushback($p$)
  **end if**
**end while**
// now we have the conflict information, choose those net conflict with least nets
**return**

- Exchange a non-corner node with another non-corner node. This gives a gain -2.

We notice that none of them produces a positive gain, hence in the process of finding $k$ s.t. $G = g_1 + g_2 + ... + g_k$, the maximized value 0 should be reached when $k = n$. The algorithm stops here. It is a local minimum of K-L.