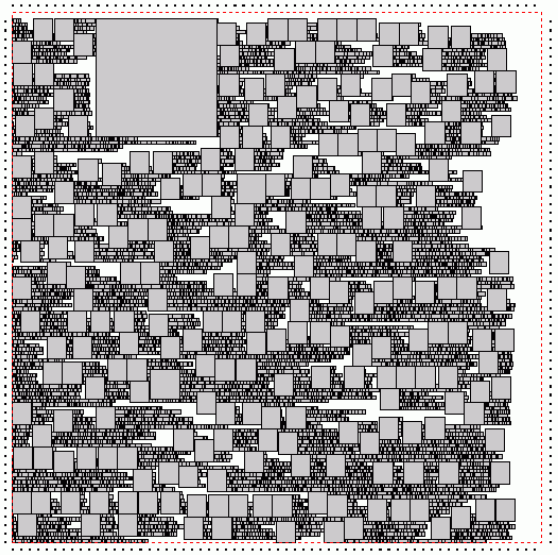
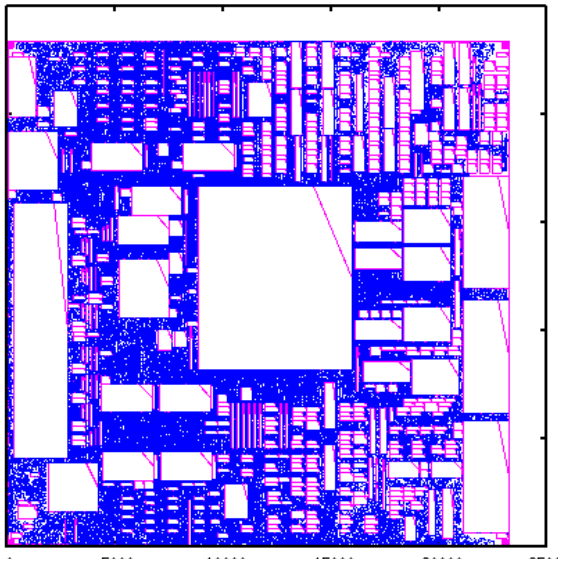
Unit 5: Placement

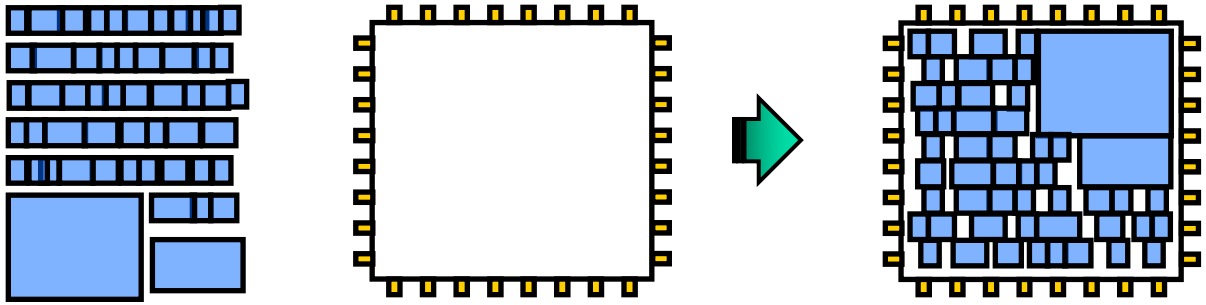
- Course contents:
  - Partitioning-based method
    - Min-cut placement
  - Nondeterministic methods:
    - Simulated annealing (SA)
    - Genetic algorithm (GA)
  - Analytical methods:
    - Force-directed placement
    - Quadratic placement (QP)
    - Non-quadratic placement
  - Mixed-size placement
  - Appendix: Linear assignment placement
- Readings
  - W&C&C: Chapter 11
  - S&Y: Chapter 4





Courtesy of Prof. Y.-W. Chang & H.-M. Chen

VLSI Placement



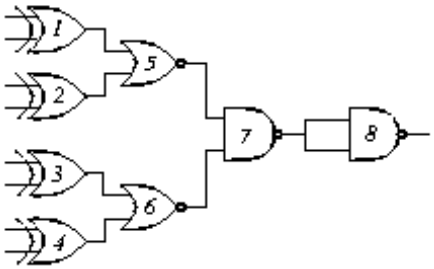
- **Placement:** assign cells to positions on the chip, such that no two cells overlap with each other (legalization) and some cost function (e.g., wirelength) is optimized.
- A major step in physical design that has been studied for 40+ years.
  - *EETimes* (4/10/2003): 1.46--2.38 X from the optimal wirelength
  - Is still far away from optimal??
- More than 16 new academic placers since 2000
- ACM ISPD Placement Contests in 2005 and 2006

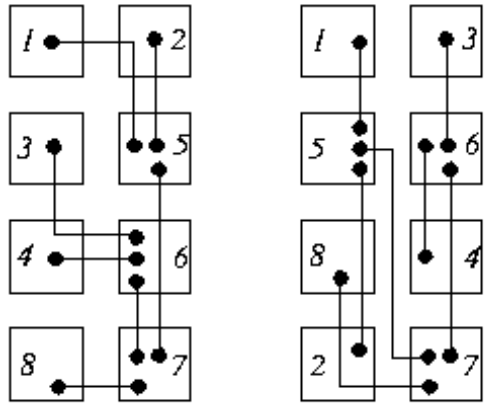
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement Problem

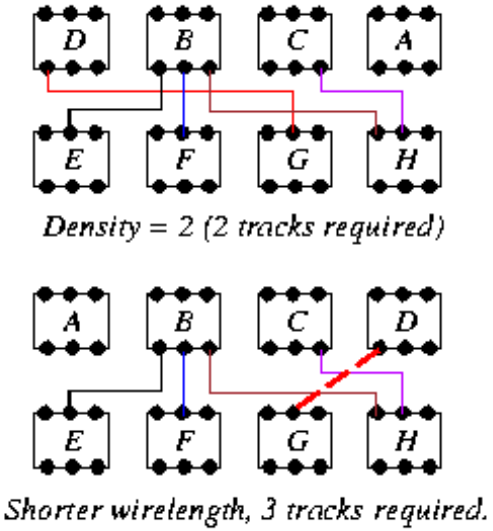
- Inputs: A set of **fixed** cells/modules, a netlist.
- Goal: Find the best position for each module on the chip according to appropriate cost functions.
  - Considerations: **routability/channel density**, **wirelength**, power, timing, thermal, I/O pads, manufacturability, etc.

placement也要考慮後續的routing可行性





相同的面積下，以wire length 評估，左側結果較好

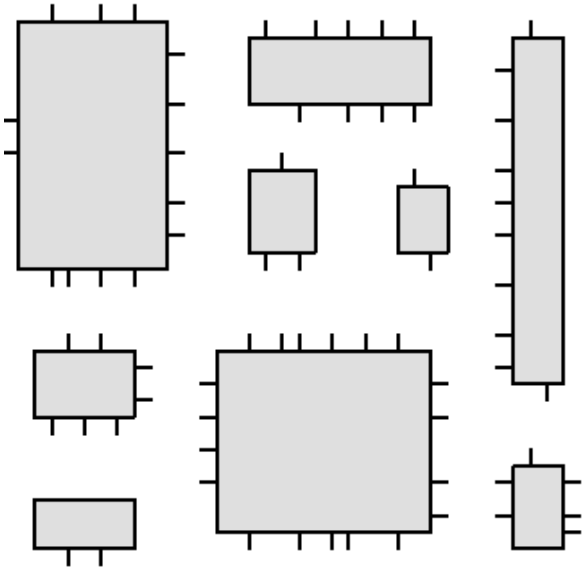


中間只有兩個channel的話，以下方的placement無法繞線

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement Style: Building Blocks

- Different design styles create different placement problems.
  - E.g., building-block, standard-cell, gate-array placement
- Building block: The cells to be placed have arbitrary shapes.
  - Similar to the floorplanning problem

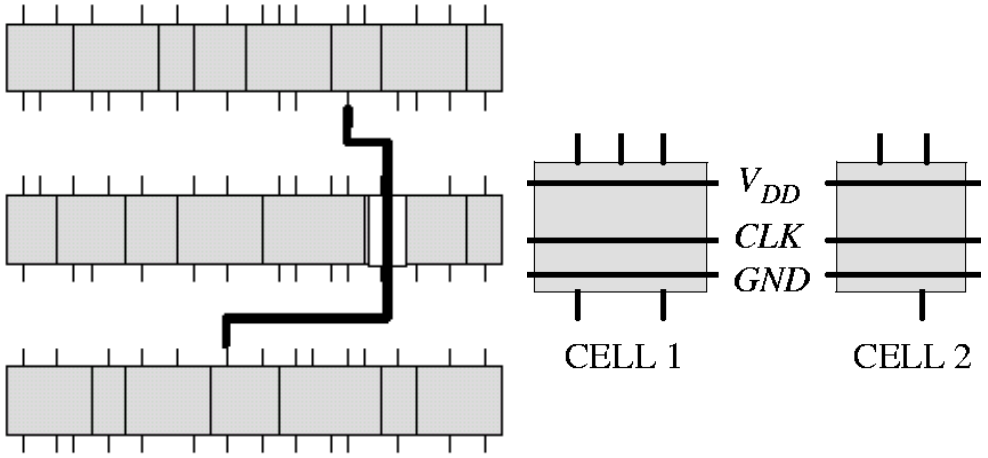


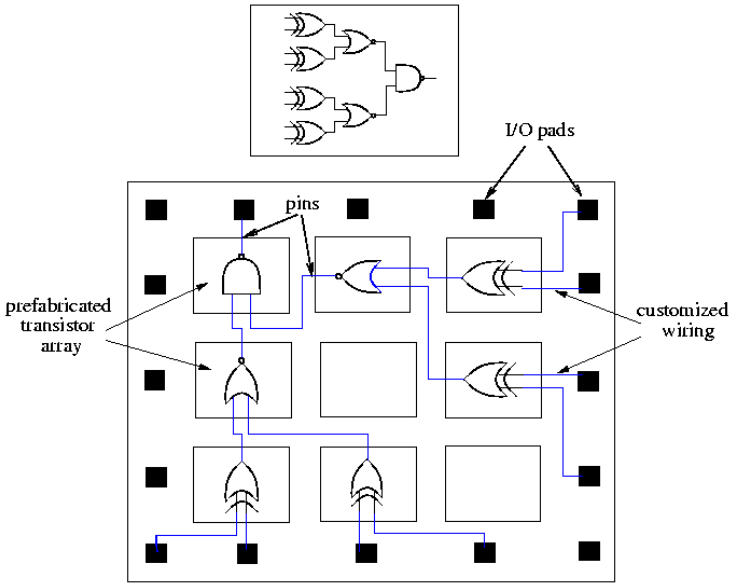
Building blocks

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement Styles: Standard Cell & Gate Array

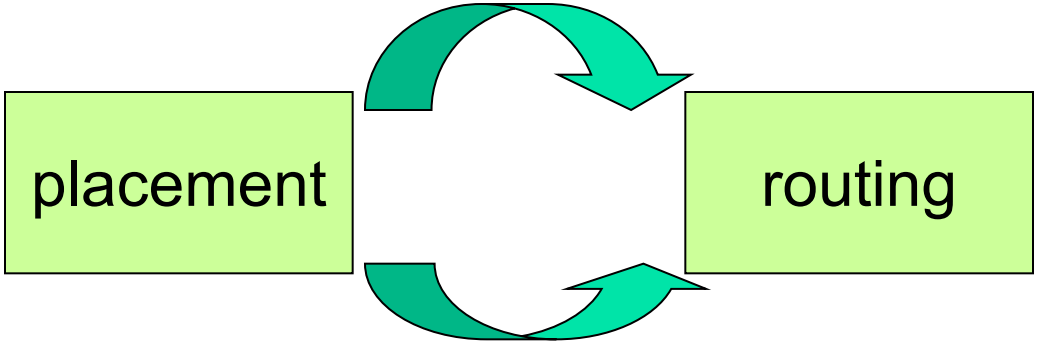
- Standard cells: power and clock connections run horizontally through the cell, and other I/O leaves the cell from the top or bottom sides.
  - The cells are placed in rows.
  - Sometimes *feedthrough* cells are added to ease wiring.
- Gate array: placement should be able to deal with fixed channel capacities





Relation with Routing

- Ideally, placement and routing (P&R) should be performed simultaneously as they depend on each other's results; it, however, often too complicated.
  - Pan and Chu, "IPR: An integrated placement and routing algorithm," DAC-07.
- Approximation: Placement estimates the wire length of a net using some wirelength *metric*.



placement與routing其實應該要同時並進，但是placement已經很複雜了，再加上routing會更難，因此實際在作的時候都是先做placement

幫placement結果"快速"打分數的model

Basic Wirelength Models

- **Half-perimeter wirelength (HPWL):** Half the perimeter of the bounding rectangle that encloses all the pins of the net to be connected. Most widely used approximation!
用最上下左右的點畫出一個矩形，看這個矩形的半周長(長+寬)-->簡單好用又快速
- **Squared Euclidean distance:** squares of all pairwise terminal distances in a net using a quadratic cost function

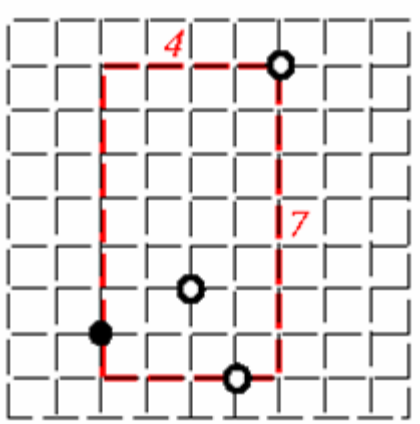
$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2]$$

相對來說計算時間較久，此方法較不實用

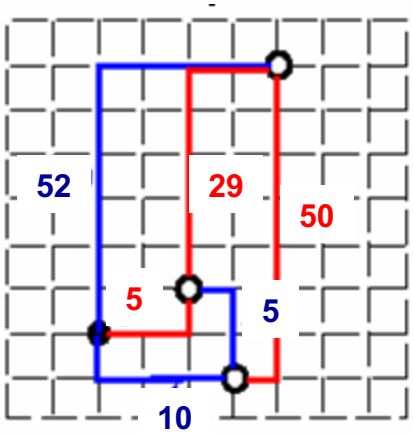
routing的時候才比較適用steiner tree(用steiner tree才能找出最佳解)，在placement的時候會有點太花時間
- **Steiner-tree approximation:** Computationally expensive.
spanning tree的準確度會比HPWL更好，但是還是有點太花時間了，所以大家還是用HPWL
- **Minimum spanning tree**: Good approximation to Steiner trees.
- **Complete graph:** Since #edges in a complete graph is $\left(\frac{n(n-1)}{2}\right)$ = $\frac{n}{2}$ × # of tree edges (n-1), *wirelength* ≈ $\frac{2}{n}\sum_{(i,j)\in net} dist(i, j)$.

Wirelength Estimation Examples

可以發現HPWL估算出來的結果與steiner很接近，而且實作起來簡單，因此被公認是最有效的做法

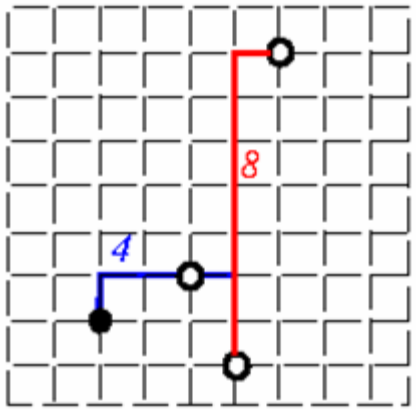


half perimeter
wirelength = 11

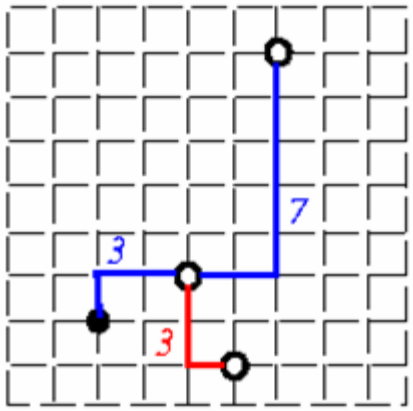


squared Euclidean
distance = 15.1 (γ_{ij} = 0.1)

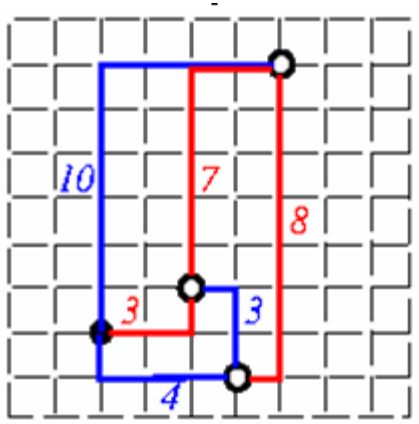
最精確的解答



min Steiner tree
wirelength = 12



min spanning tree
wirelength = 13

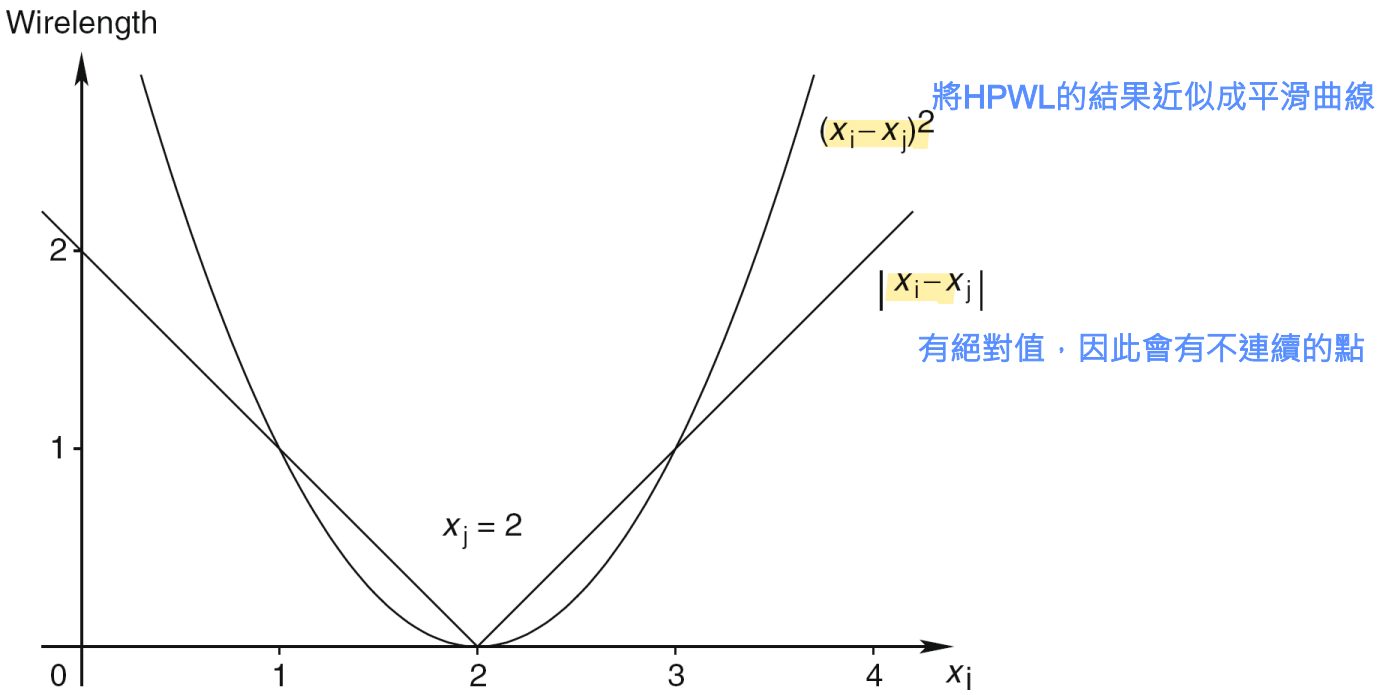


complete graph
wirelength * 2 / n= 17.5

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Quadratic Wirelength vs. Linear HPWL

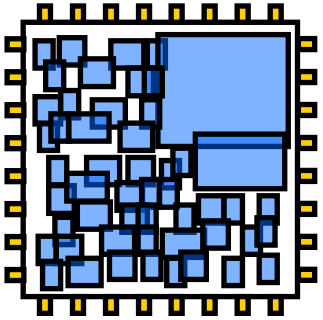
- HPWL is convex, but not differentiable
  HPWL的曲線會是不平滑的折線(因為有絕對值)，不可微-->因此會將其結果近似成另外一條平滑曲線
- Quadratic wirelength is a convex, smooth function → easy to minimize
- Both correlate “well” with each other



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Typical Placement Flow

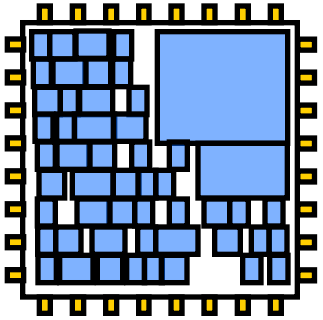
Global Placement (GP)



Computes the best position for each block to minimize the cost (e.g., wirelength

這時候排出來的結果允許有一點overlap

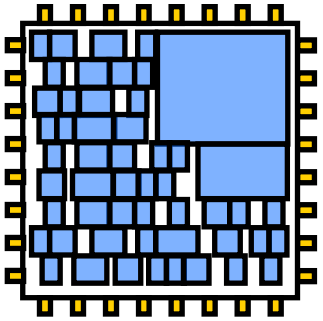
Legalization (LG)



Removes all overlaps among blocks

將DRC問題消除(消除global placement中的overlap)
此步驟產出的結果就已經是合法解了

Detailed Placement (DP)



Refines the solution

將結果進行優化

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

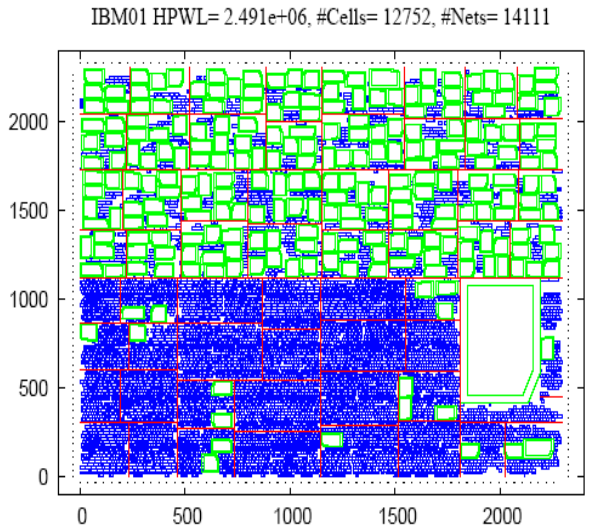
Global Placement Algorithms: Operation Perspective

- The placement problem is NP-complete.
- Popular placement algorithms:
  - **Constructive algorithms:** once the position of a cell is fixed, it is not modified anymore. cell一擺上去就不再動
    - Cluster growth, linear assignment, min cut, QP, etc.
  - **Iterative algorithms:** intermediate placements are modified in an attempt to improve the cost function.
    - Force-directed method, etc 將某一次步驟的最佳解作為下次iteration的初始狀態，不斷迭代-->ex: partition的FM
  - **Nondeterministic approaches:** simulated annealing, genetic algorithm, etc. 也算是iterative的演算法，只是他們包含隨機的因素在裡面
- Can combine multiple elements:
  - Constructive algorithms are used to obtain an initial placement.
  - The initial placement is followed by iterative improvement.
  - The results can further be improved by simulated annealing.

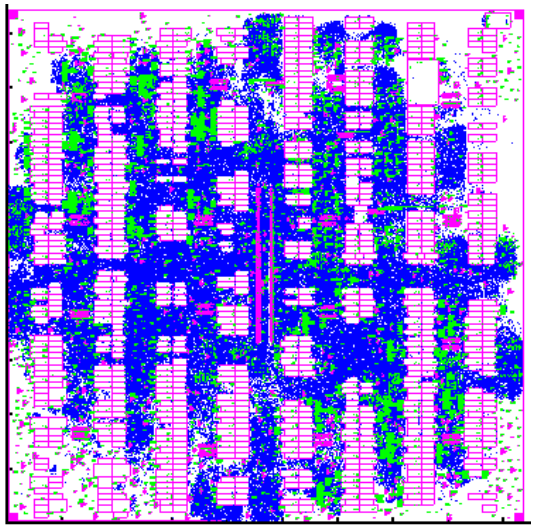
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Global Placement Algorithms: Algorithmic Perspective

- Partitioning-based method
  - Min-cut placement
- Nondeterministic methods
  - Simulated annealing (SA)
  - Genetic algorithm (GA)
- Analytical methods
  - Force-directed placement
  - Quadratic placement (QP)
  - Non-quadratic placement



Partitioning-based method



Analytical method

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

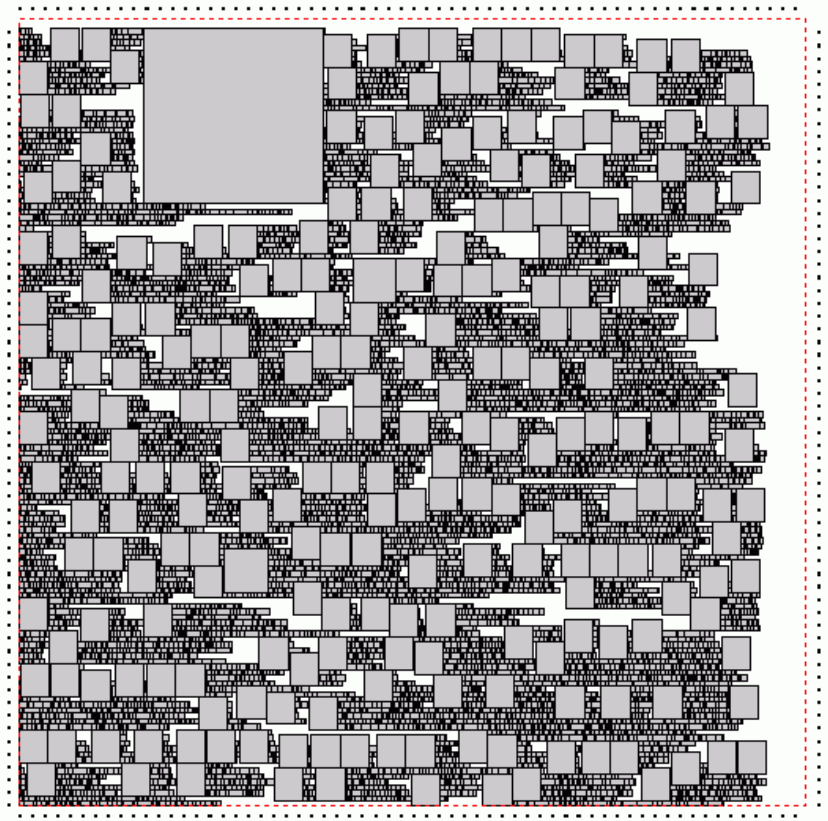
Example Academic Placers

- Partitioning/clustering based placers
  - NTUplace1 (ISPD-05): http://eda.ee.ntu.edu.tw/research.htm
  - Capo (DAC-00, ICCAD-04, ISPD-05, ISPD-06): http:vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement/Capo/
  - FungShui (ISPD-04, -05)
- Simulated annealing based placers
  - Dragon (ICCAD-00, ISPD-05, ISPD-06): http://er.cs.ucla.edu/Dragon/
- Analytical placers
  - GORDIAN (ICCAD-90)
  - Aplace (ICCAD-04, ISPD-05, ICCAD-05, ISPD-06)
  - Kraftwerk (DAC-98), Kraftwerk&Domino (ISPD-05, ICCAD-06)
  - mPL (ICCAD-03, ISPD-05, ISPD-06)
  - FastPlace (ISPD-04, -05, ASPDAC-06, ASPDAC-07)
  - NTUplace3 (ICCAD-06)
  - RQL (DAC-07)
- Hybrid placers: NTUplace2 (ISPD-06; partitioning + analytical), Dragon (ISPD-05; SA + partitioning)

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Example Placements

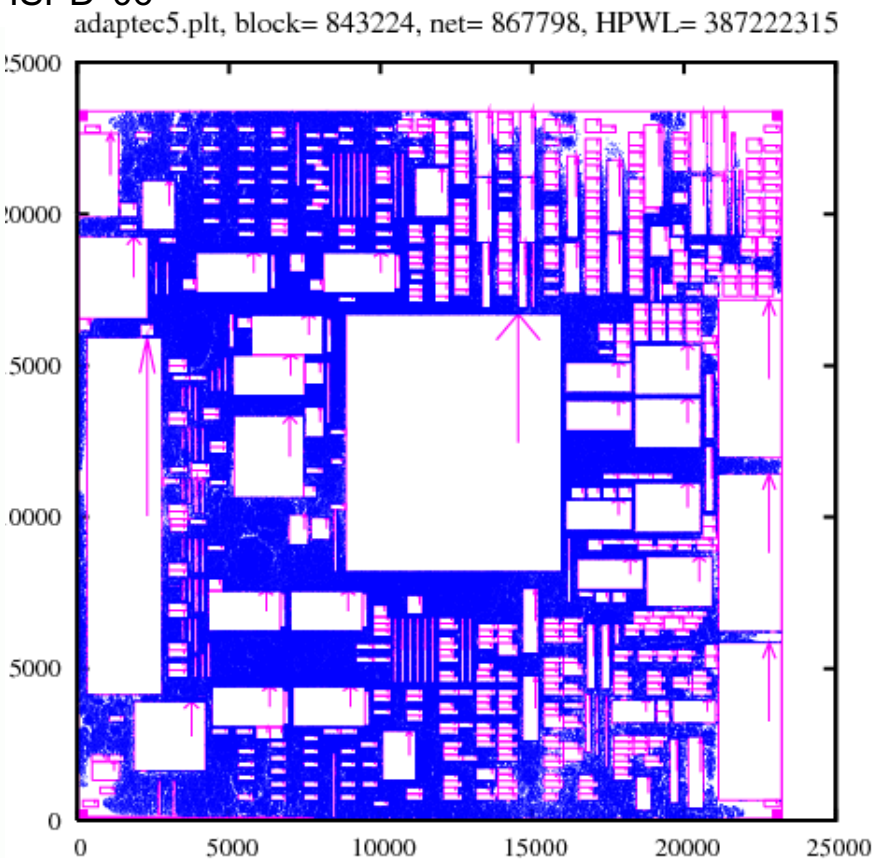
ISPD98 ibm01



12,752 cells, 247 macros

Amax/Amin = 8416

ISPD”06



842K movable cells

646 fixed macros

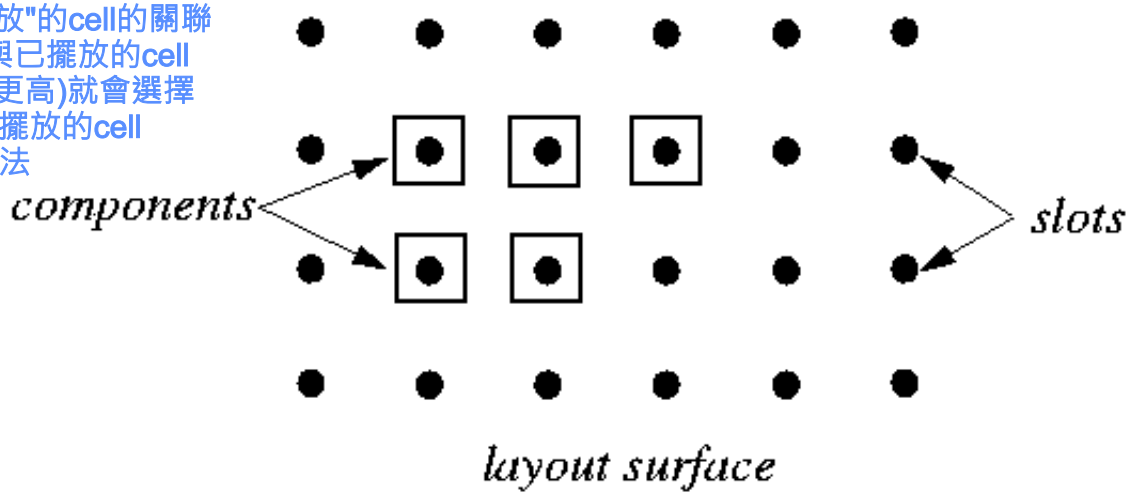
868K nets

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Bottom-Up Placement by Cluster Growth

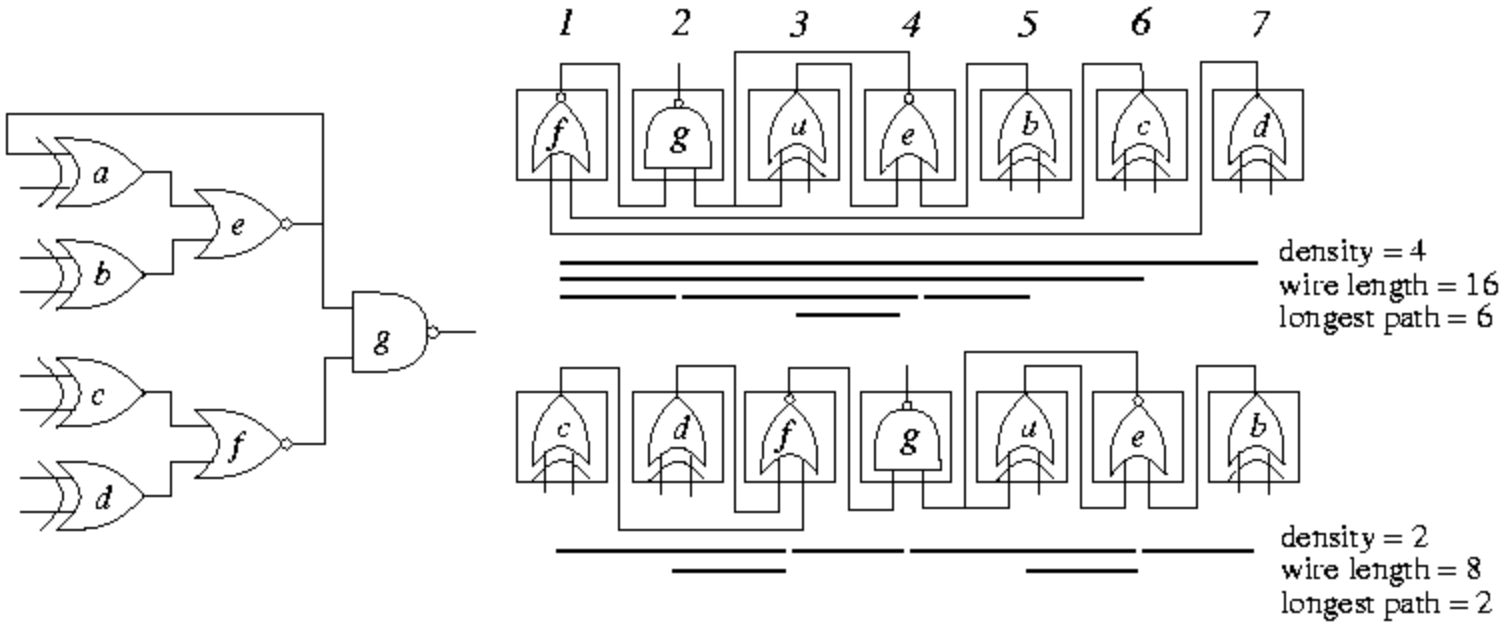
- Greedy method: Selects unplaced components and places them in available slots.
  - SELECT: Choose the unplaced component that is most strongly connected to all of the placed components (or most strongly connected to any single placed component).
  - PLACE: Place the selected component at a slot such that a certain “cost” of the partial placement is minimized.

SELECT: 從還沒擺放的cell中挑一個出來，會根據與"已擺放"的cell的關聯性來進行選擇，如果與已擺放的cell有更多的連線(關聯性更高)就會選擇這個cell做為下一個要擺放的cell (greedy) --> 不好的方法



Cluster Growth Example

- # of other terminals connected: c_a=3, c_b=1, c_c=1, c_d=1, c_e=4, c_f=3, and c_g=3 ⇒ *e* has the most connectivity.
- Place *e* in the center, slot 4. *a, b, g* are connected to *e*, and $\hat{c}_{ae} = 2, \hat{c}_{be} = \hat{c}_{eg} = 1$ ⇒ Place *a* next to *e* (say, slot 3). Continue until all cells are placed.
- Further improve the placement by swapping the gates.

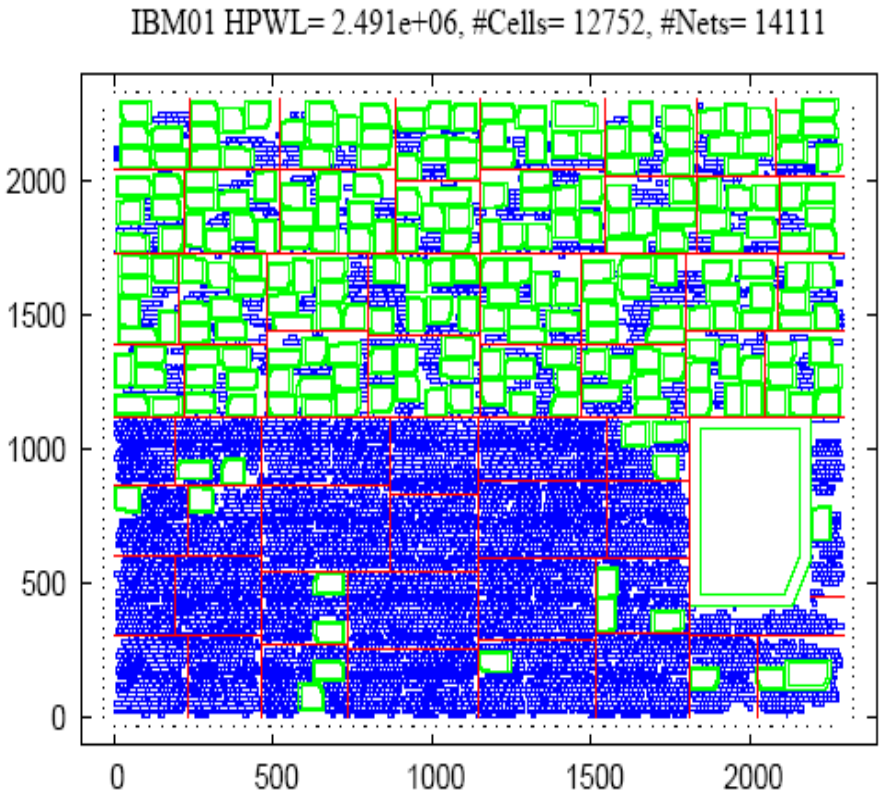


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Partitioning Based Placement

- Rely on a good partitioner (e.g., hMetis) and partition metric (e.g., exact net-weight modeling)
- Pros
  - Is very fast
  - Has very good scalability for handling large-scale designs
- Cons
  - Relatively harder to handle design with large white spaces

直接把Hmetis的結果拿來用-->好快



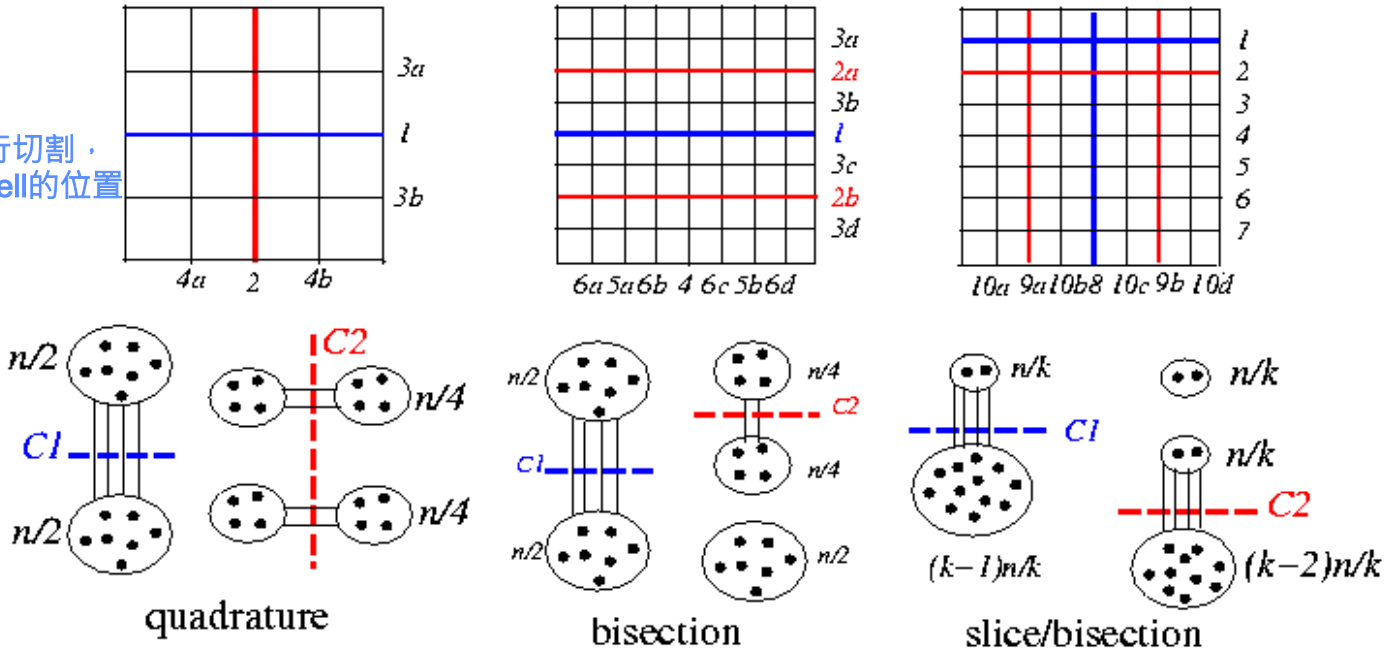
缺點: partition的時候無法知道每個cell彼此之間的間隙，因此擺放的時候也許無法那麼均勻

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Top-Down Min-Cut Placement

- Breuer, “A class of min-cut placement algorithms,” DAC-77.
- **Quadrature:** suitable for circuits with high density in the center. 一刀直的一刀橫的，交錯切
- **Bisection:** good for standard-cell placement. 水平切
- **Slice/Bisection:** good for cells with high interconnection on the periphery. 一次切一小段(不是切在中間)，不是均勻地切割

遞迴的方式去進行切割，
邊切邊決定每個cell的位置



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Algorithm for Min-Cut Placement

Algorithm: Min_Cut_Placement(*N*, *n*, *C*)

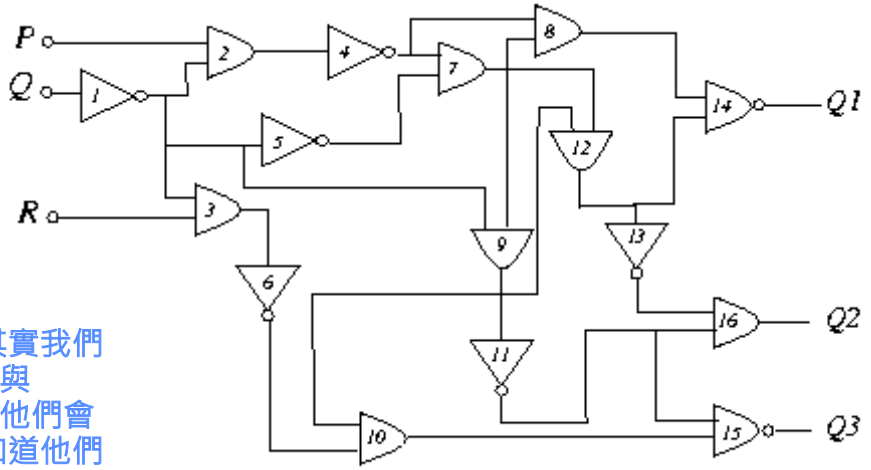
```
/* N: the layout surface */
/* n : # of cells to be placed */
/* n0: # of cells in a slot */
/* C: the connectivity matrix */

1 begin
2 if (n ≤ n0) then PlaceCells(N, n, C)
3 else
4     (N1, N2) ← CutSurface(N);
5     (n1, C1), (n2, C2) ← Partition(n, C);
6  Call Min_Cut_Placement(N1, n1, C1);
7  Call Min_Cut_Placement(N2, n2, C2);
8 end
```

切到大小足夠小的時候就可以直接擺進去(不用再切)

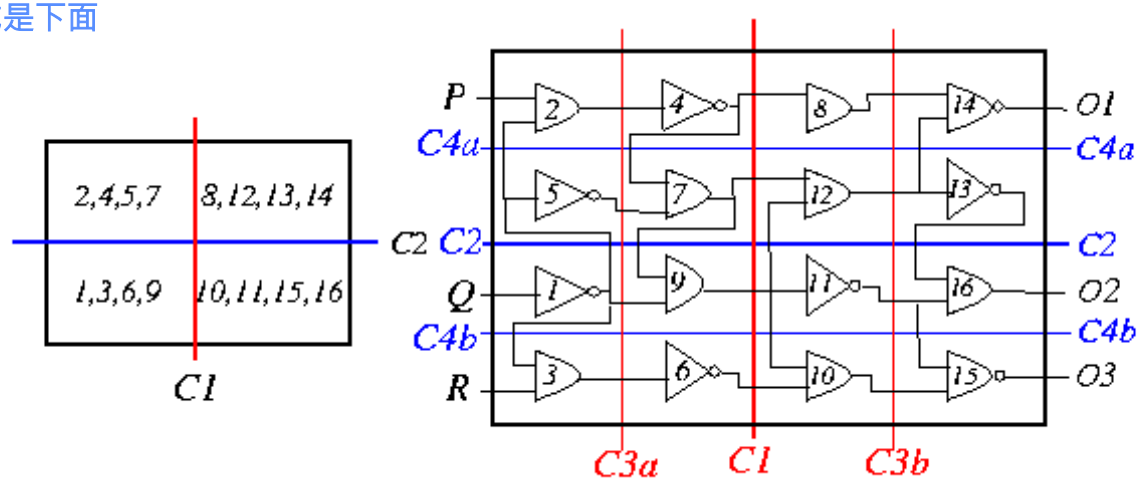
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Quadrature Placement Example

- Apply the F-M or K-L heuristic to partition + Quadrature Placement: Cost $C_1 = 4$, $C_{2L} = C_{2R} = 2$, etc.



切割出來的東西該擺在哪裡其實我們是不知道的，以圖中{1,3,6,9}與{2,4,5,7}來說，雖然我們知道他們會被分成兩半，但是我們並不知道他們兩個誰該擺在上面或是下面



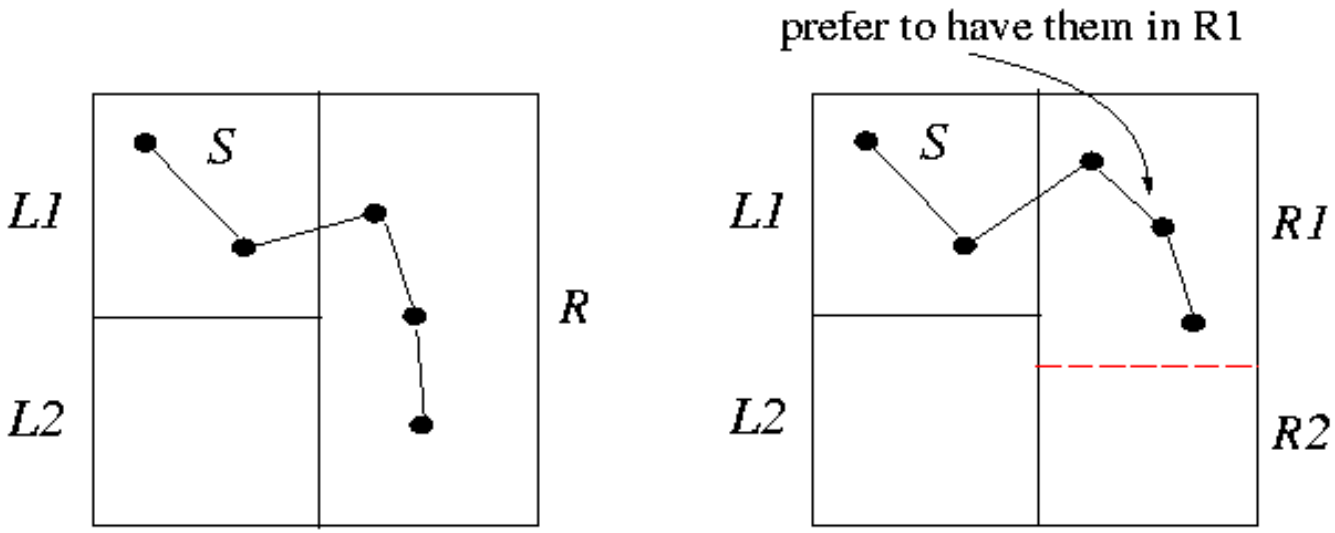
What if we swap {1, 3, 6, 9} and {2, 4, 5, 7}??

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

為了解決剛剛不知道誰該擺哪的問題

Min-Cut Placement with Terminal Propagation

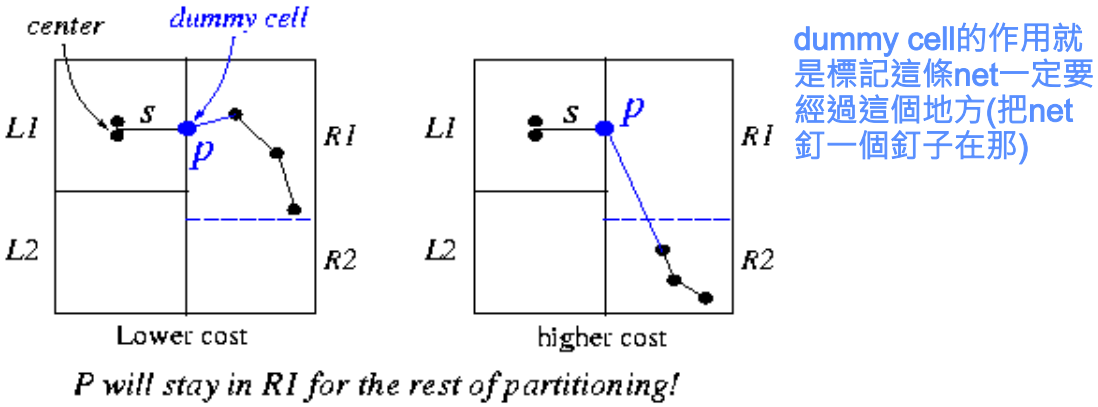
- Dunlop & Kernighan, “A procedure for placement of standard-cell VLSI circuits,” *IEEE TCAD*, Jan. 1985.
- Drawback of the original min-cut placement: Does not consider the positions of terminal pins that enter a region.
  - What happens if we swap {1, 3, 6, 9} and {2, 4, 5, 7} in the previous example?



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

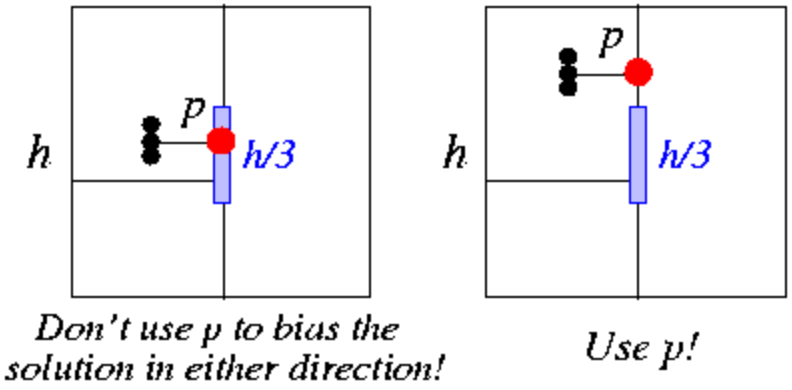
Terminal Propagation

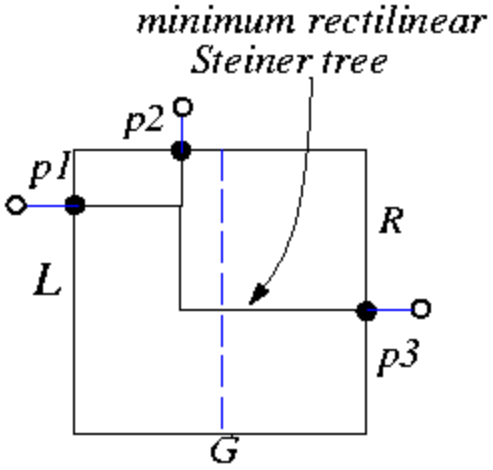
- We should use the fact that *s* is in L_1!



dummy cell的作用就是標記這條net一定要經過這個地方(把net釘一個釘子在那)

- When not to use *p* to bias partitioning? Net *s* has cells in many groups?

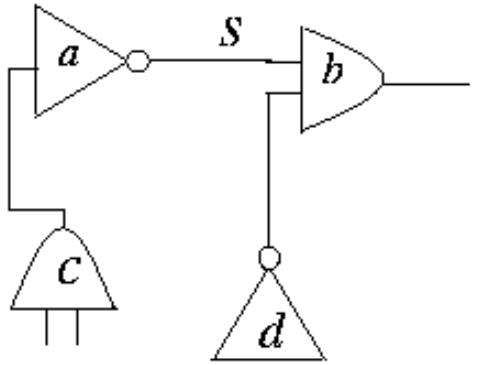


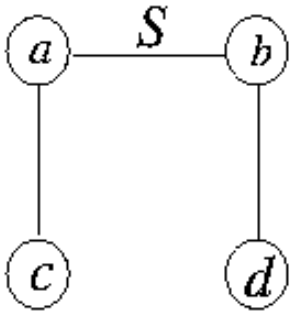


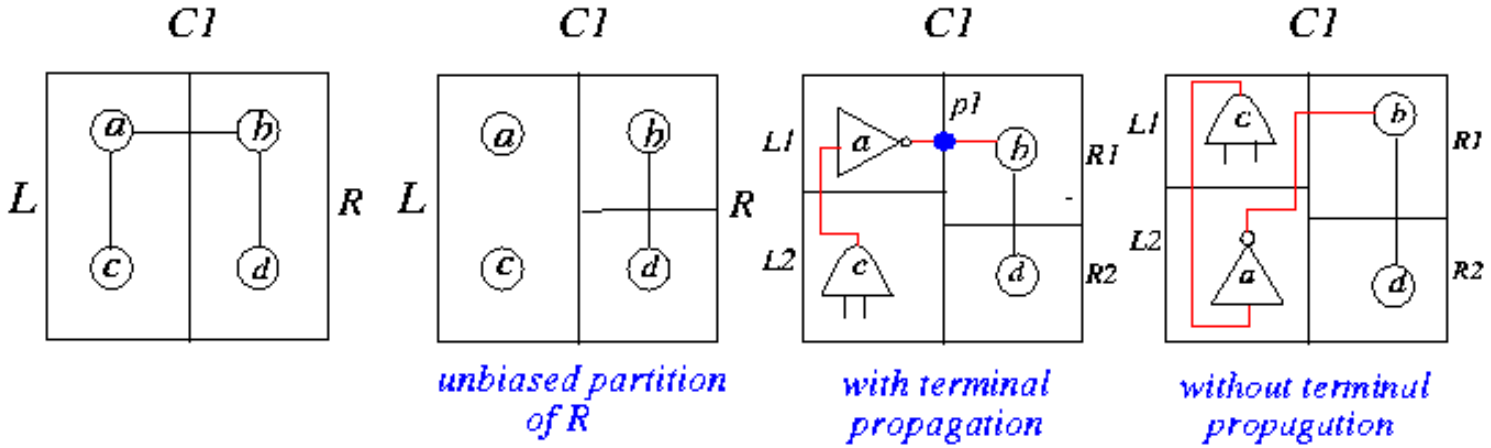
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Terminal Propagation Example

. Partitioning must be done breadth-first, not depth-first.





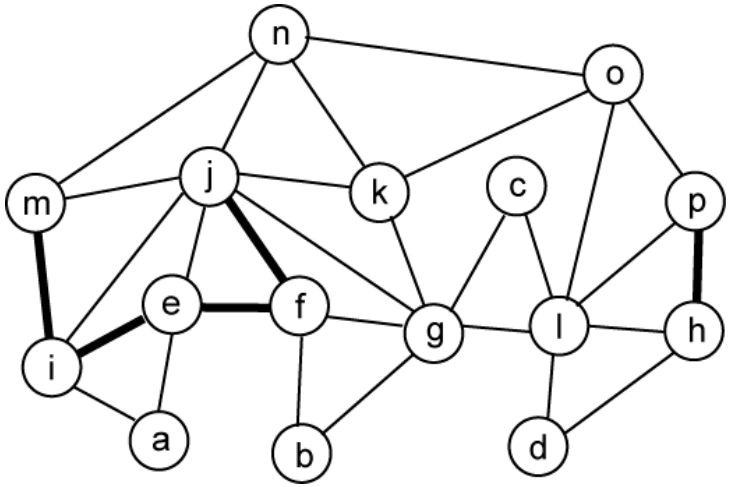


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Mincut Placement : Quadrature Mincut

- Perform quadrature mincut onto 4 × 4 grid
  - Start with vertical cut first

$$
\begin{aligned}
n_1 &= \{e, f\} \\
n_2 &= \{a, e, i\} \\
n_3 &= \{b, f, g\} \\
n_4 &= \{c, g, l\} \\
n_5 &= \{d, l, h\} \\
n_6 &= \{e, i, j\} \\
n_7 &= \{f, j\} \\
n_8 &= \{g, j, k\} \\
n_9 &= \{l, o, p\} \\
n_{10} &= \{h, p\} \\
n_{11} &= \{i, m\} \\
n_{12} &= \{j, m, n\} \\
n_{13} &= \{k, n, o\}
\end{aligned}
$$

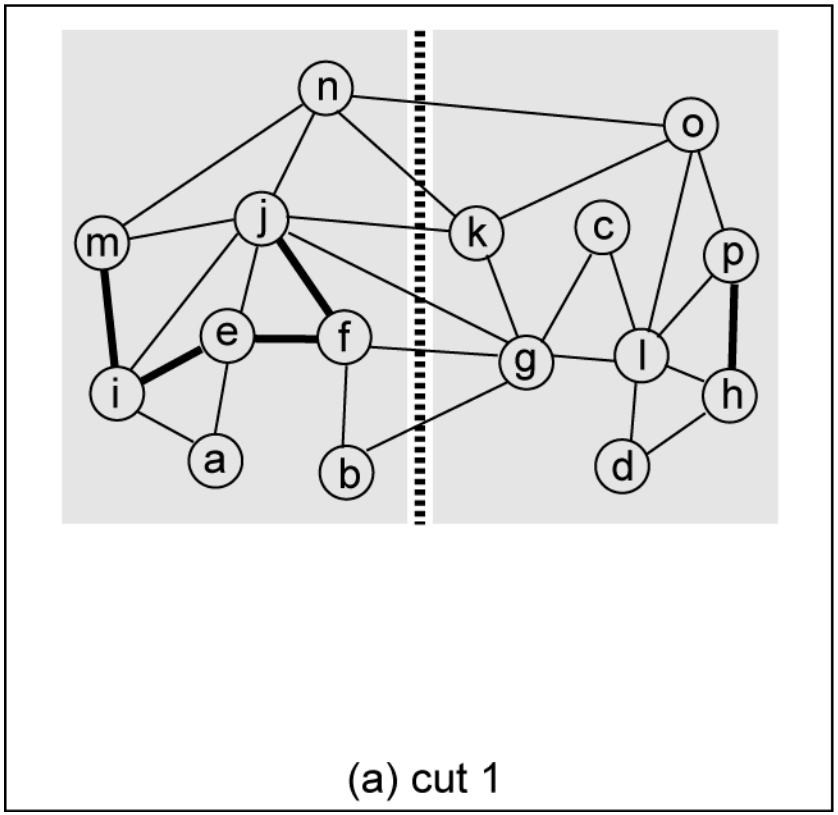


Clique-based graph model
thin edges = weight 0.5, thick edges = weight 1

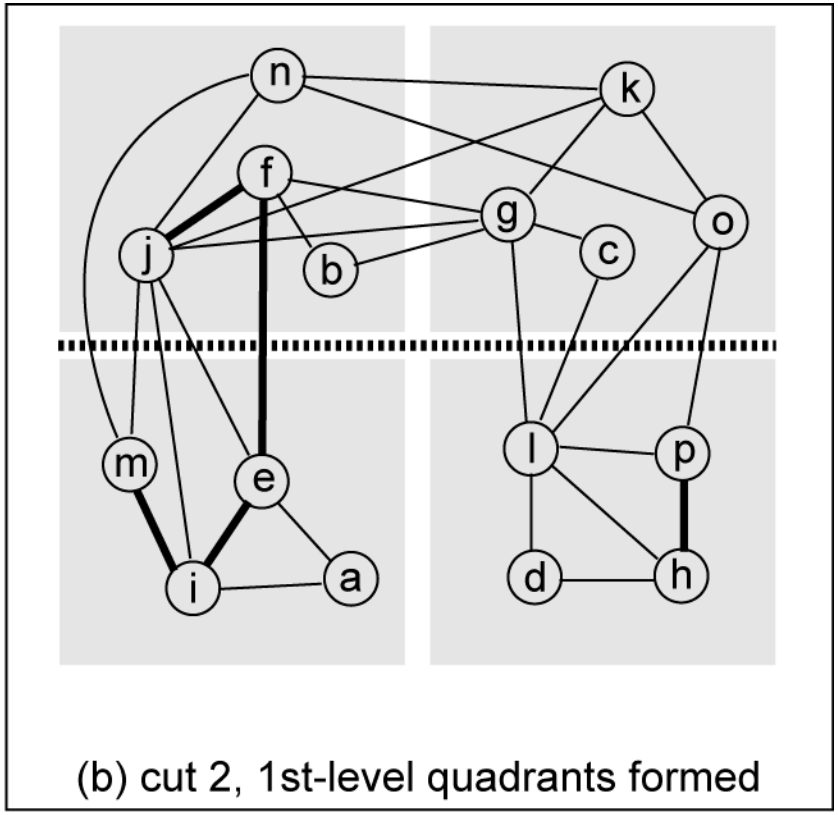
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cut 1 and 2

- First cut has min-cutsize of 3 (not unique)
  - Both cuts 1 and 2 divide the entire chip



(a) cut 1

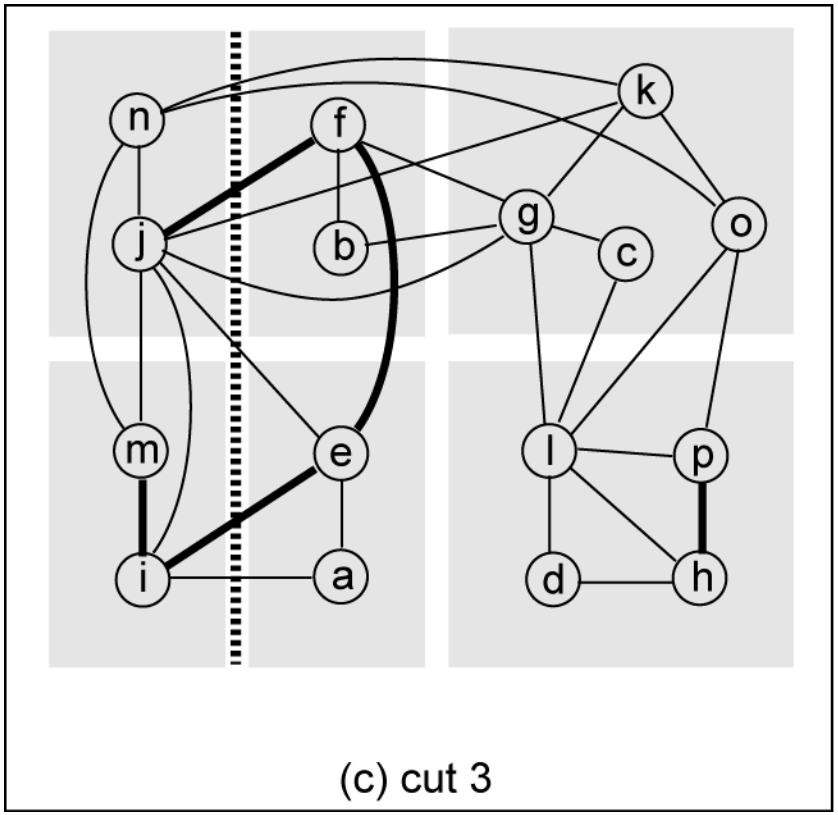


(b) cut 2, 1st-level quadrants formed

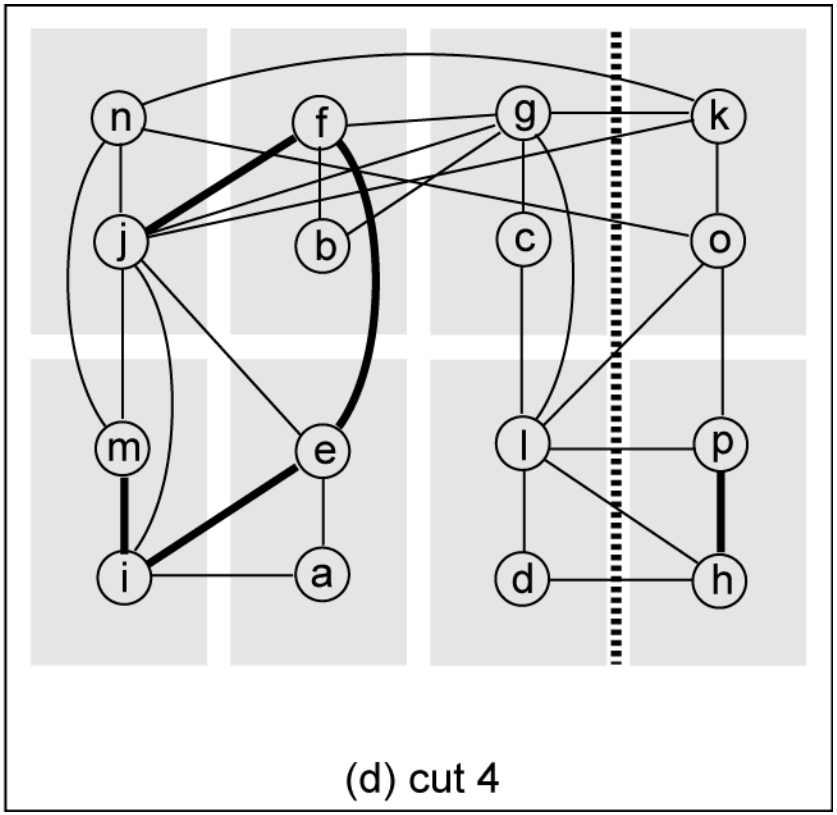
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cut 3 and 4

- Each cut minimizes cutsize
  - Helps reduce overall wirelength



(c) cut 3

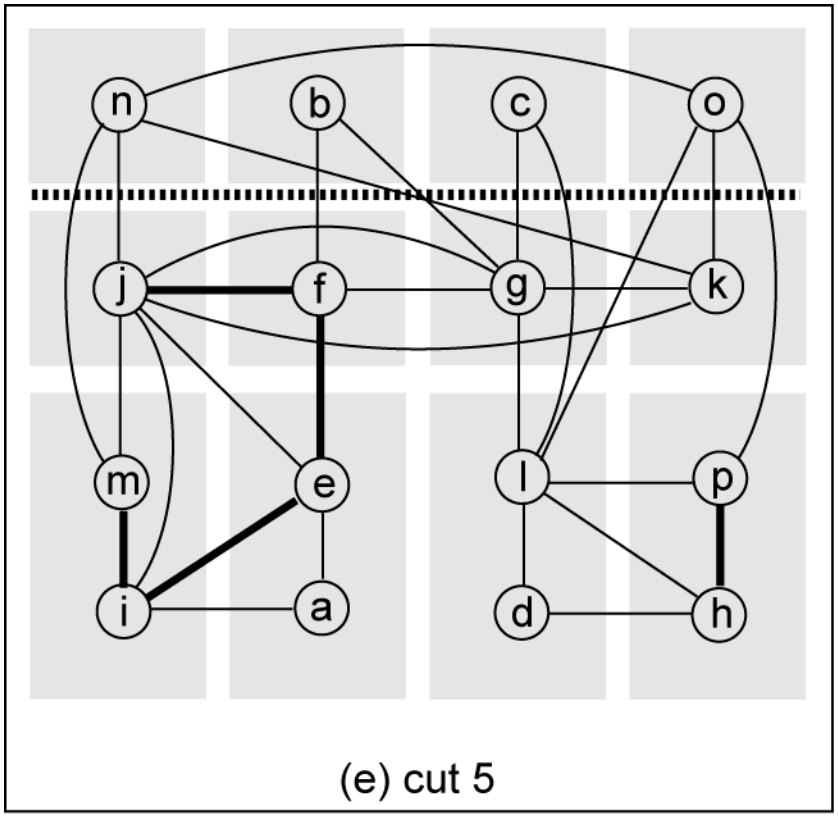


(d) cut 4

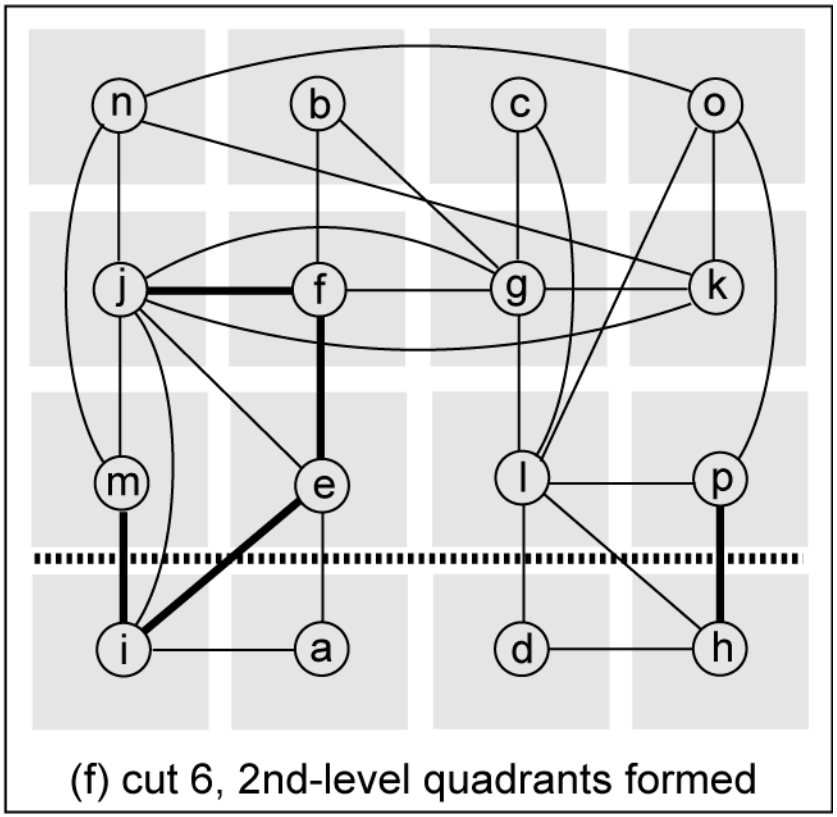
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cut 5 and 6

- 16 partitions generated by 6 cuts
  - HPBB wirelength = 27



(e) cut 5

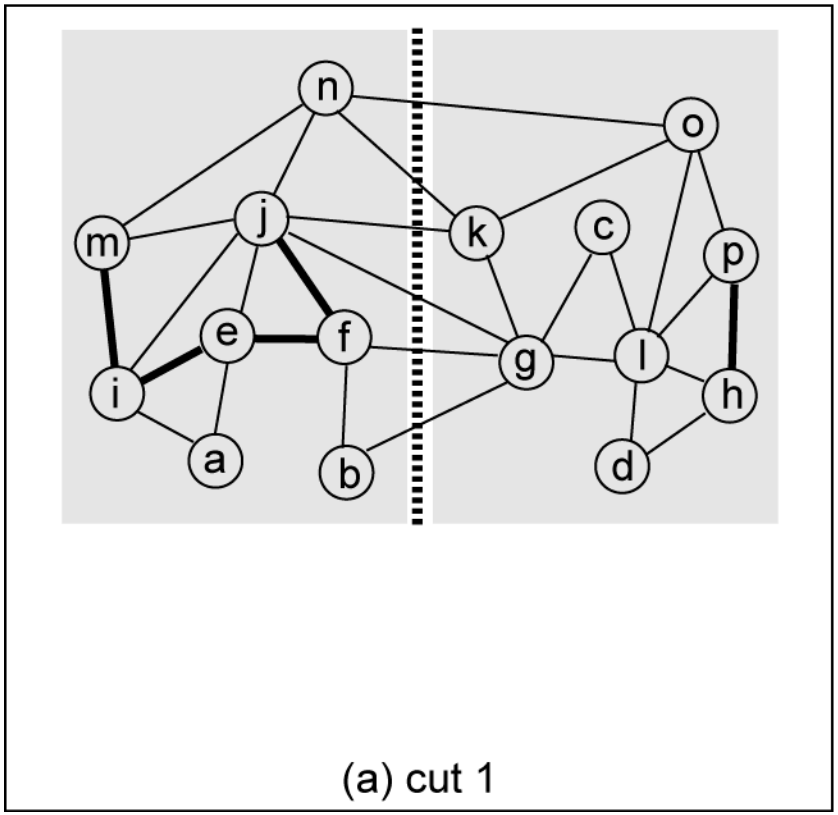


(f) cut 6, 2nd-level quadrants formed

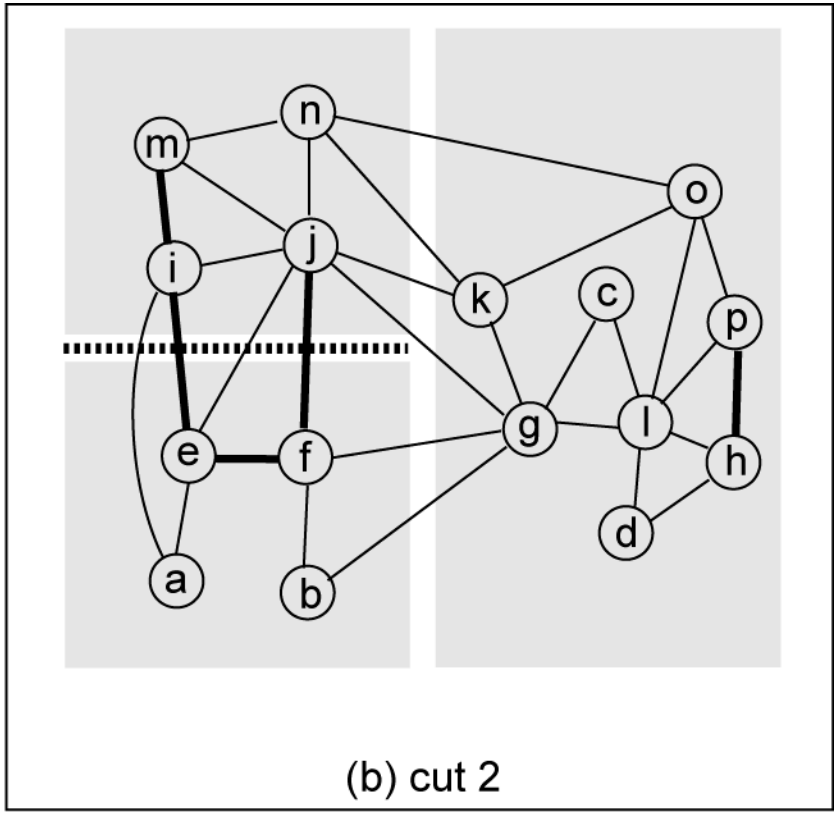
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Mincut Placement: Recursive Bisection

. Start with vertical cut



(a) cut 1

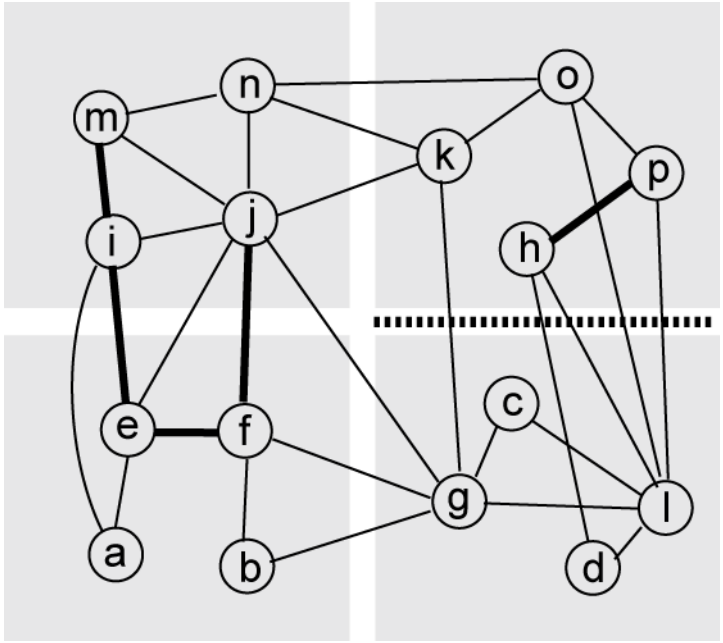


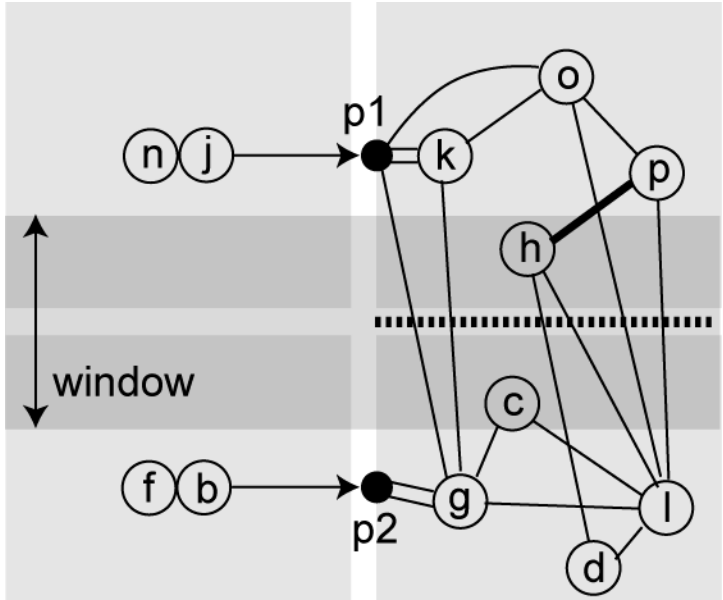
(b) cut 2

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cut 3: Terminal Propagation

- Two terminals are propagated: “pulling” nodes
  - Node *k* and *o* connect to *n* and *j*: p_1 propagated (outside window)
  - Node *g* connect to *j, f* and *b*: p_2 propagated (outside window)
  - Terminal p_1 pulls *k*/*o*/*g* to top partition, and p_2 pulls *g* to bottom



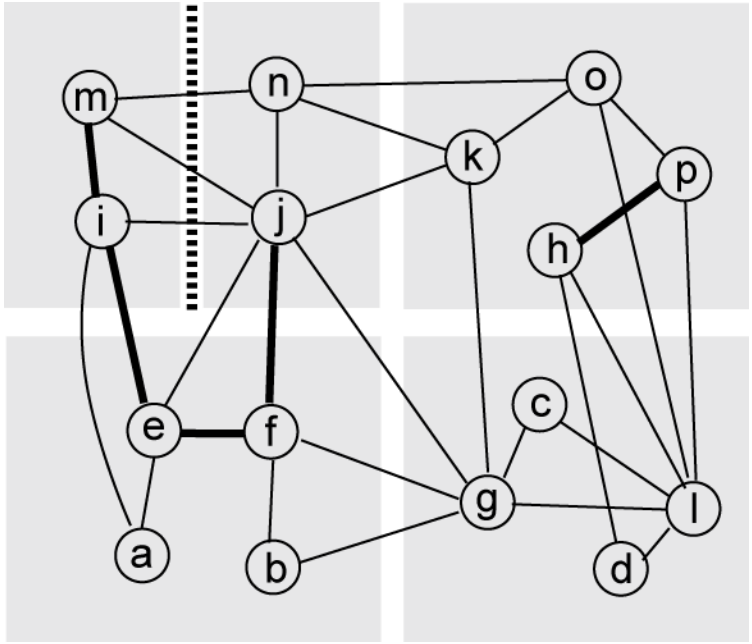


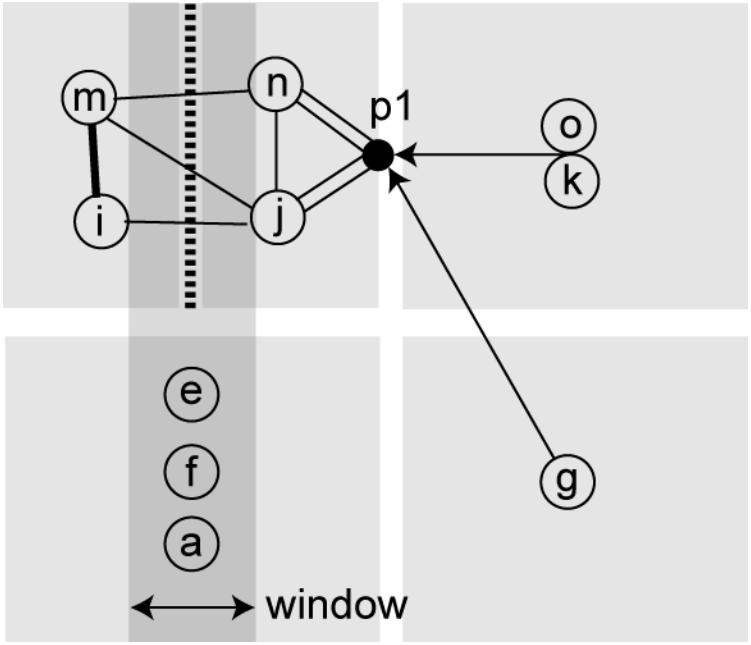
每次要進行切割之前，都會去看dummy node的位置
猜想: 把dummy node當成一個node(兩個set中都會有一些dummy node)，也是放在某個set中且lock住，進行partition的時候也要將其考慮進去來算其他cell的gain

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cut 4: Terminal Propagation

- One terminal propagated
  - Node *n* and *j* connect to *o*/*k*/*g*: p_1 propagated
  - Node *i* and *j* connect to *e*/*f*/*a*: no propagation (inside window)
  - Terminal p_1 pulls *n* and *j* to right partition

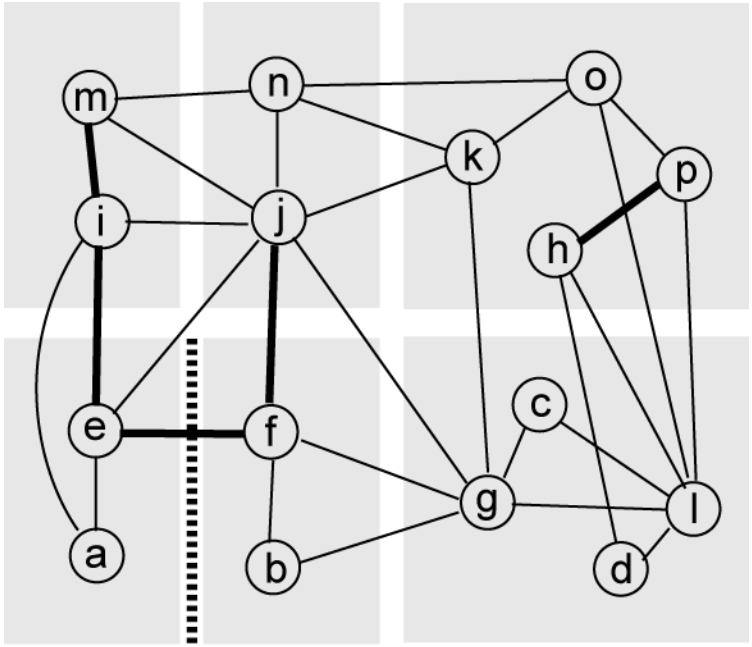


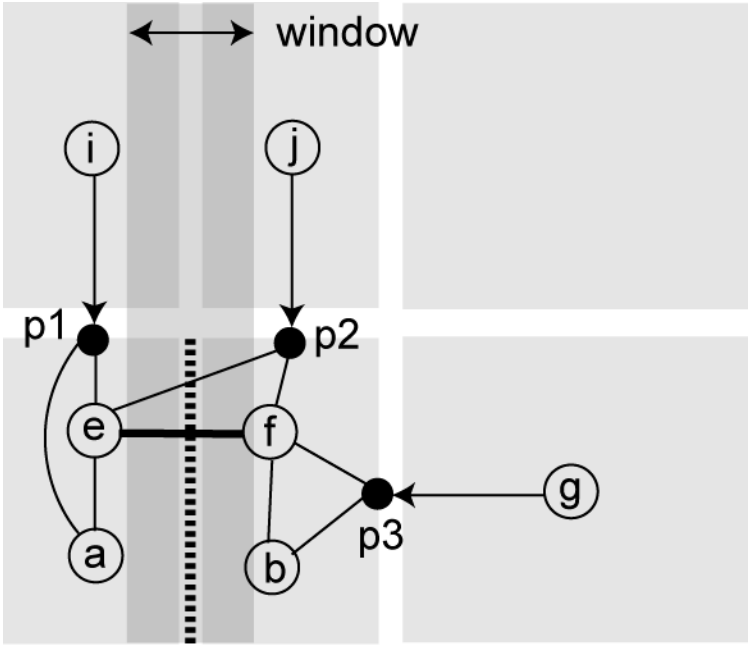


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cut 5: Terminal Propagation

- Three terminals propagated
  - Node *i* propagated to p_1, *j* to p_2, and *g* to p_3
  - Terminal p_1 pulls *e* and *a* to left partition
  - Terminal p_2 and p_3 pull *f*/*b*/*e* to right partition

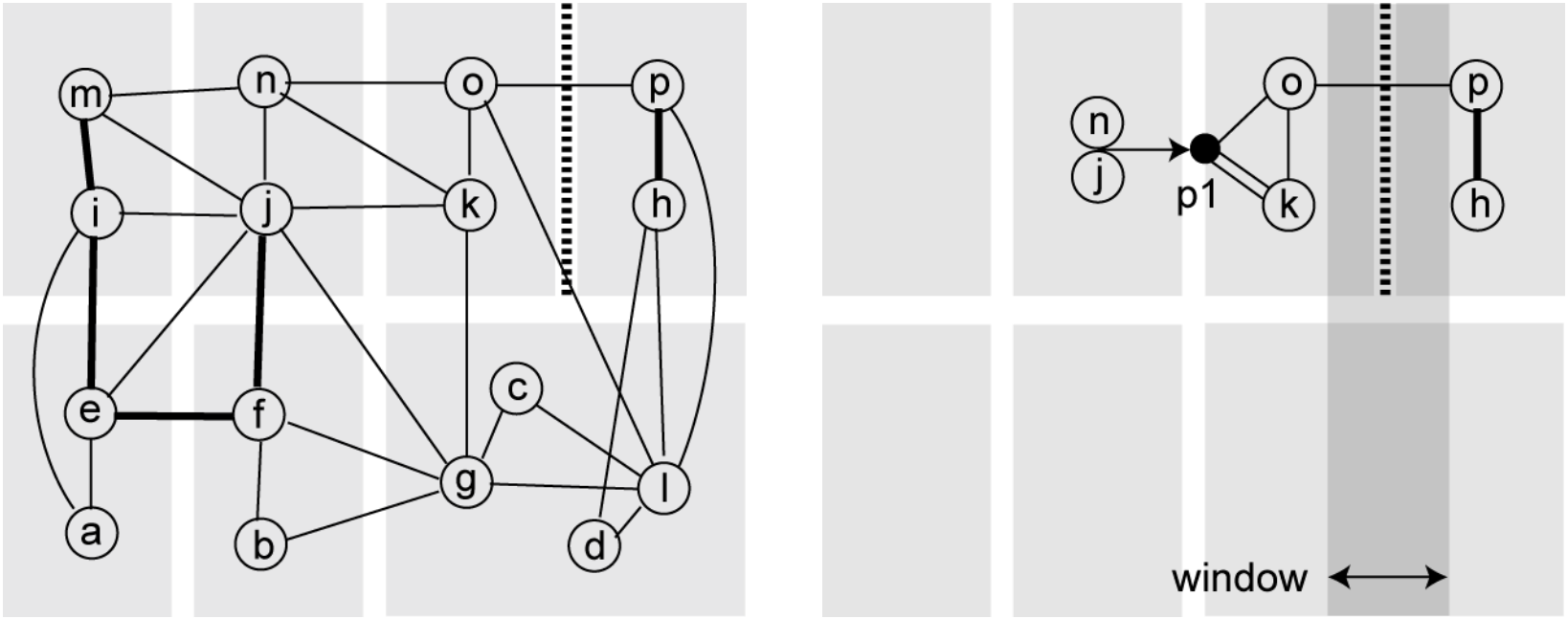




Courtesy of Prof. Y.-W. Chang & H.-M. Chen

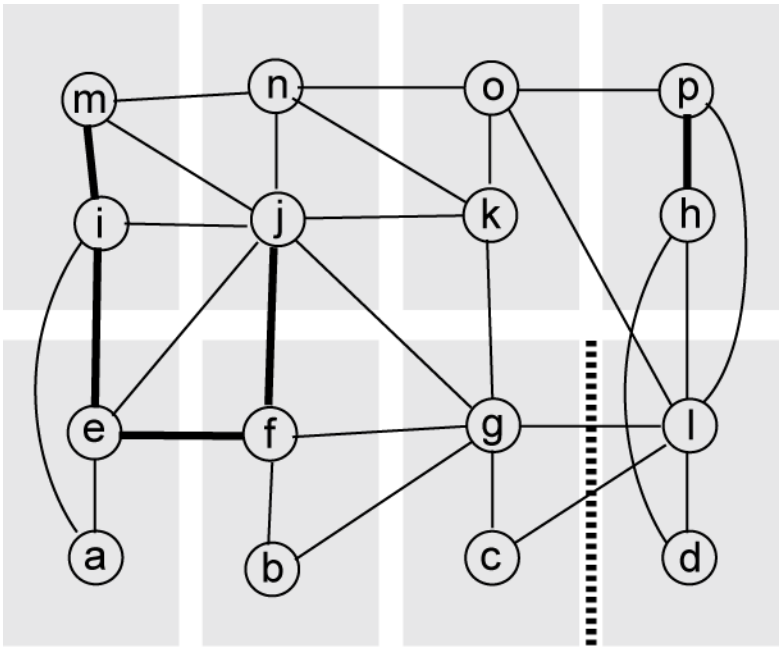
Cut 6: Terminal Propagation

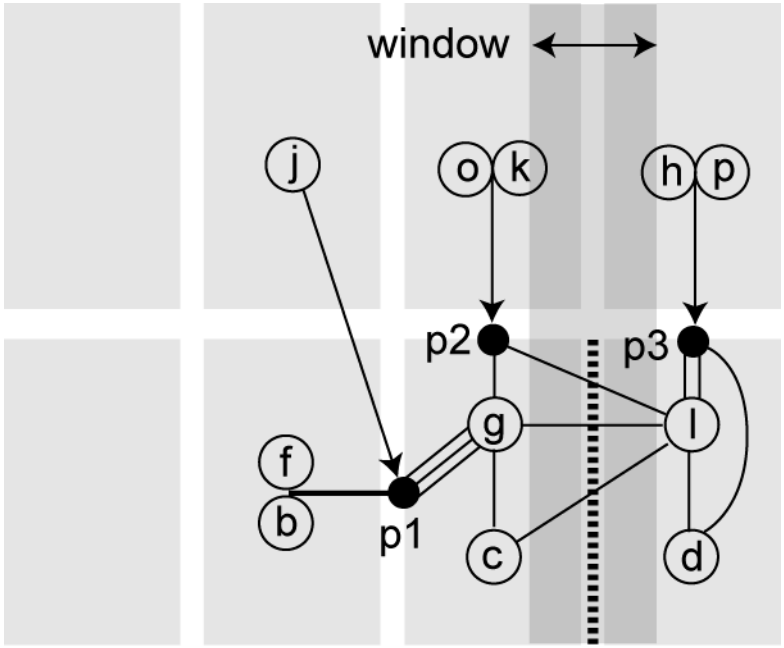
- One terminal propagated
  - Node n and j are propagated to p_1
  - Terminal p_1 pulls o and k to left partition



Cut 7: Terminal Propagation

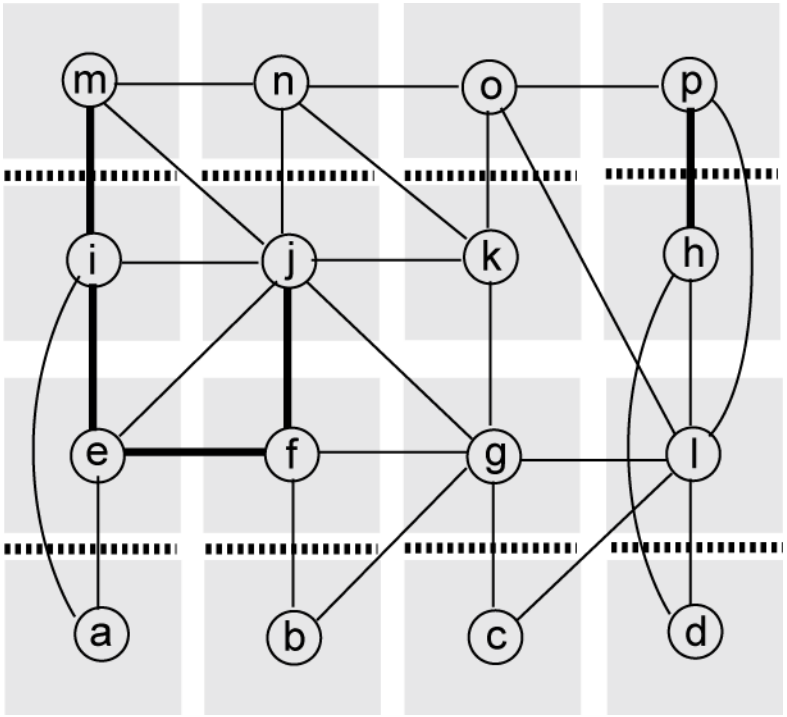
- Three terminals propagated
  - Node *j*/*f*/*b* propagated to p_1, *o*/*k* to p_2, and *h*/*p* to p_3
  - Terminal p_1 and p_2 pull *g* and *l* to left partition
  - Terminal p_3 pull *l* and *d* to right partition





Cut 8 to 15

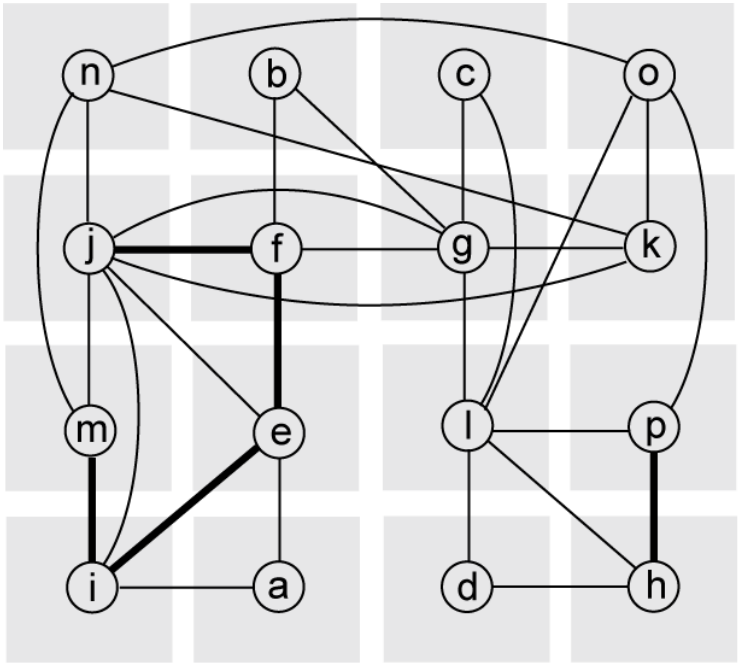
- 16 partitions generated by 15 cuts
  - HPBB wirelength = 23



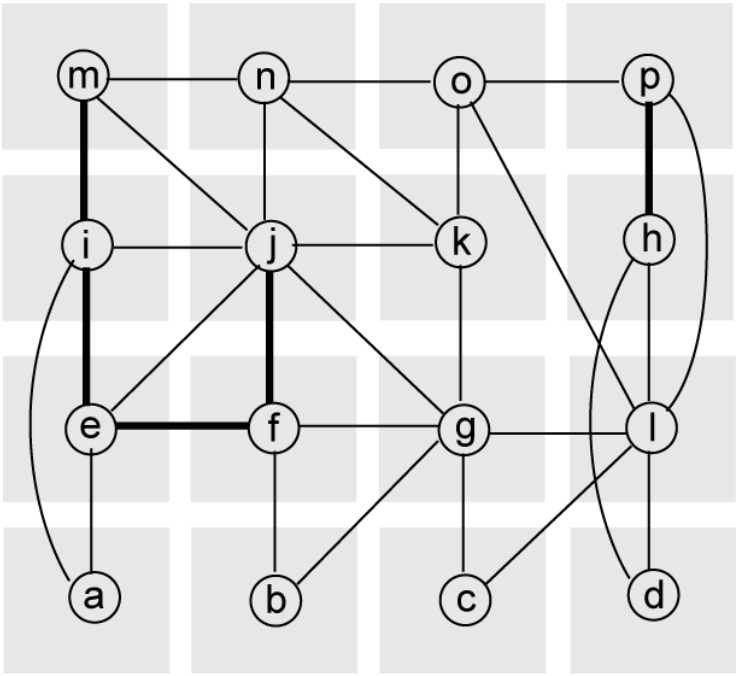
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Comparison

- Quadrature vs. recursive bisection + terminal propagation
  - Number of cuts: 6 vs. 15
  - Wirelength: 27 vs. 23



quadrature



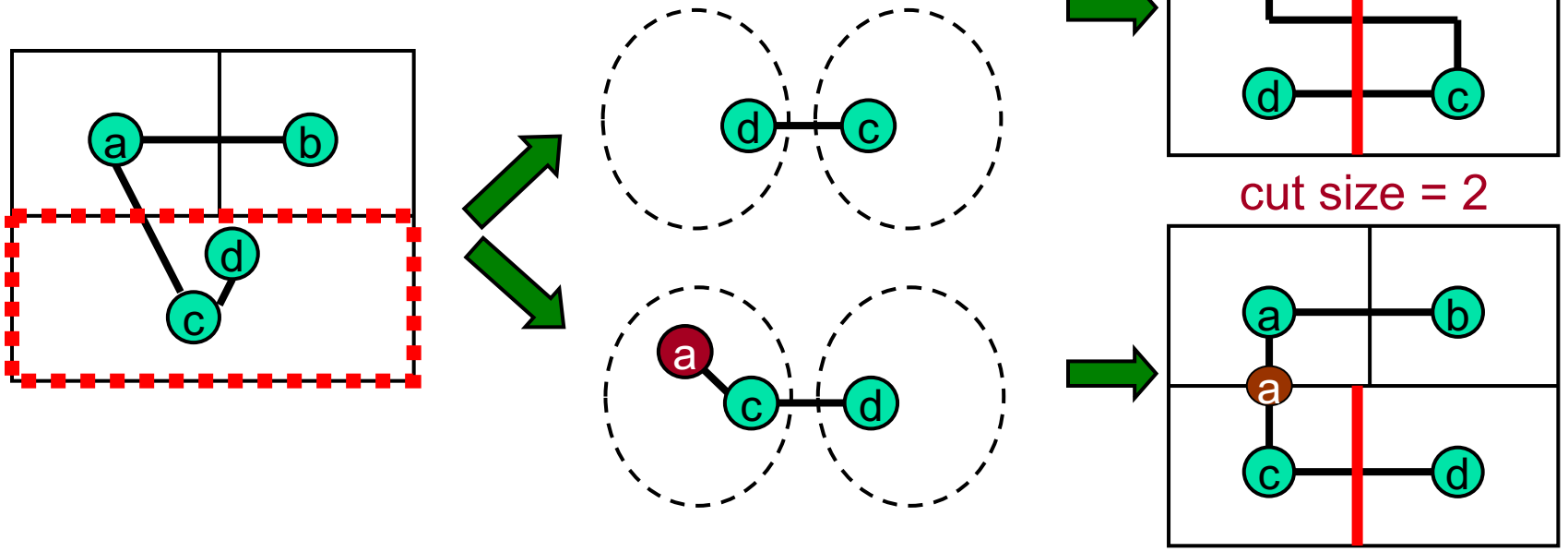
bisection

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Exact Net-Weight Modeling

- Chen, Chang, Lin, "IMF: Interconnect-driven multilevel floorplanning for large-scale building-module designs," ICCAD-05 (TCAD-08).
- After partitioning nodes *a* and *b*, how do we assign the regions for *c* and *d* to minimize the wire length?

A possible solution **without** using terminal propagation

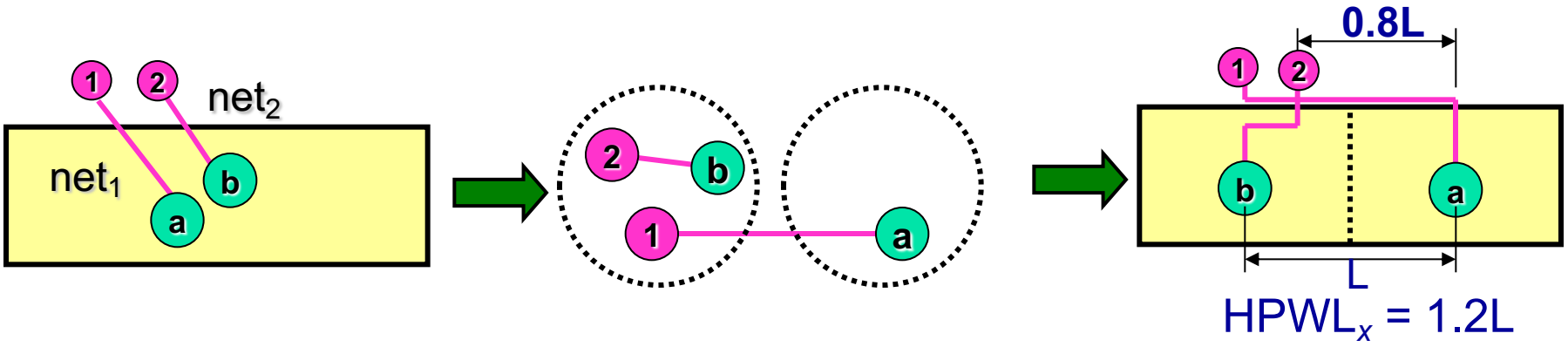


With terminal propagation
(Node a is a fixed node)

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Traditional Terminal Propagation (TP)

- Traditional TP considers only the cut size, which is *not* equivalent to finding the minimum HPWL exactly.
- Traditional TP might lead to a sub-optimal solution.
  - Nodes 1 and 2 are *fixed*, partitioning *a* and *b* to different regions gives the same cut size but incurs different HPWL's.

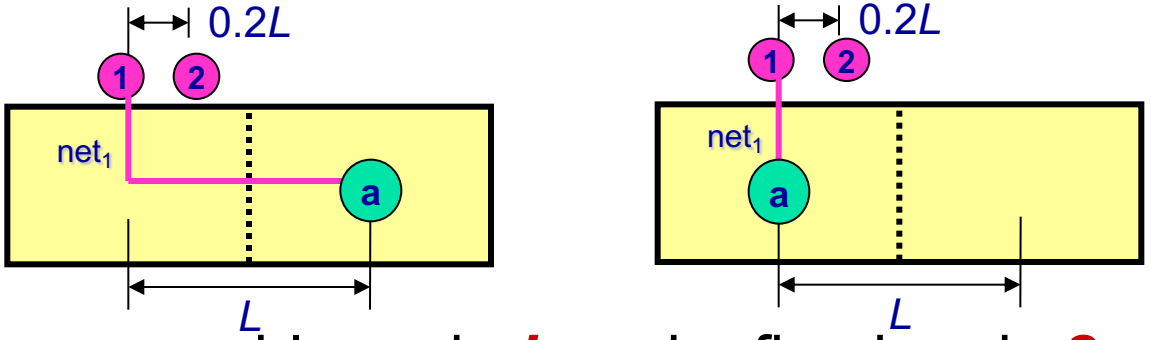


- Problem: edge weight is a **constant** value!
  - Shall map the min-cut cost to the HPWL change.
  - Shall assign the edge weight as the value of the HPWL contribution if the edge is cut.

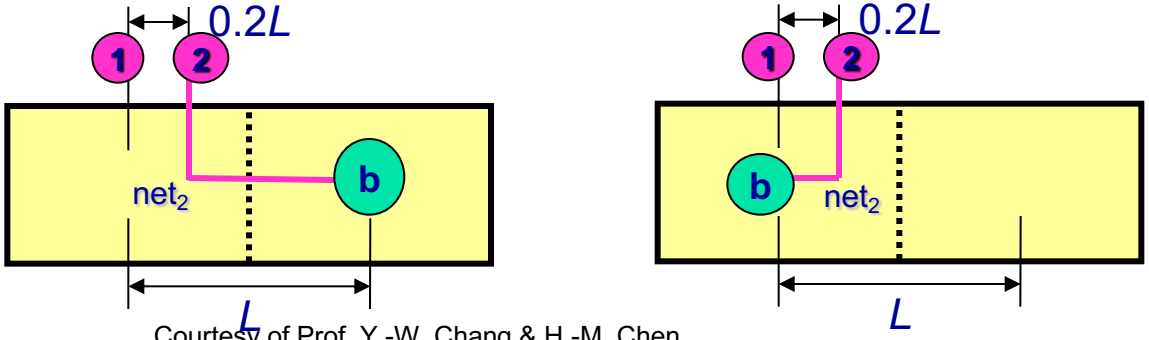
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

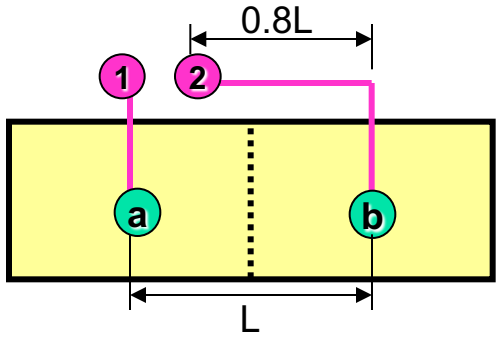
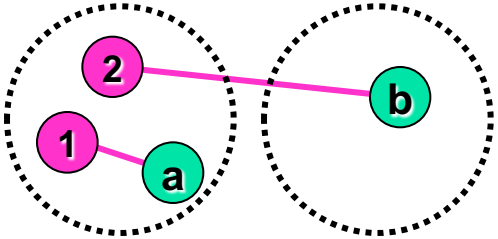
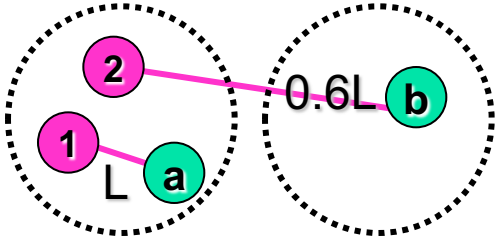
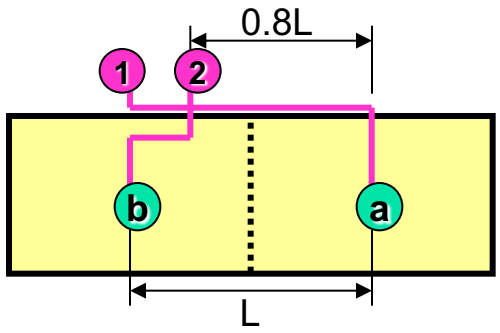
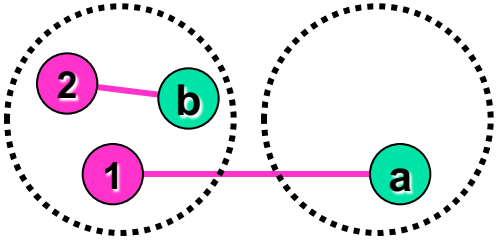
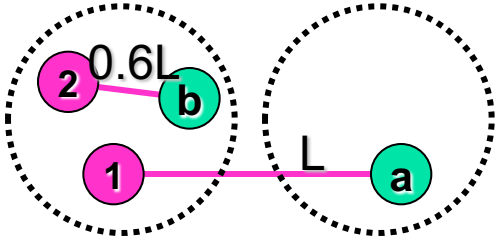
Net Weight Assignment

- Set edge weight to the value of HPWL if the edge is cut.
- net_1 connects a movable node ***a*** and a fixed node **1**.
  Weight(net_1) = Length(net_1 is cut) – Length(net_1 is not cut)
  $= L - 0L = L$



- net_2 connects a movable node ***b*** and a fixed node **2**.
  Weight(net_2) = Length(net_2 is cut) – Length(net_2 is not cut)
  $= 0.8L - 0.2L = 0.6L$



Physical Partitions	Traditional Terminal Propagation	Exact Net-Weight Modeling
 $HPWL_x = 0.8L$	 Cut size = 1	 Cut weight = 0.6L
 $HPWL_x = 1.2L$	 Cut size = 1	 Cut weight = 1.0L

Cut weight is proportional to the HPWL!

HPWL = Cut weight + 0.2L

(0.2L is the lower bound for the net's HPWL)

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Exact Net-Weight Modeling

- Chen, Chang, Lin, "IMF: Interconnect-driven multilevel floorplanning for large-scale building-module designs," ICCAD-05 (TCAD-08).
- Theorem: $HPWL_i = w_{1,i} + n_{cut,i}$
  - $n_{cut,i}$: cut weight for net i
  - $w_{1,i}$: the HPWL lower bound for net i (movable modules are all at the side that leads to the smallest HPWL)
- Then, we have

Constant!

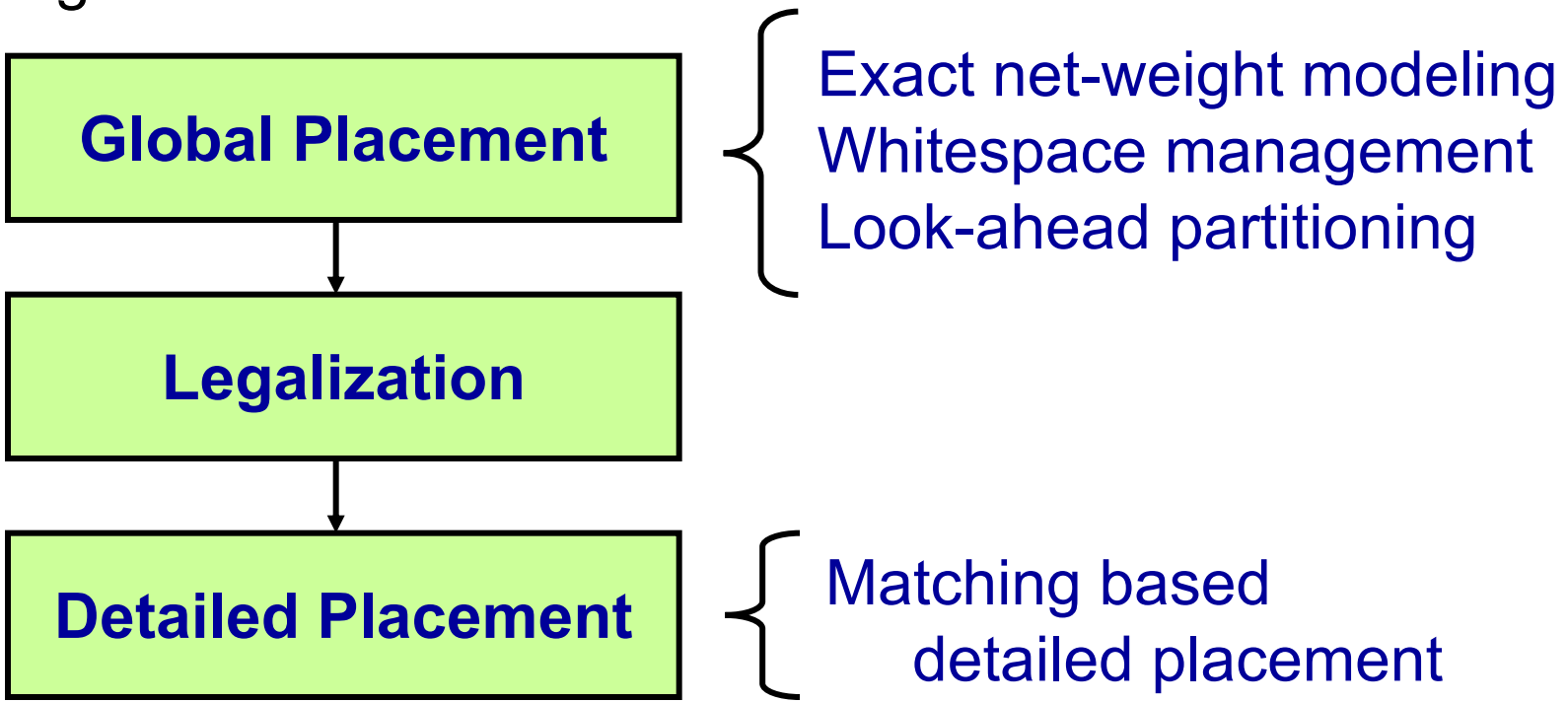
$$\min \sum HPWL_i \Rightarrow \min \sum (w_{1,i} + n_{cut,i}) \Rightarrow \sum w_{1,i} + \min \sum n_{cut,i}$$

Finding the minimum HPWL is equivalent to finding the min-cut!!

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

NTUplace1 Overview

- Chen, Hsu, Jiang, Chang, “A ratio partitioning based placement algorithm for large-scale mixed-size design,” ISPD-05
- Is based on the min-cut partitioning technique
- Algorithm overview



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Whitespace Management (1/2)

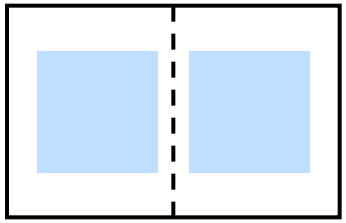
- Traditional min-cut placers uniformly distribute whitespace and tend to produce excessive wirelength when the whitespace is large
- Adya, Markov, Villarrubia use dummy cells to control the whitespace allocation [ICCAD-03]
  - Add dummy cells to increase the utilization ratio
  - Whitespace is distributed according to the dummy cell locations
  - However, their method tends to increase the number of cells, leading to longer running time and larger memory usage

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Whitespace Management (2/2)

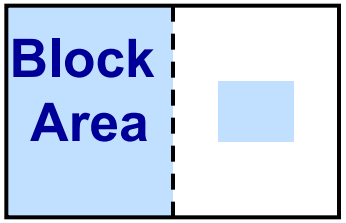
- NTUplace1 directly controls the balance criteria during partitioning using the available free space
- Relaxing the balance criteria leads to smaller cutsize and thus smaller wirelength
- The balance criterion satisfies that the utilization of each partition is less than or equal to 1
- The criterion is then fed into the partitioner to allocate whitespace

Uniform whitespace distribution



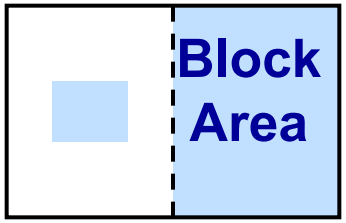
Both utilization < 1

Left partition Utilization = 1.0



Left util. = 1, right util. < 1

Right partition Utilization = 1.0

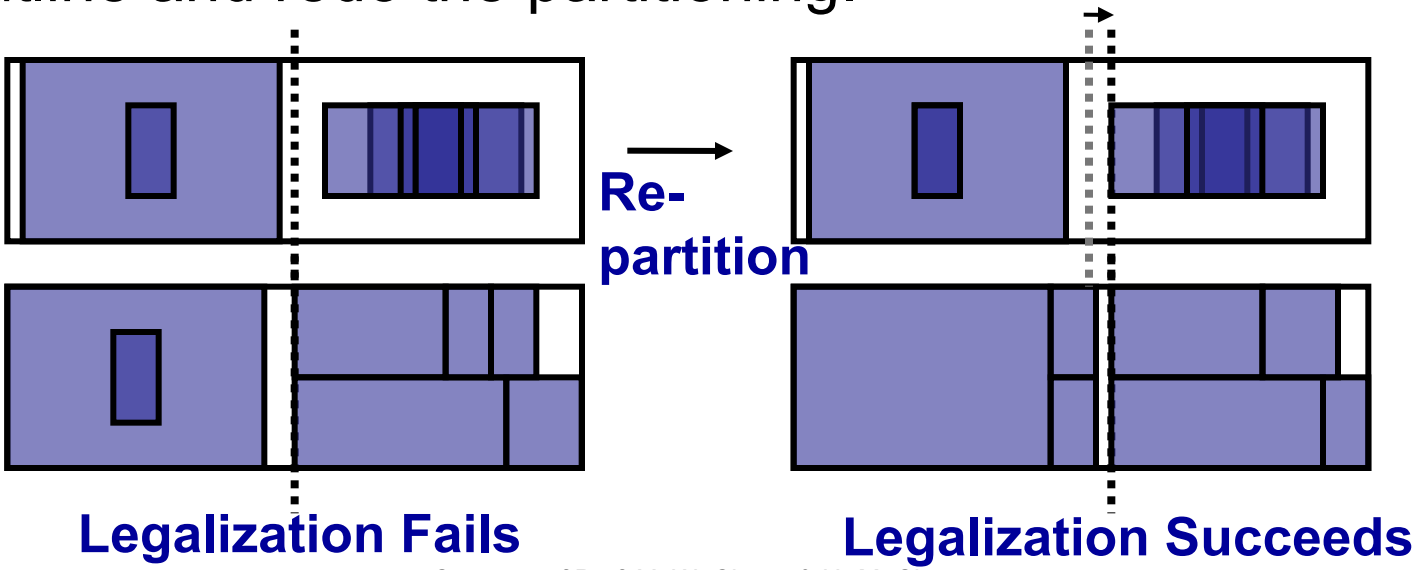


Left util. < 1, right util. = 1

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Look-Ahead Partitioning

- Simplify the idea in Cong et al., "Fast floorplanning by look-ahead enabled recursive bipartitioning," *ASPDAC-2005*
- Use the first-fit bin-packing heuristic to check if the subpartition can be legalized
  - Increase the chance of legalizing macro blocks
- If the subpartition cannot be legalized, we move the cutline and redo the partitioning.

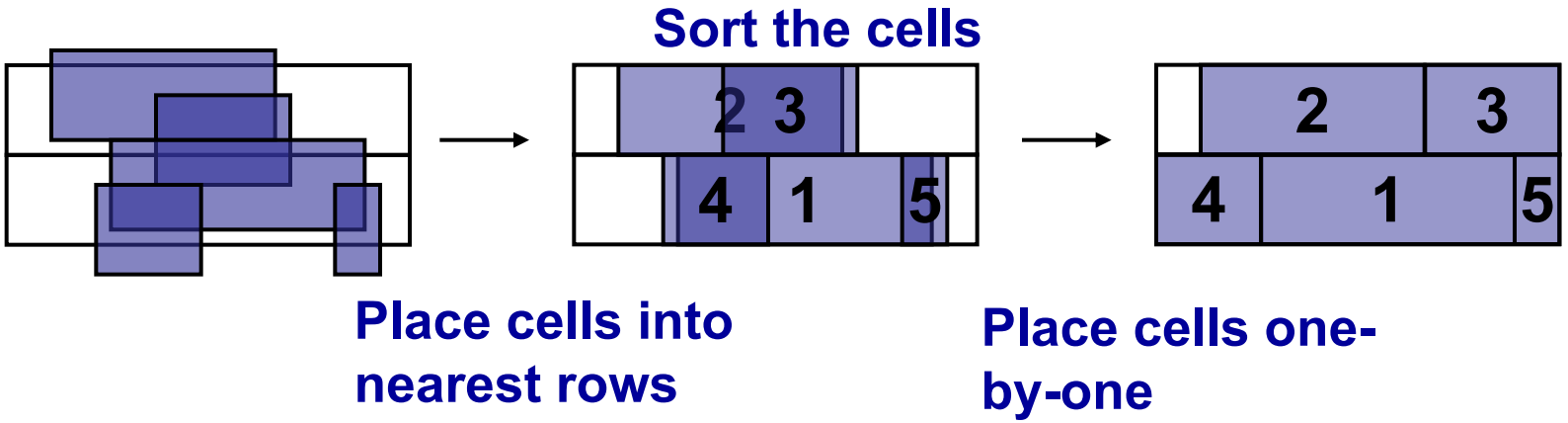


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Legalization

. Place all cells in the rows to obtain a feasible solution

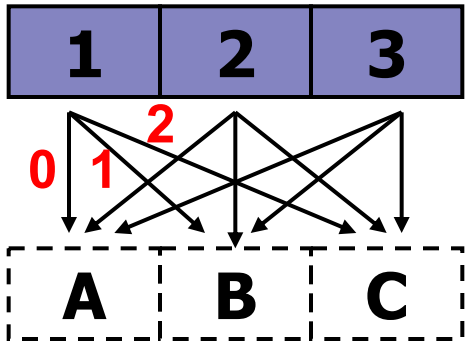
1) Place cells into their nearest rows
2) Sort all standard cells according to their sizes, from the largest to the smallest
3) Assign the x-coordinates for all cells according to the sorted order. If overlap occurs, we will find a nearest empty slot to place the cell



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Detailed Placement

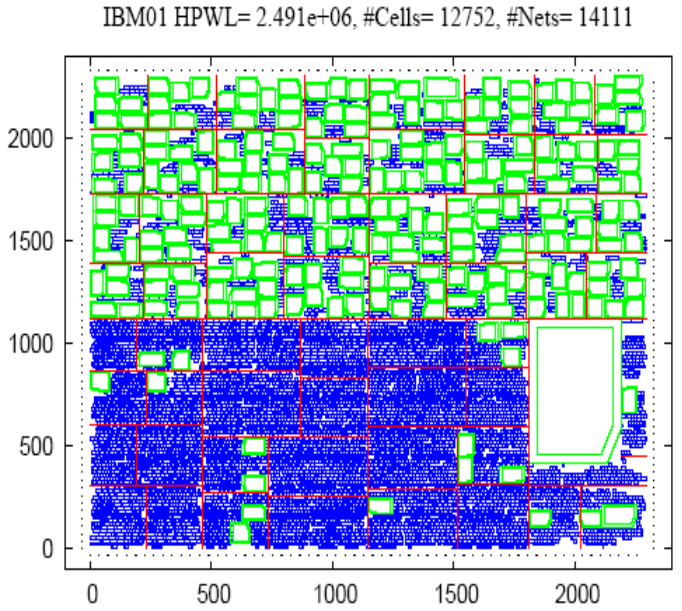
- Is based on cell location assignment (matching)
  - Each cell has different costs at different locations
  - Minimize total cost: O(n^3) time for n cells
  - Is better than O(n!) time for a branch & bound (BB) detailed placer
  - Can use a much larger window (> 64 cells)



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Capo Overview

- Pure recursive bi-partitioning placer 根據不同的數量用不同的partition
  - Multi-level FM for instances with > 200 cells
  - Flat FM for instances with 35-200 cells
  - Branch-and-bound for instances with < 35 cells



- Careful handling of partitioning tolerance
  - Uncorking: Prevent large cells from blocking smaller cells to move
  - Repartitioning: Several FM calls with decreasing tolerance
  - Block splitting heuristics: Higher tolerance for vertical cut
  - Hierarchical tolerance computation: Instance with more whitespace can have a bigger partitioning tolerance
- Implementation with good interface (LEF/DEF and GSRC bookshelf) available on web

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

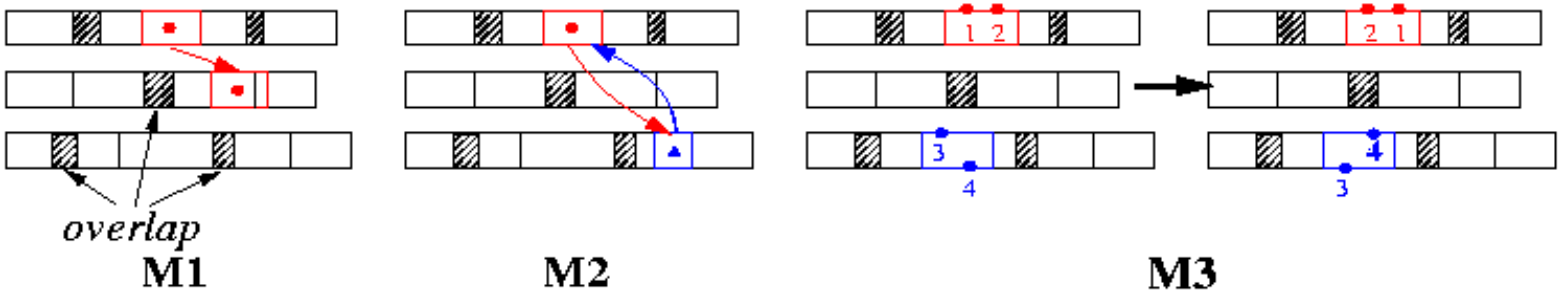
Placement by Simulated Annealing

- Sechen and Sangiovanni-Vincentelli, "The TimberWolf placement and routing package," *IEEE J. Solid-State Circuits*, Feb. 1985; "TimberWolf 3.2: A new standard cell placement and global routing package," DAC-86.
- iTools: http://www.twolf.com
- TimberWolf: Stage 1
  - Modules are moved between different rows as well as within the same row.
  - Modules overlaps are allowed.
  - When the temperature is reached below a certain value, stage 2 begins.
- TimberWolf: Stage 2
  - Remove overlaps.
  - Annealing process continues, but only interchanges adjacent modules within the same row.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

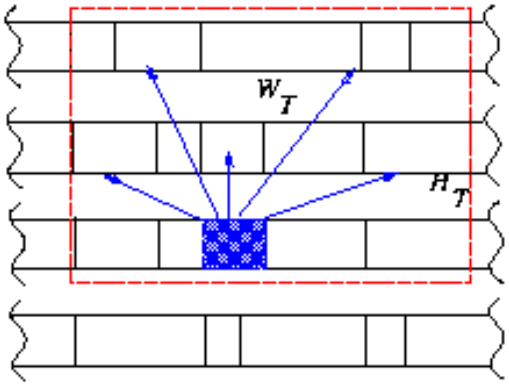
Solution Space & Neighborhood Structure

- **Solution Space:** All possible arrangements of the modules into rows, possibly with overlaps. 直接透過座標來代表solution
- **Neighborhood Structure:** 3 types of moves
  - M_1: Displace a module to a new location.
  - M_2: Interchange two modules.
  - M_3: Change the orientation of a module. 左右鏡射-->也許可以減少wire length



Neighborhood Structure

- TimberWolf first tries to select a move between M_1 and M_2: $Prob(M_1) = 0.8$, $Prob(M_2) = 0.2$.
- If a move of type M_1 is chosen and it is rejected, then a move of type M_3 for the same module will be chosen with probability 0.1.
- Restrictions: (1) what row for a module can be displaced? (2) what pairs of modules can be interchanged?
- **Key: Range Limiter**
  - At the beginning, (W_T, H_T) is big enough to contain the whole chip.
  - Window size shrinks as the temperature decreases. Height, width ∝ $log(T)$.
  - Stage 2 begins when window size is so small that no inter-row module interchanges are possible.



Cost Function

- Cost function: $C = C_1 + C_2 + C_3$.
- C_1: **total estimated wirelength**.
  - $C_1 = \sum_{i \in Nets}(\alpha_i w_i + \beta_i h_i)$
  - α_i, β_i are horizontal and vertical weights, respectively. (α_i=1, β_i =1 $\Rightarrow$(1/2) ✕ perimeter of the bounding box of Net *i*.)
  - Critical nets: Increase both α_i and β_i .
  - If vertical wirings are "cheaper" than horizontal wirings, use smaller vertical weights: $\beta_i < \alpha_i$.
- C_2: **penalty function for module overlaps.**
  - $C_2 = \gamma \sum_{i \neq j} O^2_{ij}$, γ: penalty weight.
  - O_{ij}: amount of overlaps in the *x*-dimension between modules *i* and *j*.
- C_3: **penalty function that controls the row length.**
  - $C_3 = \delta \sum_{r \in Rows}|L_r - D_r|$, δ : penalty weight.
  - D_r: desired row length.
  - L_r: sum of the widths of the modules in row *r*.

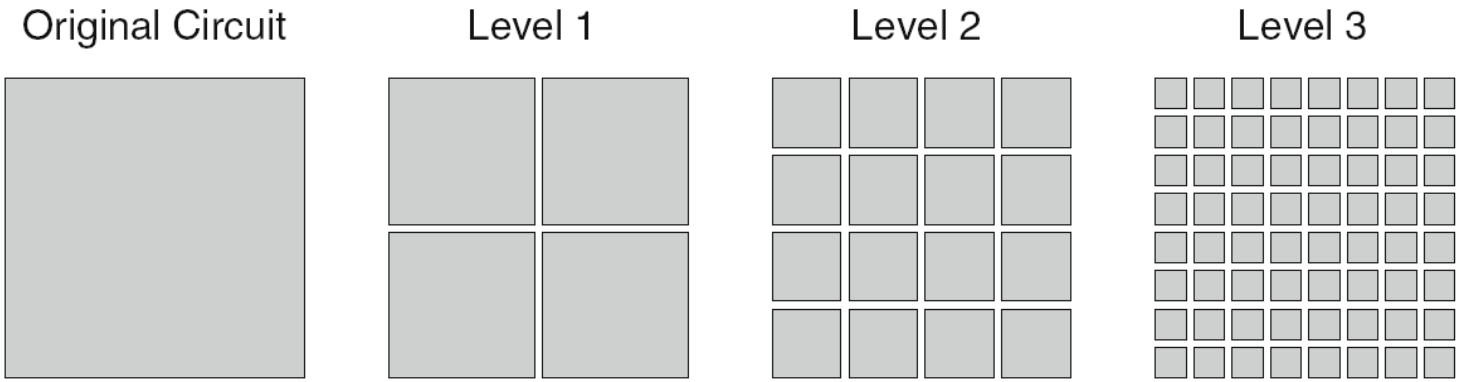
Annealing Schedule

- $T_k = r_k T_{k-1}$, $k = 1, 2, 3, \ldots$
- r_k increases from 0.8 to max value 0.94 and then decreases to 0.8.
- At each temperature, a total # of nP attempts is made.
- n: # of modules; P: user specified constant.
- Termination: $T < 0.1$.

Dragon Overview

原本在SA時是一個cell一個cell在進行move，這種方式是把move的東西改成partition分出來的東西

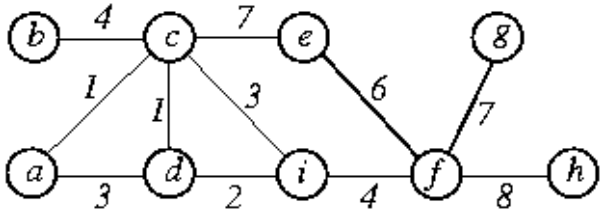
- Dragon takes a hybrid approach that combines simulated annealing and partitioning
  - Recursive partitioning to reduce the problem size
  - Annealing to refine the solution generated by partitioning
- Top-down hierarchical placement
  - hMetis to recursively quadrisect into 4^h bins at level h
  - Swapping of bins at each level by SA to minimize WL
  - Terminates when each bin contains < 7 cells
  - At final stage of GP, switch single cells locally among bins by SA to further minimize WL
- Detailed placement is done by greedy algorithm



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement by the Genetic Algorithm

- Cohoon & Paris, “Genetic placement,” ICCAD-86.
- **Genetic algorithm:** A search technique that emulates the biological evolution process to find the optimum.
- Generic approaches:
  - Start with an initial set of random configurations (**population**); each individual is a string of symbol (symbol string ↔ **chromosome**: a solution to the optimization problem, symbol ↔ **gene**).
  - During each iteration (**generation**), the individuals are evaluated using a **fitness** measurement.
  - Two fitter individuals (**parents**) at a time are selected to generate new solutions (**offsprings**).
  - Genetic operators: **crossover, mutation, inversion**
- In the example, string = [*aghcbidef*]; fitness value = $1/\sum_{(i,j)\in E} w_{ij}\, d_{ij} = 1/85$.



6	7	8
3	4	5
0	1	2

d	*e*	*f*
c	*b*	*i*
a	*g*	*h*

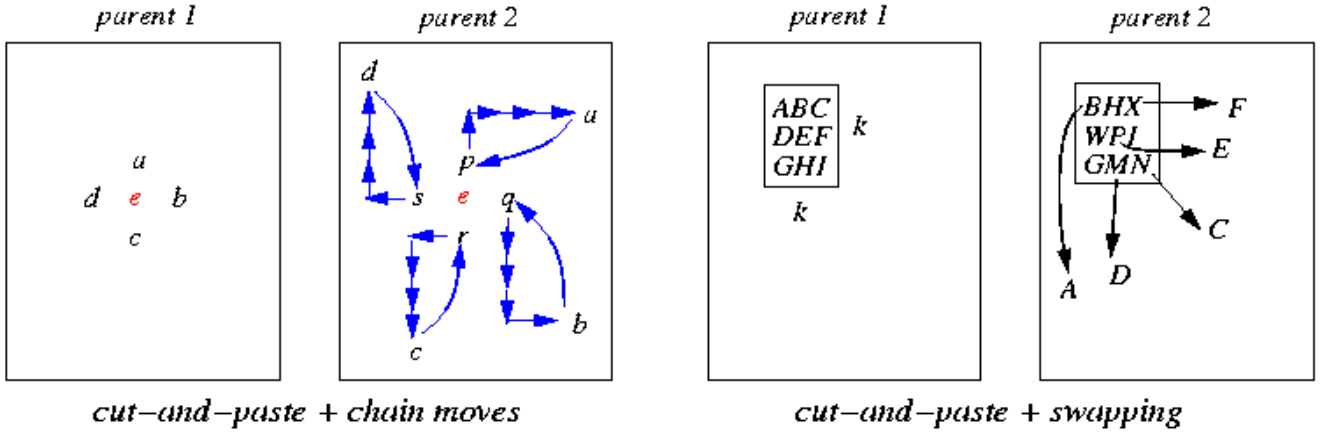
string: aghcbidef

Genetic Operator: Crossover

- Main genetic operator: Operate on two individuals and generates an offspring.
  - $[bidef|aghc](\frac{1}{86}) + [bdefi|gcha](\frac{1}{110}) \rightarrow [bidefgcha](\frac{1}{63}).$
  - Need to avoid repeated symbols in the solution string!
- **Partially mapped crossover** for avoiding repeated symbols:
  - $[bidef|gcha](\frac{1}{86}) + [aghcb|idef](\frac{1}{85}) \rightarrow [bgcha|idef].$
  - Copy *idef* to the offspring; scan [*bidef*|*gcha*] from the left, and then copy all unrepeated genes.

Two More Crossover Operations

- Cut-and-paste + Chain moves:
  - Copy a randomly selected cell *e* and its four neighbors from parent 1 to parent 2.
  - The cells that earlier occupied the neighboring locations in parent 2 are shifted outwards.
- Cut-and-paste + Swapping:
  - Copy $k \times k$ square modules from parent 1 to parent 2 (k: random # from a normal distribution with mean 3 and variance 1).
  - Swap cells not in both square modules.



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Genetic Operators: Mutation & Inversion

- **Mutation:** prevents loss of diversity by introducing new solutions.
  - Incremental random changes in the offspring generated by the crossover.
  - A commonly used mutation: pairwise interchange.
- **Inversion:** [*bid*|*efgch*|*a*] → [*bid*|*hcgfe*|*a*].
- Apply mutation and inversion with probability $P_{\square}$ and P_i respectively.

Algorithm: Genetic_Placement(N_p, N_g, N_o, P_i, P_m)

```
/* Np: population size; */
/* Ng: # of generation; */
/* No: # of offsprings; */
/* Pi : inversion probability; */
/* Pm: mutation probability; */

1 begin
2 ConstructPopulation(Np);  /* randomly generate the initial population */
3 for j ← 1 to Np
4     Evaluate Fitness(population(j));
5 for i ← 1 to Ng
6    for j ← 1 to No
7       (x, y) ← ChooseParents; /* choose parents with probability ∝ fitness value */
8       offspring(j) ← GenerateOffspring(x, y); /* perform crossover to generate offspring */
9      for h ← 1 to Np
10        With probability Pm, apply Mutation(population(h));
11     for h ← 1 to Np
12        With probability Pi, apply Inversion(population(h));
13     Evaluate Fitness(offspring(j));
14    population ← Select(population, offspring, Np);
15 return the highest scoring configuration in population;
16 end
```

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Genetic Placement Experiment: GINIE

- Termination condition: no improvement in the best solution for 10,000 generations.
- Population size: 50. (Each generation: 50 unchanged throughout the process.)
- Each generation creates 12 offsprings.
- Comparisons with simulated annealing:
  - Similar quality of solutions and running time.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

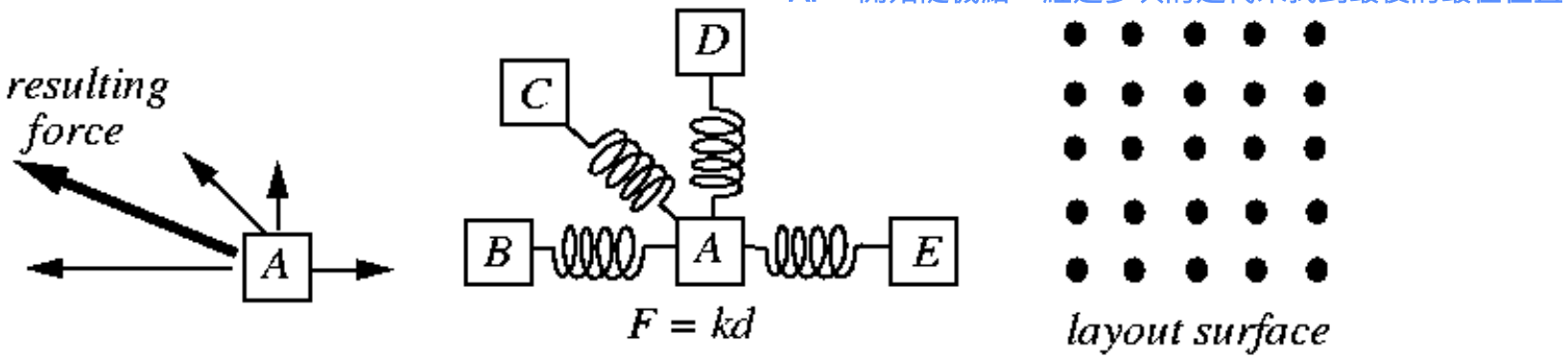
Analytical Placement

- Key: Solve a relaxed placement problem "optimally"
  - Ignore overlaps (fixed at later stages)
  - Adopt linear or nonlinear wirelength estimation
  - Need pads to pull the cells outwards
- Approaches for overlap removal
  - Cell spreading
  - Legalization
- Pro: Obtain very good quality for existing benchmarks
- Con: hard to handle cell rotation, big macros, region constraints, etc.
- Example tools
  - APlace (Kahng's group @ UCSD, ISPD-04, ICCAD-05)
  - FastPlace (Chu's group @ Iowa State, ISPD-04, ICCAD-05, DAC-07)
  - GORDIAN (Johannes's group @ TU of Munich, ICCAD-90)
  - Kraftwerk (Johannes's group @ TU of Munich, DAC-98, ICCAD-06)
  - mPL (Cong's group @ UCLA, ISPD-04, ISPD-05)
  - NTUplace3 (Chang's group @ Nat'l Taiwan U, ICCAD-06)

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement by the Force-Directed Method

- Hanan & Kurtzberg, “Placement techniques,” in *Design Automation of Digital Systems*, Breuer, Ed, 1972.
- Quinn, Jr. & Breuer, “A force directed component placement procedure for printed circuit boards,” *IEEE Trans. Circuits and Systems*, June 1979.
- Reducing the placement problem to solving a set of simultaneous linear equations to determine equilibrium locations for cells.
- Analogy to Hooke's law: $F = kd$, F: force, k: spring constant, d: distance. 用彈簧的概念實現floorplanning --> 嘗試用物理的觀念來解演算法問題
- Goal: Map cells to the layout surface. 已知BCDE的位置，可以根據彈力平衡來決定A的位置
  Q: 但是要如何得知BCDE的初始位置?
  A: 一開始隨機給，經過多次的迭代來找到最後的最佳位置



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

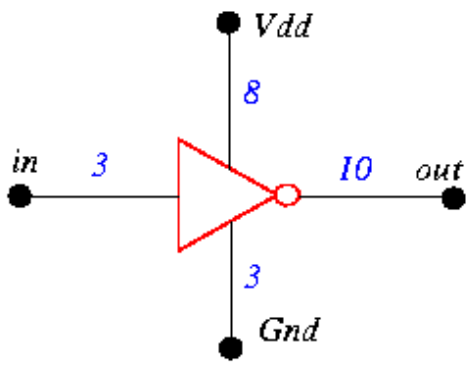
Finding the Zero-Force Target Location

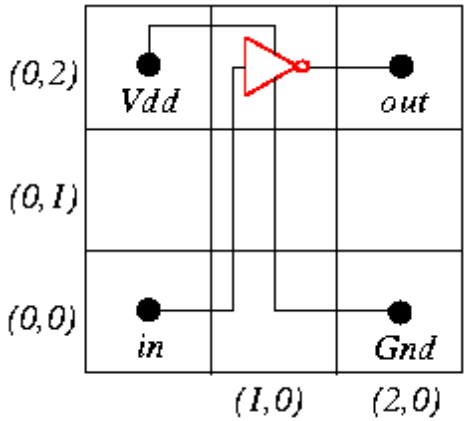
- Cell *i* connects to several cells *j*'s at distances d_{ij}'s by wires of weights w_{ij}'s. Total force: $F_i = \sum_j w_{ij} d_{ij}$
- The zero-force target location ($\hat{x}_i$, $\hat{y}_i$) can be determined by equating the *x*- and *y*-components of the forces to zero:

$$\sum_j w_{ij} \cdot (x_j - \hat{x}_i) = 0 \quad \Rightarrow \quad \hat{x}_i = \frac{\sum_j w_{ij} x_j}{\sum_j w_{ij}}$$

$$\sum_j w_{ij} \cdot (y_j - \hat{y}_i) = 0 \quad \Rightarrow \quad \hat{y}_i = \frac{\sum_j w_{ij} y_j}{\sum_j w_{ij}}$$

- In the example, $\hat{x}_i = \frac{8\times0+10\times2+3\times0+3\times2}{8+10+3+3} = 1.083$ and $\hat{y}_i$ = 1.50.





Force-Directed Placement

- Approach I (constructive):
  - Start with an initial placement.
  - Compute the zero-force locations for all cells.
  - Apply **linear assignment (matching)** to determine the "ideal" locations for the cells.
- Approach II (can be constructive or iterative):
  - Start with an initial placement.
  - Select a "most profitable" cell *p* (e.g., maximum *F*, critical cells) and place it in its zero-force location.
  - "Fix" placement if the zero-force location has been occupied by another cell *q*.
- Popular options to fix:
  - **Ripple move:** place *p* in the occupied location, compute a new zero-force location for *q*, …
  - **Chain move:** place *p* in the occupied location, move *q* to an adjacent location, …
  - Move *p* to a free location close to *q*.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

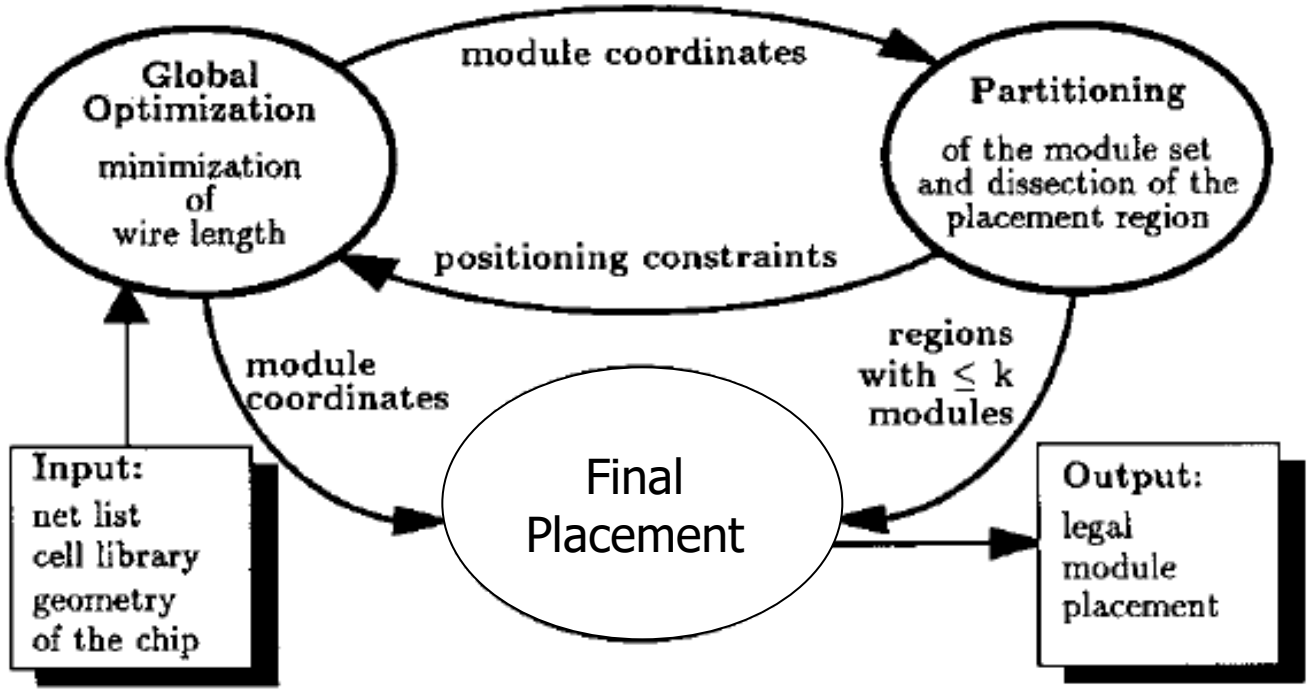
Algorithm: Force-Directed_Placement

```
1 begin
2 Compute the connectivity for each cell;
3 Sort the cells in decreasing order of their connectivities into list L;
4 while (IterationCount < IterationLimit) do
5     Seed ← next module from L;
6     Declare the position of the seed vacant;
7         while (EndRipple = FALSE) do
8             Compute target location of the seed;
9             case the target location
10            VACANT:
11                Move seed to the target location and lock;
12                EndRipple ← TRUE; AbortCount ← 0;
13            SAME AS PRESENT LOCATION:
14                EndRipple ← TRUE; AbortCount ← 0;
15            LOCKED:
16                Move selected cell to the nearest vacant location;
17                EndRipple ← TRUE; AbortCount ← AbortCount + 1;
18                if (AbortCount > AbortLimit) then
19                    Unlock all cell locations;
19                    IterationCount ← IterationCount + 1;
20            OCCUPIED AND NOT LOCKED:
21                Select cell as the target location for next move;
22                Move seed cell to target location and lock the target location;
23                EndRipple ← FALSE; AbortCount ← 0;
26 end
```

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

最早使用analytical placement的方法GORDIAN Placement

- Kleinhans, Sigl and Johannes, “GORDIAN: VLSI placement by quadratic programming and slicing optimization,” TCAD, October 91 (ICCAD-90). 雖然quality沒到最好，但是速度很快
- Global placement (analytical scheme) + detailed placement (combinatorial search)



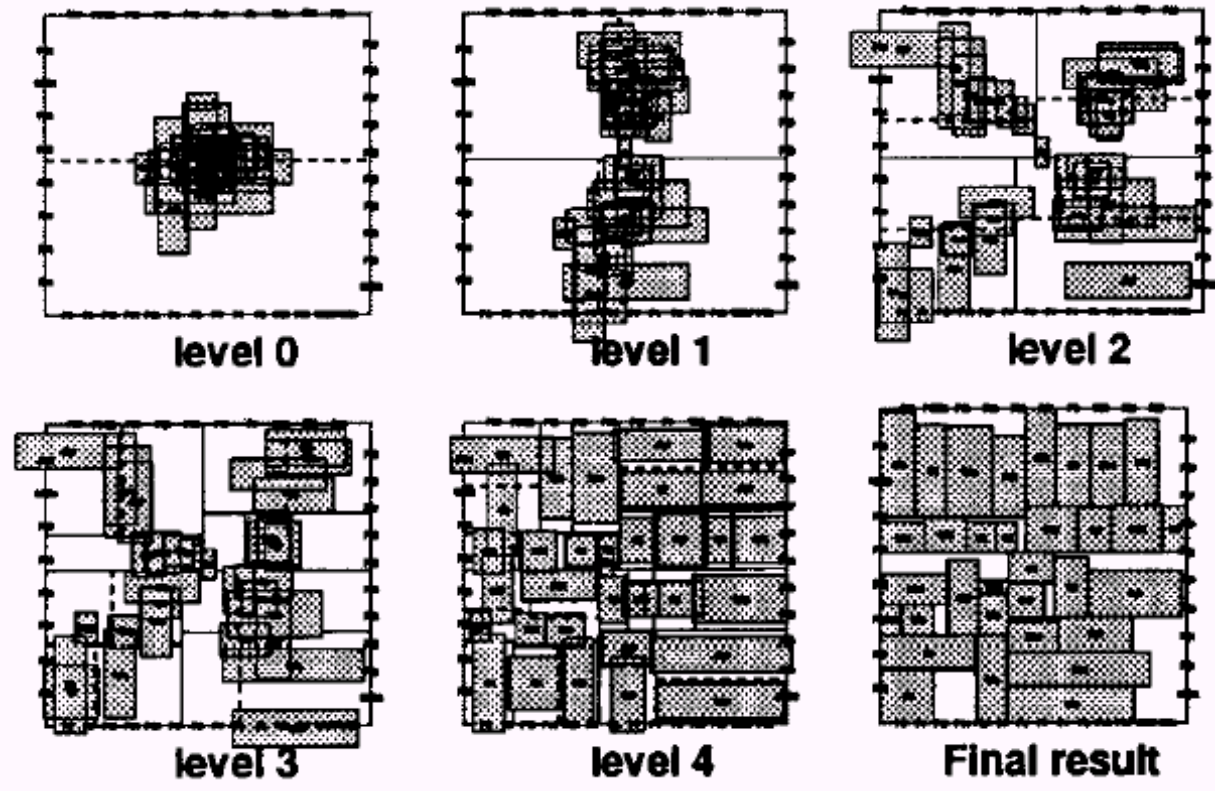
Data flow in the placement procedure GORDIAN.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Global Placement Optimization

- Apply quadratic programming.
- Objective function: wirelength.
  - Squared distance between modules and nets.
- Constraints: the centers of the regions on a level of partitioning for the movable modules.

level 0: 先做analytical placement得到初始座標(都很靠近中間)
level 1: 做partition，將level 0中比較偏上的cell都往上面的set擺，剩下的就放下面
level 2~n: 繼續partition，然後這些cell就會慢慢散開



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Objective Function

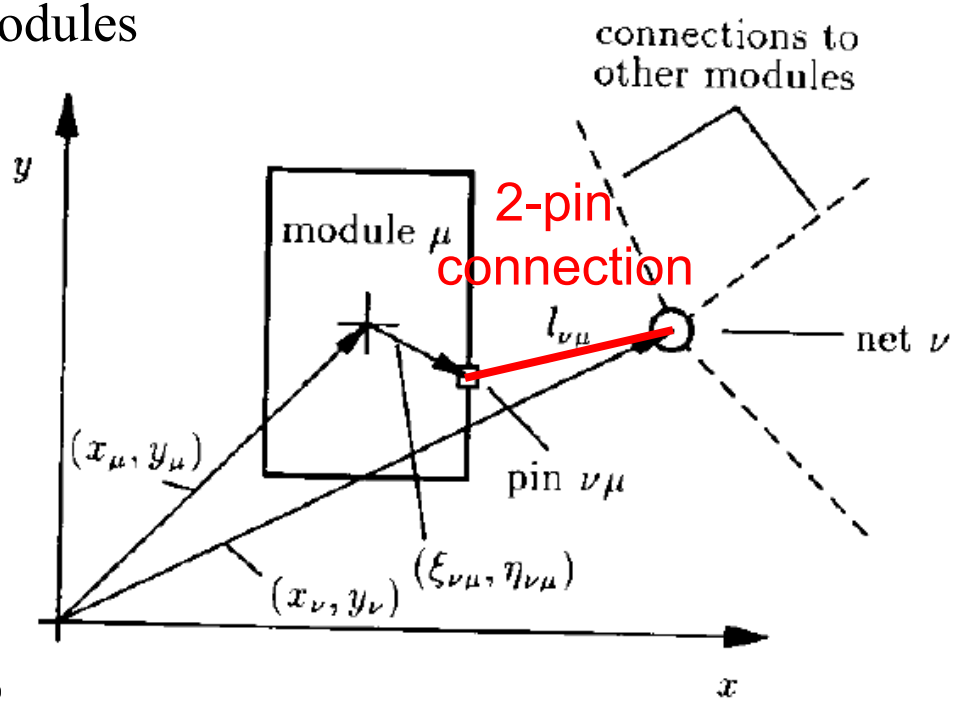
. Objective function: Squared distance between modules and nets. $\phi(x,y) = \frac{1}{2}\sum_{\nu\in\mathcal{N}}\sum_{\mu\in\mathcal{M}}\left[(x_\mu + \xi_{\nu\mu} - x_\nu)^2 + (y_\mu + \eta_{\nu\mu} - y_\nu)^2\right]\cdot t_{\nu\mu}\cdot w_\nu$

- w_v is the weight of net v.
- t_{vu} = 1 if net v connects to module u; t_{vu} = 0, otherwise.
- $\phi(x, y) = \frac{1}{2}x^T Q x - d_x^T x + \frac{1}{2}y^T Q y - d_y^T y,$ 實作的時候x,y是分開做的

x and y: coordinate of modules

Q: connectivity matrix,

d: originates from the contributions of the pads and the relative pin coordinates.



More on the Objective Function

(x_i, y_i) ≡ Coordinates of the center of cell i

c_{ij} ≡ Weight of the net between cells i and j

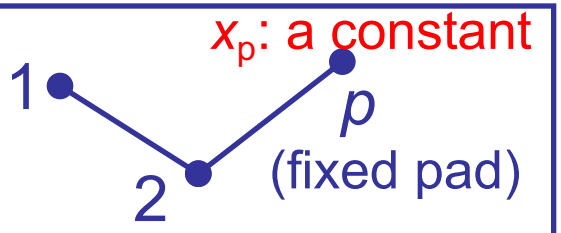
$\mathbf{x}, \mathbf{y}$ ≡ Solution vectors

Cost of the net between cells i and j

$$\equiv \frac{1}{2} c_{ij} \left[(x_i - x_j)^2 + (y_i - y_j)^2 \right]$$

$$\text{Total cost} \equiv \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + d_x^T \mathbf{x} + \frac{1}{2} \mathbf{y}^T Q \mathbf{y} + d_y^T \mathbf{y} + \text{constant}$$

Horizontal cost



$$\equiv \frac{1}{2} c_{12} (x_1 - x_2)^2 + \frac{1}{2} c_{2p} (x_2 - x_p)^2$$

$$\equiv \frac{1}{2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{pmatrix} c_{12} & -c_{12} \\ -c_{12} & c_{12} + c_{2p} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{bmatrix} 0 & -c_{2p} x_p \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2} c_{2p} x_p^2$$

Solution of Quadratic Placement

$$\text{Total cost } \tilde{L} = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + d_x{}^T\mathbf{x} + \frac{1}{2}\mathbf{y}^T Q\mathbf{y} + d_y{}^T\mathbf{y} + \text{const}$$

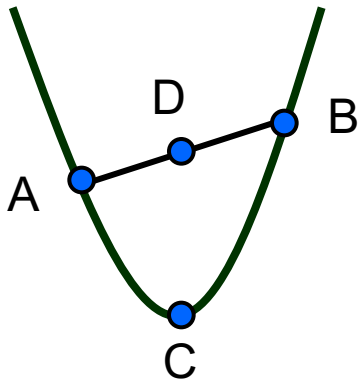
- The problems in x- and y-directions can be separated and solved independently
- *Ignore non-overlapping and other constraints for the time being*
- Minimize $\tilde{L}_x = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + d_x{}^T\mathbf{x}$
- Q can be proved to be positive definite
- Thus, the function is convex
- Minimum solution can be found by setting derivatives to 0:

$$Q\mathbf{x} + d_x = \mathbf{0}$$

Convexity

- The objective function is a convex quadratic function.
- Convexity:
  - If **x** is a single variable, $f''(\mathbf{x}) \geq 0$.
  - If $\mathbf{x} = [x_1, x_2, ..., x_n]$, $\mathbf{x}^T\mathbf{H}_f\mathbf{x} \geq 0$.

$$H_f \square \begin{Bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & & \vdots \\ \vdots & & & \vdots \\ f_{n1} & \cdots & \cdots & f_{nn} \end{Bmatrix}$$



A: $(x, f(x))$

B: $(x', f(x'))$

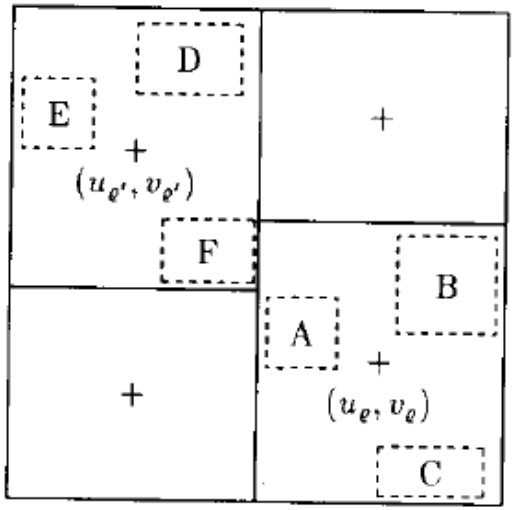
C: $(cx + (1-c)x', f(cx + (1-c)x'))$

D: $(cx + (1-c)x', cf(x) + (1-c)f(x'))$

Convex: $cf(x) + (1-c)f'(x') \geq f(cx + (1-c)x')$

Constraints

- Constraints: make the center of module gravity correspond to the center of the region, $A^{(l)}x \square u^{(l)}$; each entry in A gives the area ratio of the module in the corresponding region.
- Constraints are linear (and thus also convex).



$$\mathbf{A}^{(l)} = \begin{array}{c} \\ \vdots \\ \varrho \\ \varrho' \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & \cdots \end{array} \\ \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & * & * & * & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$

$$x_a f_a + x_b f_b + x_c f_c = u_\rho (f_a + f_b + f_c)$$

$$x_d f_d + x_e f_e + x_f f_f = u_{\rho'} (f_d + f_e + f_f)$$

f_m: the area of module m, u_ρ: the center of a region ρ,
x_m: the coordinate of the center of module m.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Quadratic Programming

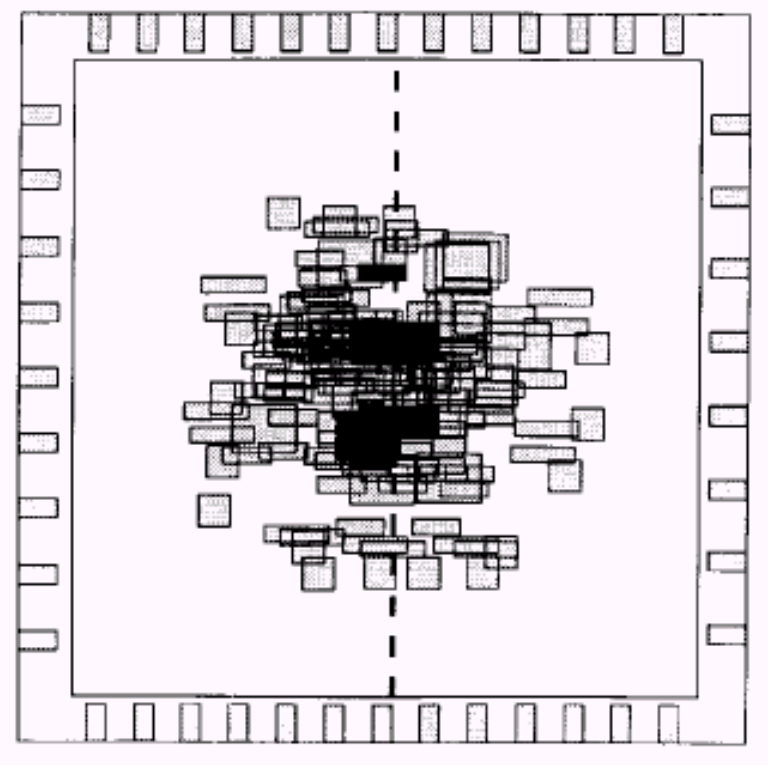
- The objective function is a convex quadratic function.
  - C is positive definite.
- Constraints are linear (and thus convex).
- QP has a unique global minimum $\phi(x^*)$.
- QP can be solved optimally in polynomial time.
- Ignore the non-overlapping constraint at the global placement stage.
- The problem in x- and y-directions can be separated and solved independently.

$$\text{QP: } \min_x \{\phi(x) \square \frac{1}{2} x^T C x + d^T x \,|\, A^{(l)} x \square u^{(l)}\}.$$

- Minimum solution can be found by setting derivatives to 0.

Partitioning

- After global optimization, modules are divided into two parts such that the module areas of both subsets are about the same.
  - Sort the x- (y-) coordinates of the modules and cut the region vertically (horizontally).



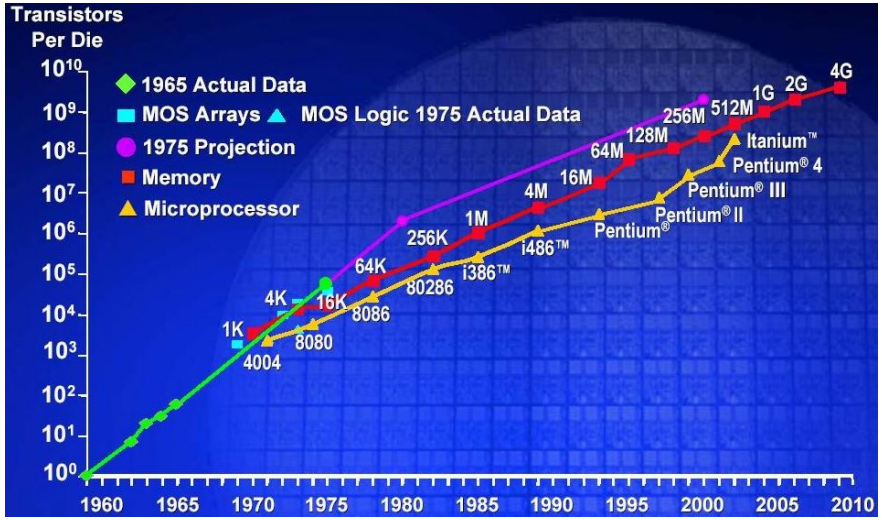
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Exhaustive Slicing Optimization

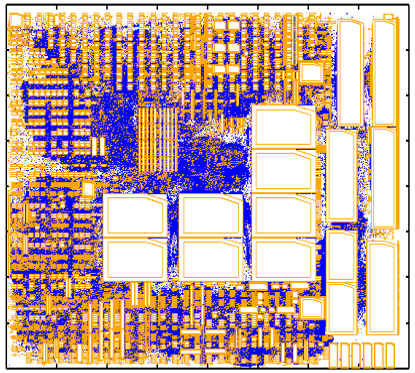
- Combine enumeration of slicing subtrees with global optimization and evaluation of shape functions.
- Enumerate all possible subtrees for small subsets of modules.
  - t(k) k! placements need to be evaluated for t(k) subtrees.
    - t(2) = 2, t(3) = 6, t(4) = 22, …
  - Time for k! can be saved by global optimization.
- Allocation of modules to regions is derived from the coordinates of the modules in the last global optimization phase.
- All possible shapes of all exhaustively optimized regions simultaneously contribute to the final placement.

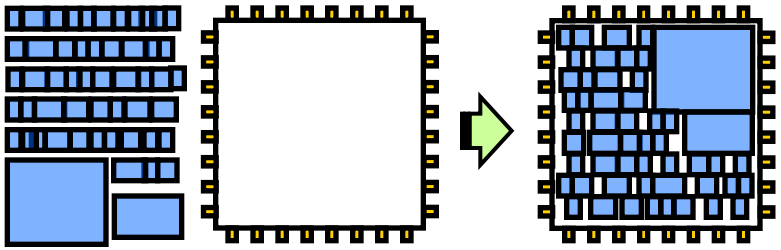
Modern Placement Challenges

- High complexity
  - Millions of objects to be placed
- Mixed-size placement
  - Hundreds of movable macro blocks with millions of small standard cells
- Placement constraints
  - Preplaced blocks
  - Chip density
  - etc.



2.5M placeable objects

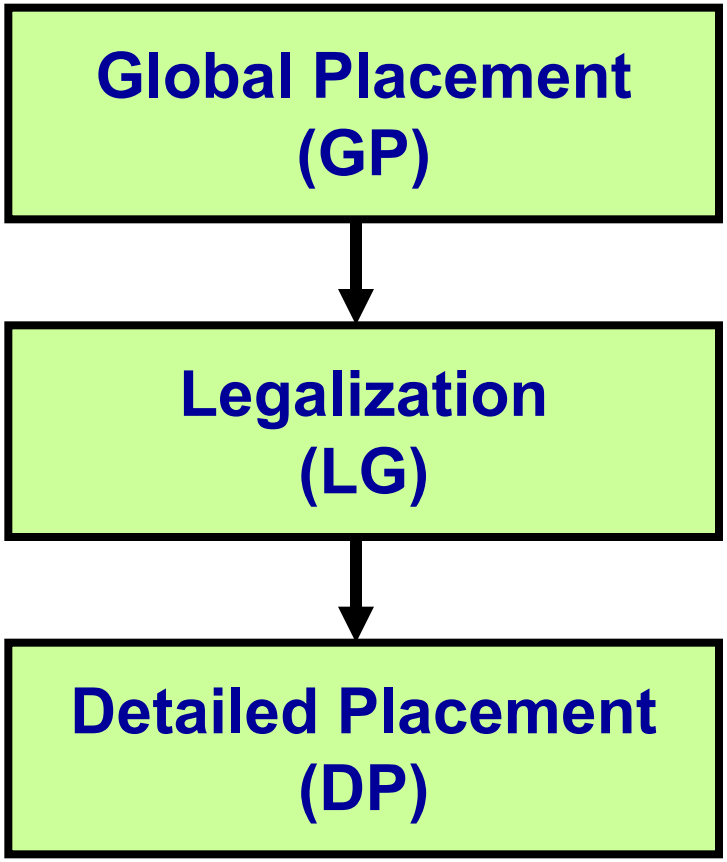


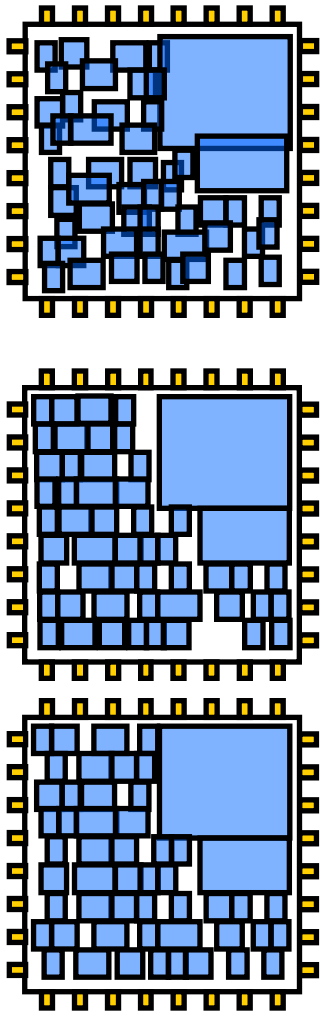


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

NTUplace3 Standard-Cell Placement Flow

- Chen, et al., "A high quality analytical placer considering preplaced blocks and density constraint," ICCAD-06 (TCAD-08).





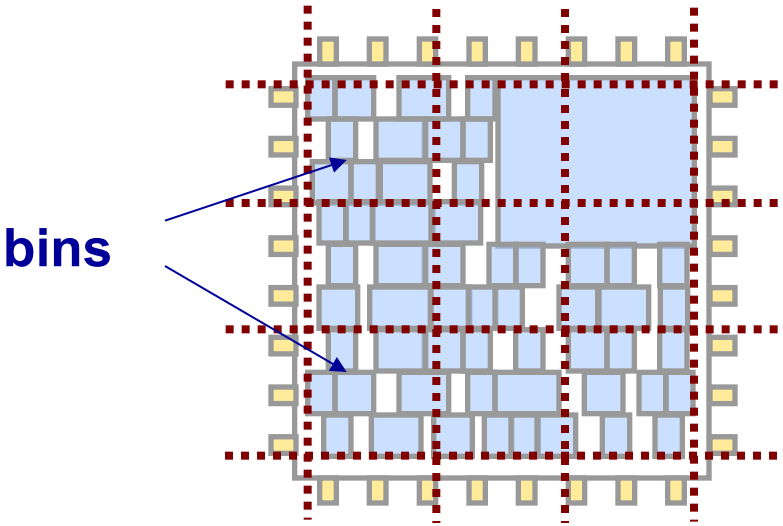
prototyping

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement with Density Constraint

- Given the chip region and block dimensions, divide the placement region into bins
- Determine (x, y) for all movable blocks

min W(x, y) -- wirelength function
s.t. 1. $\text{Density}_b(x, y) \leq \text{MaximumDensity}_b$ for each bin b
2. No overlap between blocks

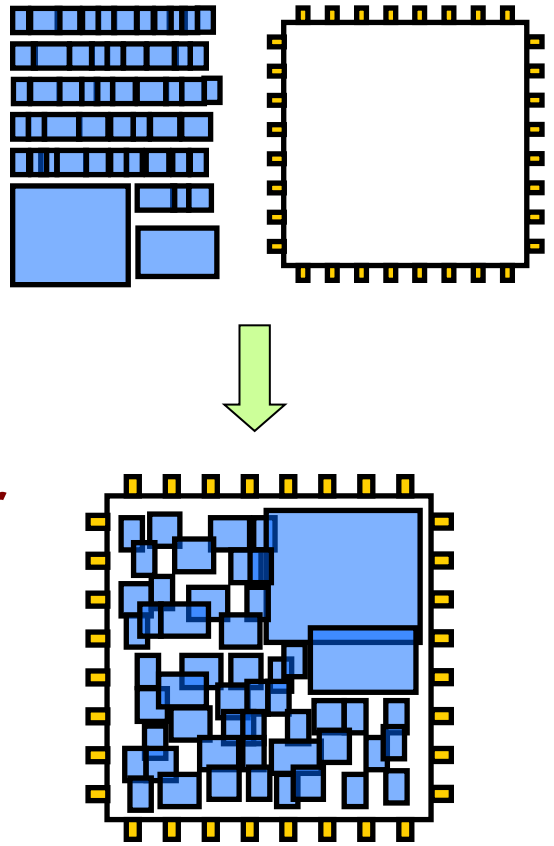


$$\text{Density} = \frac{A_{block}}{A_{bin}}$$

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Global Placement

- Placement flow
  - ***Global placement***
    - ***Multilevel framework***
    - ***Analytical placement with a nonlinear objective function***
    - ***Free-space allocation for density control***
    - ***Smoothing functions for preplaced blocks***
  - Legalization
  - Detailed placement

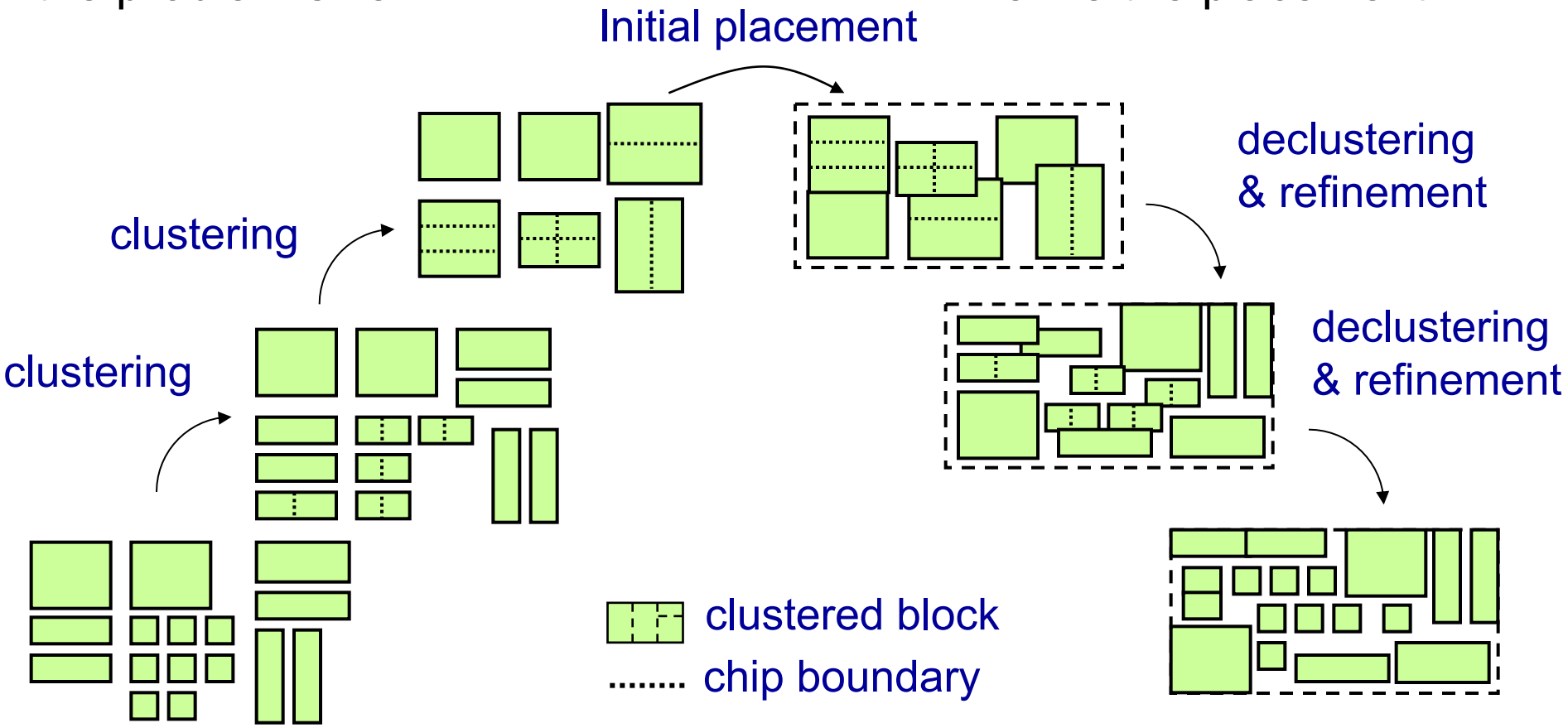


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Multilevel Global Placement

Cluster the blocks based on connectivity/size to reduce the problem size.

Iteratively decluster the clusters and further refine the placement



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Analytical Placement Model

- **Analytical placement during declustering**
- **Global placement problem (allow overlaps)**

min $W(x, y)$
s.t. $D_b(x, y) \le M_b$

Minimize HPWL
D_b: density for bin b
M_b: max density for bin b

- **Relax the constraints into the objective function**

min $W(x, y) + \boldsymbol{\lambda}\Sigma(D_b(x, y) - M_b)^2$

  - Use the gradient method to solve it
  - Increase **λ** gradually to find the optimal (x, y)

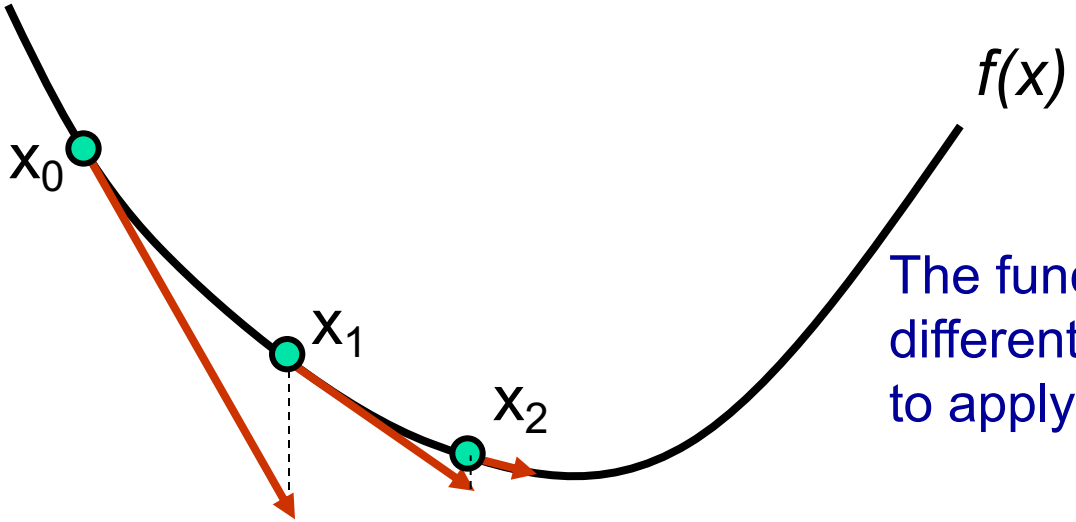
Gradient Solver

min *f(x)*

[Gradient Solver]

$x_0 \leftarrow$ initial value
Repeat until convergence

$$x_{i+1} = x_i - f'(x)|_{x=xi} * \text{stepsize}$$



The function *f* must be differentiable and smooth to apply the gradient solver.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

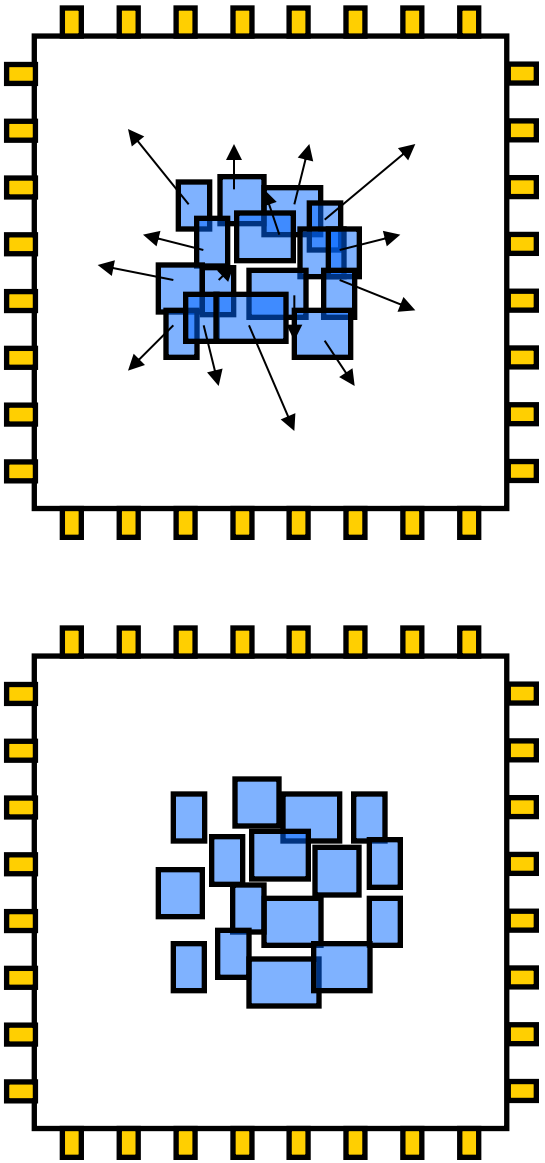
Dynamic Step-Size Control

- Step size is too large
  - → May not converge to a good solution
- Step size is too small
  - → Incur long running time
- Adjust the step size s.t. the average Euclidean movement of all blocks is a fixed value

$$\text{step size } \alpha_k = \frac{s}{\|\mathbf{d}_k\|_2}$$

$\mathbf{d}_k$ conjugate directions

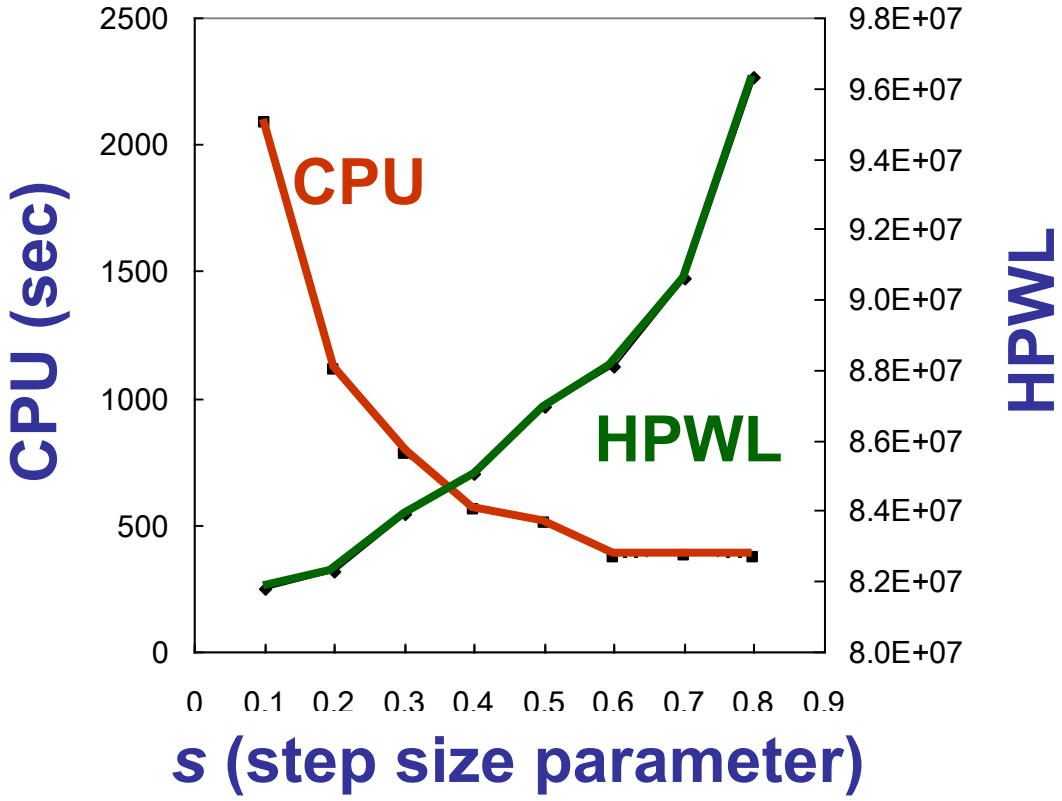
s a user-specified value



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Effects of the Step-Size Parameter

- Small step size
  - Larger runtime, smaller HPWL
- Large step size
  - Smaller runtime, larger HPWL



Wirelength Model

- Half perimeter wirelength (HPWL): not smooth/differentiable

$$W(\mathbf{x},\mathbf{y}) = \sum_{\text{net } e} \left(\max_{v_i, v_j \in e} |x_i - x_j| + \max_{v_i, v_j \in e} |y_i - y_j| \right)$$

- Apply the **log-sum-exp wirelength model**: NTUplace3, mPL, APlace

$$\gamma \sum_{e \in E} \left(\log \sum_{v_k \in e} \exp(x_k/\gamma) + \log \sum_{v_k \in e} \exp(-x_k/\gamma) + \right.$$
$$\left. \log \sum_{v_k \in e} \exp(y_k/\gamma) + \log \sum_{v_k \in e} \exp(-y_k/\gamma) \right).$$

  - Is a patented method proposed by Naylor et al.
  - Is an effective smooth & differentiable function for HPWL approximation
  - Approaches exact HPWL when γ → 0
  - Achieves the best result among recent academic placers

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Other Smooth Wire Models

- Quadratic model: BonnPlace, FastPlace, Kraftwerk, mFAR
  - Not exact wirelength modeling

$$\sum_{e \in E} \left(\sum_{v_i, v_j \in e, i<j} w_{ij}(x_i - x_j)^2 + \sum_{v_i, v_j \in e, i<j} w_{ij}(y_i - y_j)^2 \right)$$

- Lp-norm model
  - Approaches exact HPWL when p → ∞
  - Needs more time to evaluate the exponential function

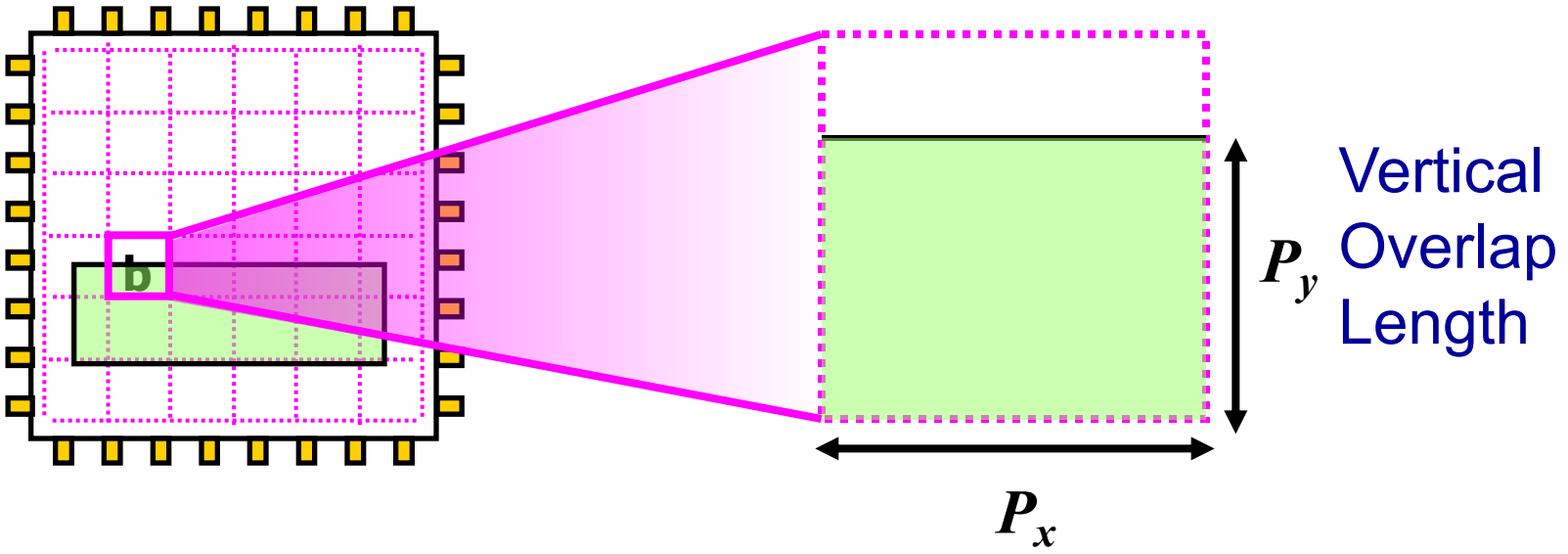
$$\sum_{e \in E} \left(\left(\sum_{v_k \in e} x_k^p \right)^{\frac{1}{p}} - \left(\sum_{v_k \in e} x_k^{-p} \right)^{-\frac{1}{p}} + \left(\sum_{v_k \in e} y_k^p \right)^{\frac{1}{p}} - \left(\sum_{v_k \in e} y_k^{-p} \right)^{-\frac{1}{p}} \right)$$

- Typical quality & speed ranking: **log-sum-exp > Lp-norm > Quadratic**

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Density Model

. Compute the block area in each bin to obtain the bin density

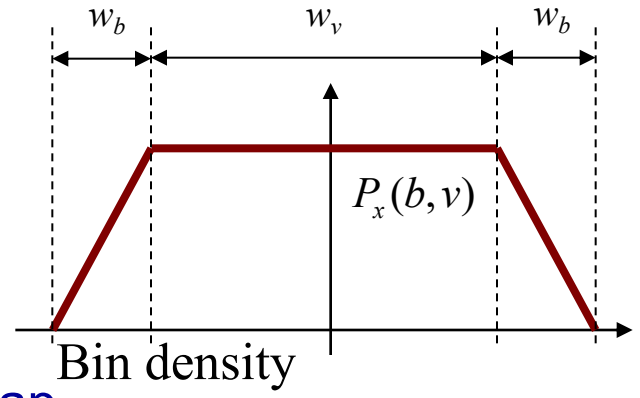


Bin density

$$D_b(\mathbf{x},\mathbf{y}) \square \sum_{v \in V} P_x(b,v) P_y(b,v)$$

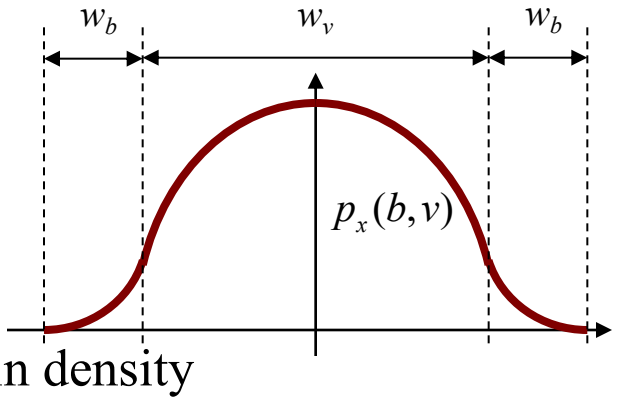
Density Smoothing

- Apply the bell-shaped function to make bin density function smooth (Kahng & Wang)



Bell-shaped smooth function

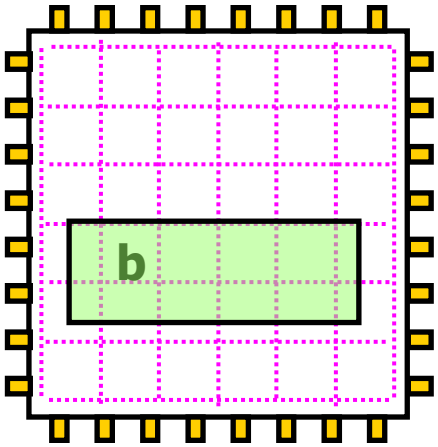
Continuous & differentiable



Overlap length function

$$D_b(\mathbf{x},\mathbf{y}) = \sum_{v \in V} P_x(b,v) P_y(b,v)$$

$$D'_b(\mathbf{x},\mathbf{y}) = \sum_{v \in V} c_v p_x(b,v) p_y(b,v)$$



$$p_x(b,v) = \begin{cases} 1 - ad_x^2, & 0 \le d_x \le w_v/2 + w_b \\ b(d_x - 2w_b - 2w_g)^2, & w_v/2 + w_b \le d_x \le w_v/2 + 2w_b \\ 0, & w_v/2 + 2w_b \le d_x, \end{cases}$$

where

$$a = 4/((w_v + 2w_b)(w_v + 4w_b))$$
$$b = 2/(w_b(w_v + 4w_b)),$$

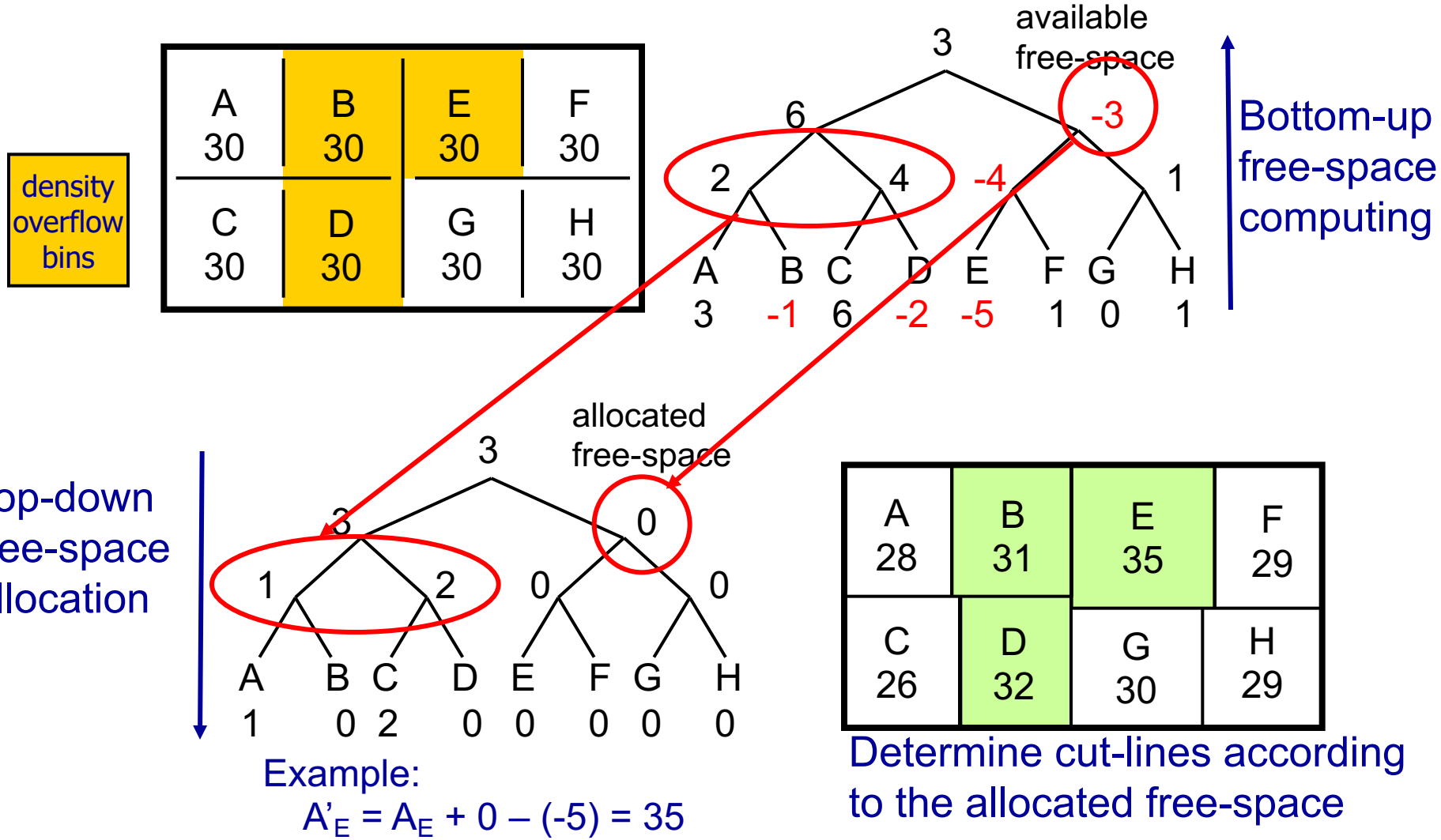
w_b bin width

w_v block width

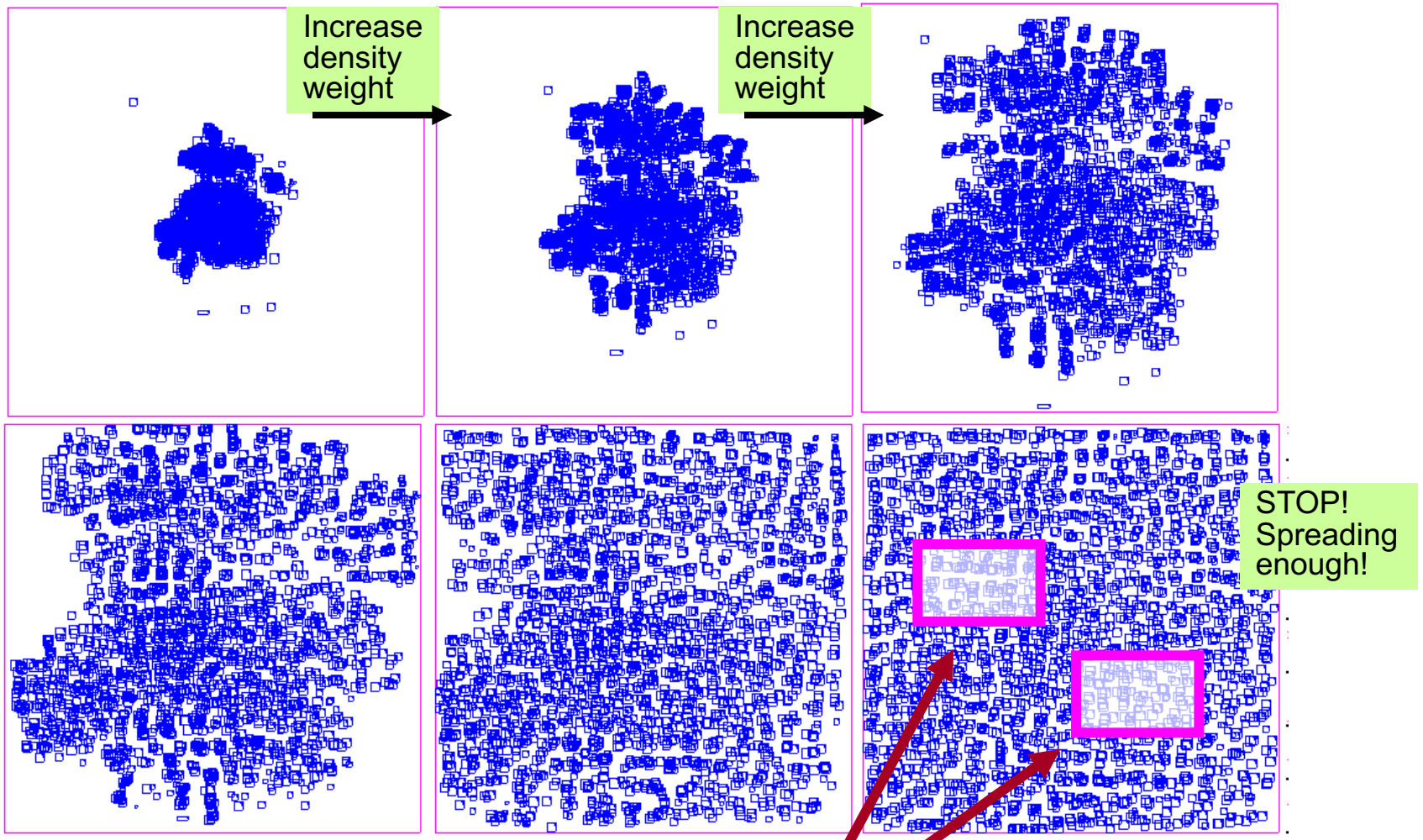
c_v normalization factor

Free-Space Allocation for Density Control

- Reallocate free-space to the density overflow bins



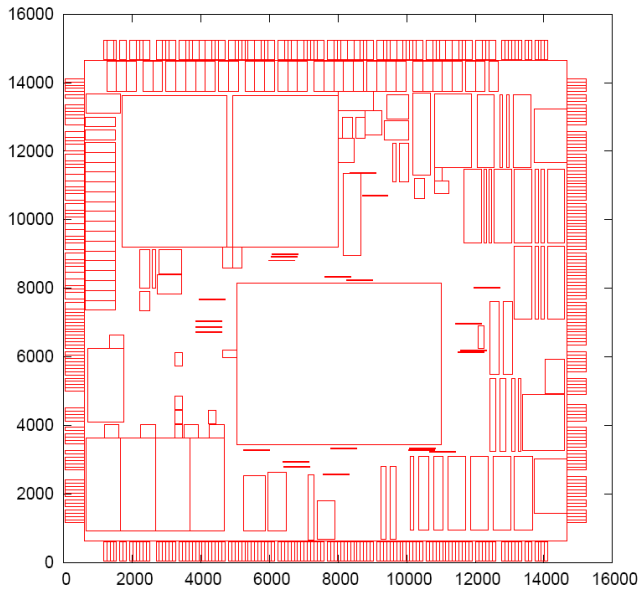
Placement Process

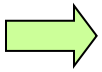


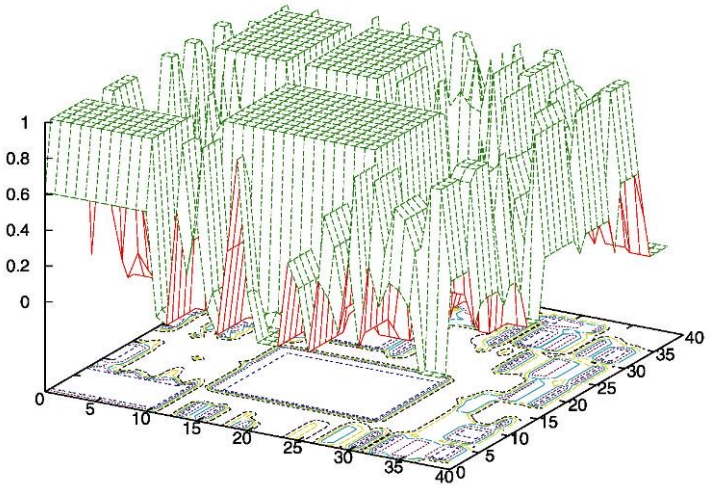
- How to handle preplaced blocks?
  - Pre-defined density makes cell spreading harder.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Density Map with Preplaced Blocks







Placement with preplaced blocks

Corresponding density map

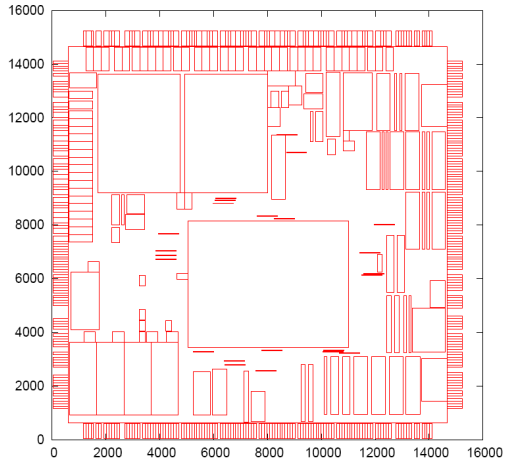
Two problems

☹ the density map is not smooth

☹ the density is too high to spread movable blocks over the mountains

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

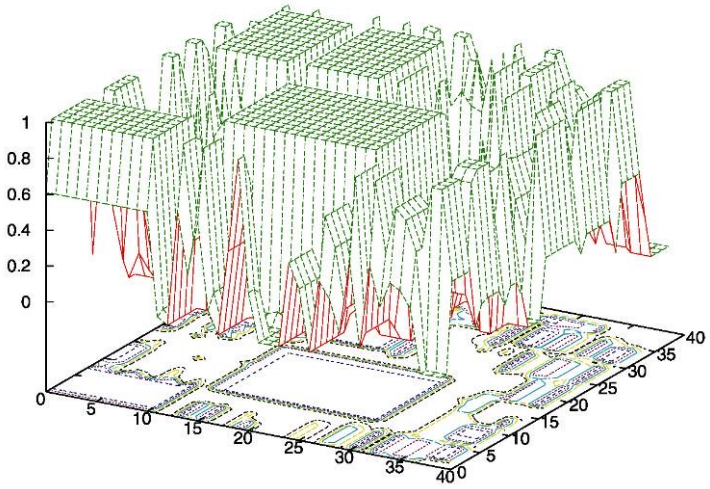
Bell-Shaped Block Smoothing??



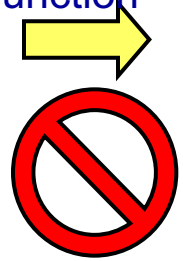
☹ mountain heights are quite different

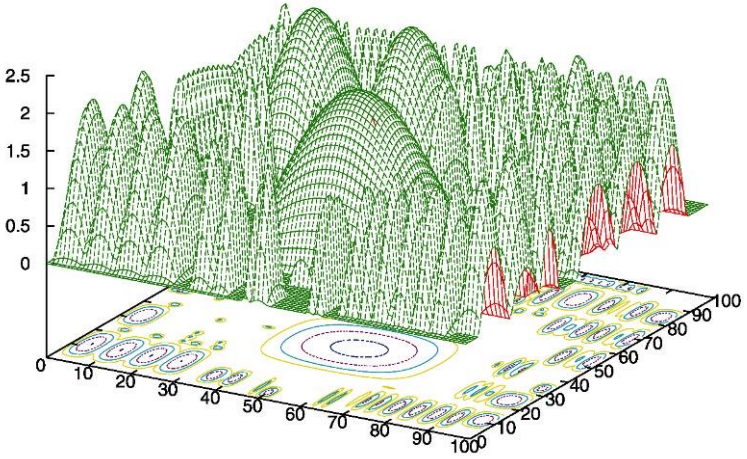
☹ there are many valleys among mountains

➔ might mis-guide the movement of blocks



Apply the bell-shaped function

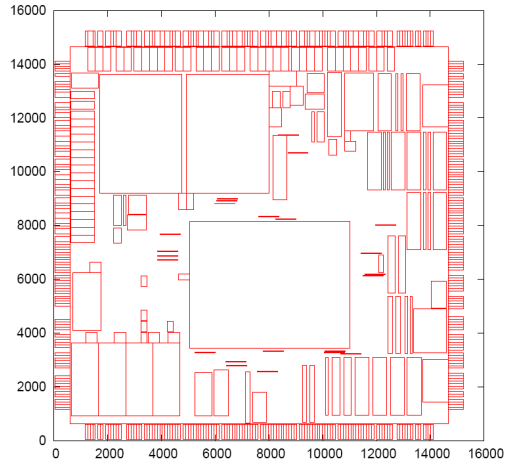


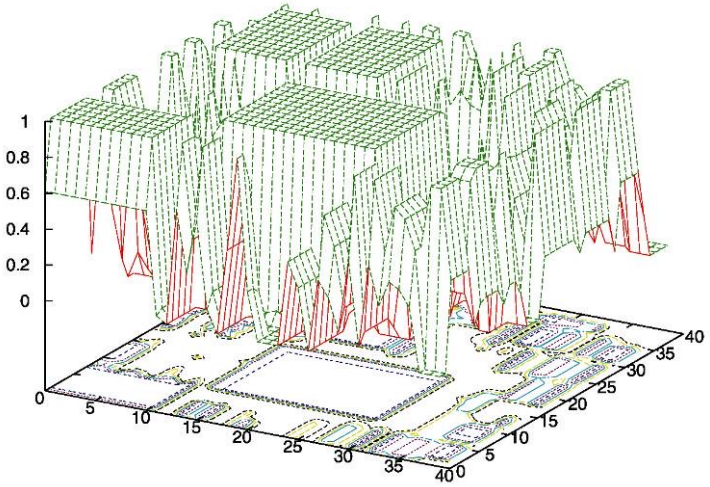


Bell-shaped smoothing (Kahng et al.)

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Preplaced Block Smoothing

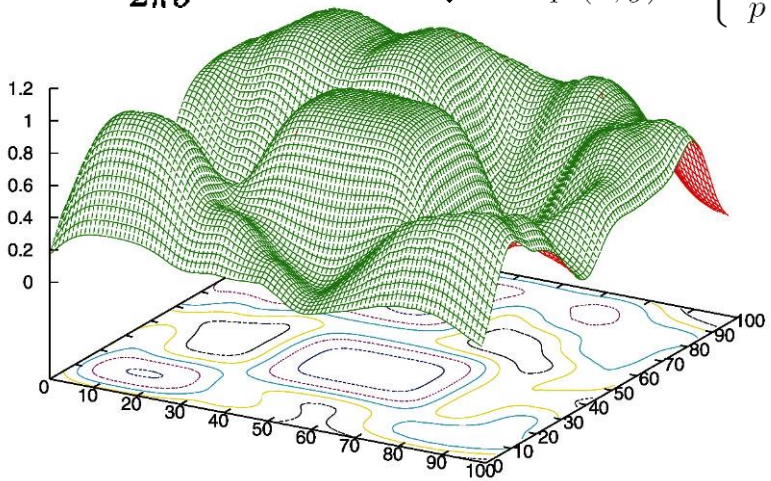




Convolute with Gaussian

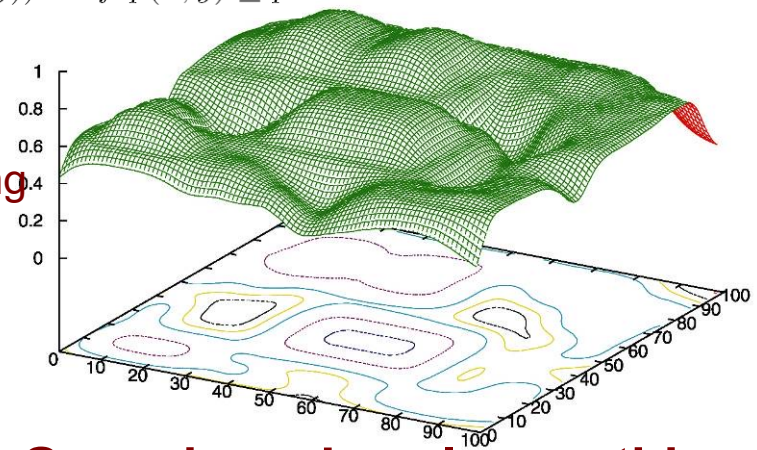
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$p'(x,y) = \begin{cases} \overline{p} + (p(x,y) - \overline{p})^{\delta} & if\ p(x,y) \geq \overline{p} \\ \overline{p} - (\overline{p} - p(x,y))^{\delta} & if\ p(x,y) \leq \overline{p} \end{cases}$$



Gaussian smoothing

Level smoothing

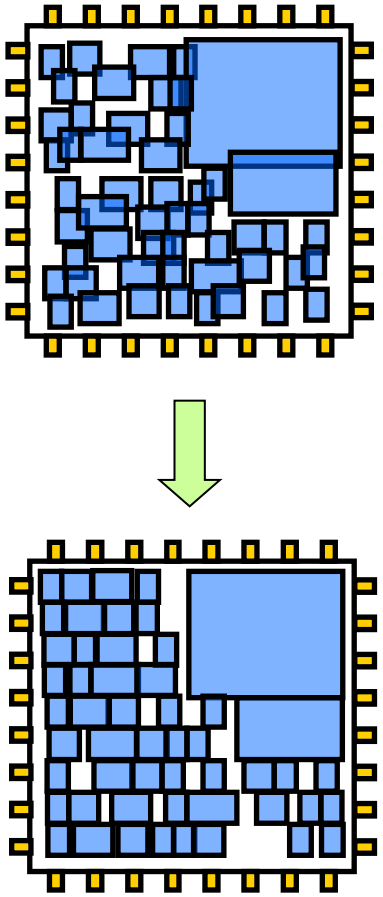


Gaussian + Level smoothing

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Legalization

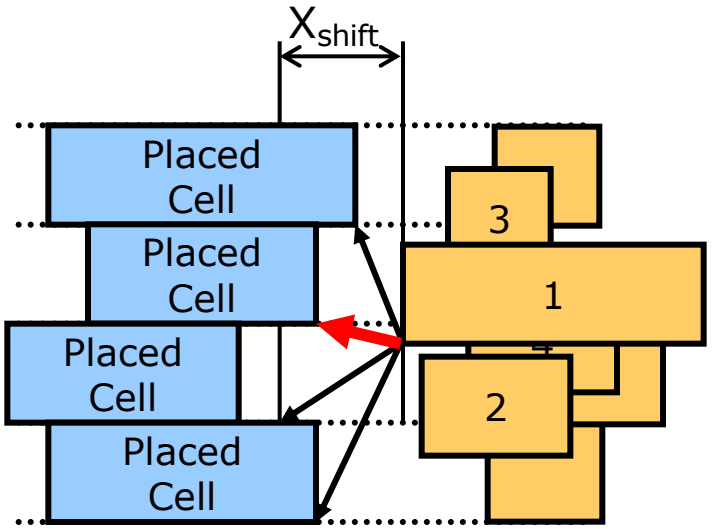
- Placement flow
  - Global placement
  - ***Legalization***
    - ***Mixed-size legalization***
    - ***Look-ahead legalization***
  - Detailed placement

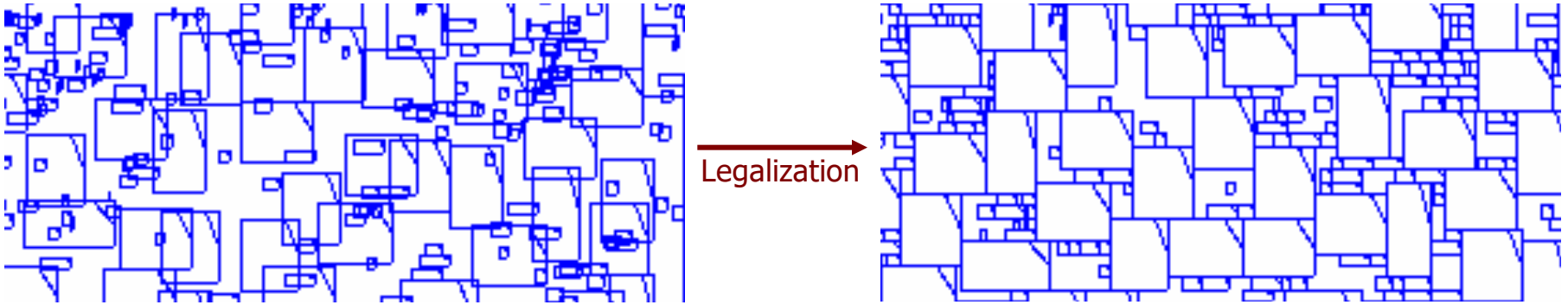


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Mixed-Size Legalization

- Determine the block legalization sequence by the x coordinate and the block size
  - Priority = $k_1 x_i + k_2 w_i + k_3 h_i$
    - x_i: the x coordinate of block i
    - $w_i(h_i)$: the width (height) of the block i
  - Large blocks are legalized earlier
- Select the position with the smallest wirelength within the x_{shift} range

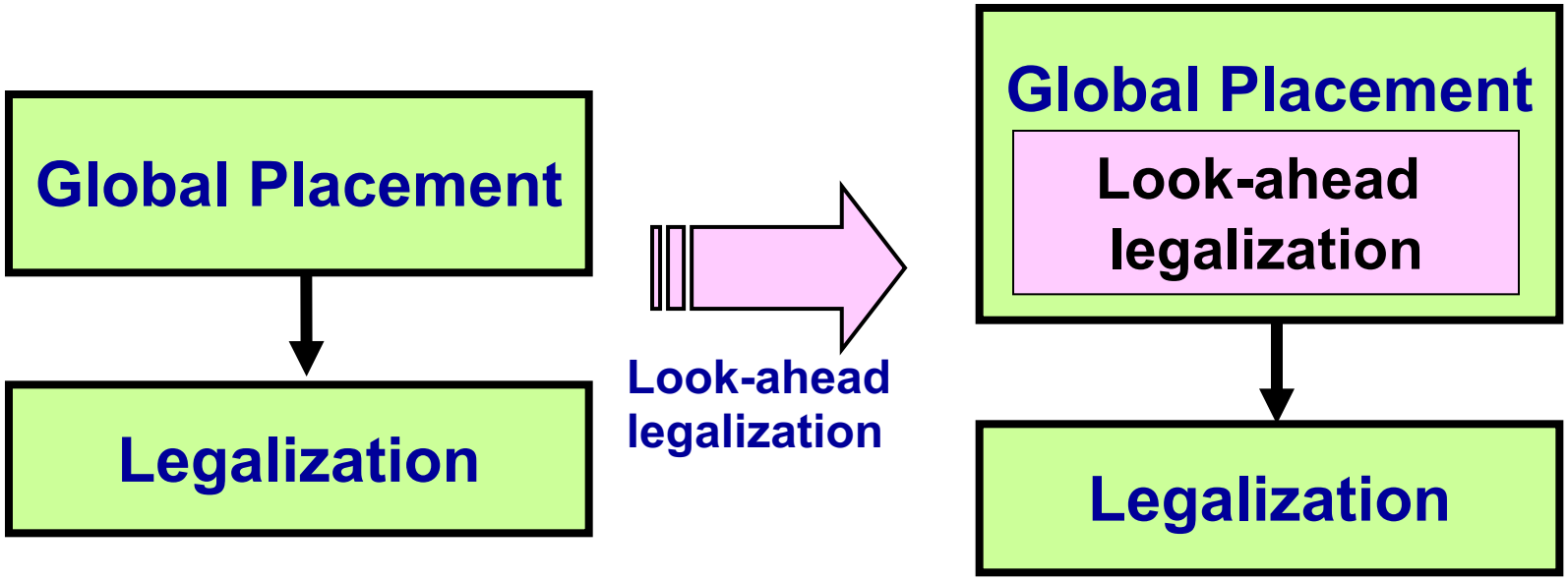




Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Look-ahead Legalization

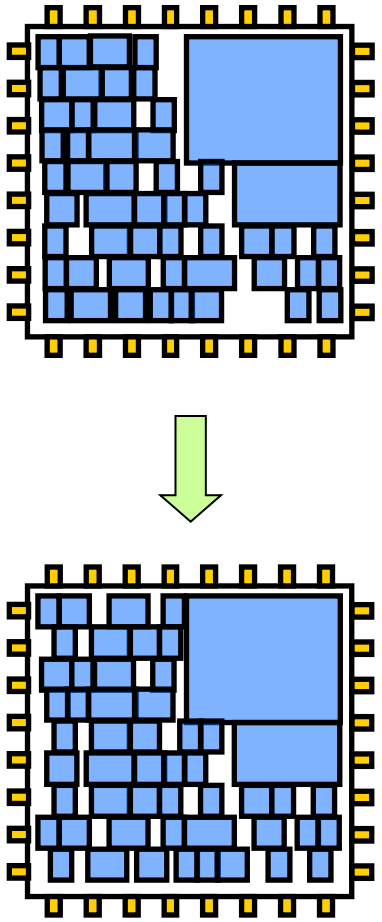
- Blocks not spread enough or spread too much lead to poor legalization results
- Incorporate look-ahead legalization into global placement
  - Perform legalization several times until blocks are spread enough
  - Reduce the gap between global placement and legalization



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Detailed Placement

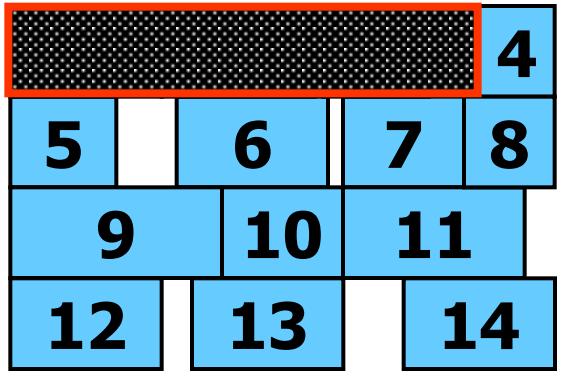
- Placement flow
  - Global placement
  - Legalization
  - ***Detailed placement***
    - ***Cell matching for wirelength minimization***
    - ***Cell sliding for density optimization***

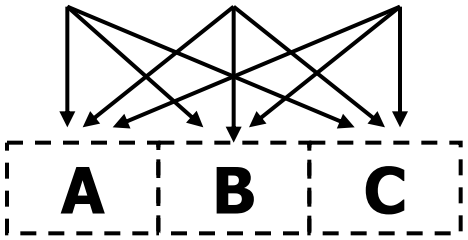


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cell Matching

- For wirelength minimization
- Steps
  1. Select a window
  2. Select blocks from the window
  3. Create a bipartite matching problem (edge weight = wirelength)
  4. Find the minimum weighted matching to optimize the wirelength
  5. Update block positions
- Handle **200-300** cells at one time
  - Compared to branch-and-bound which can handle only 6 cells at one time due to its high time complexity.



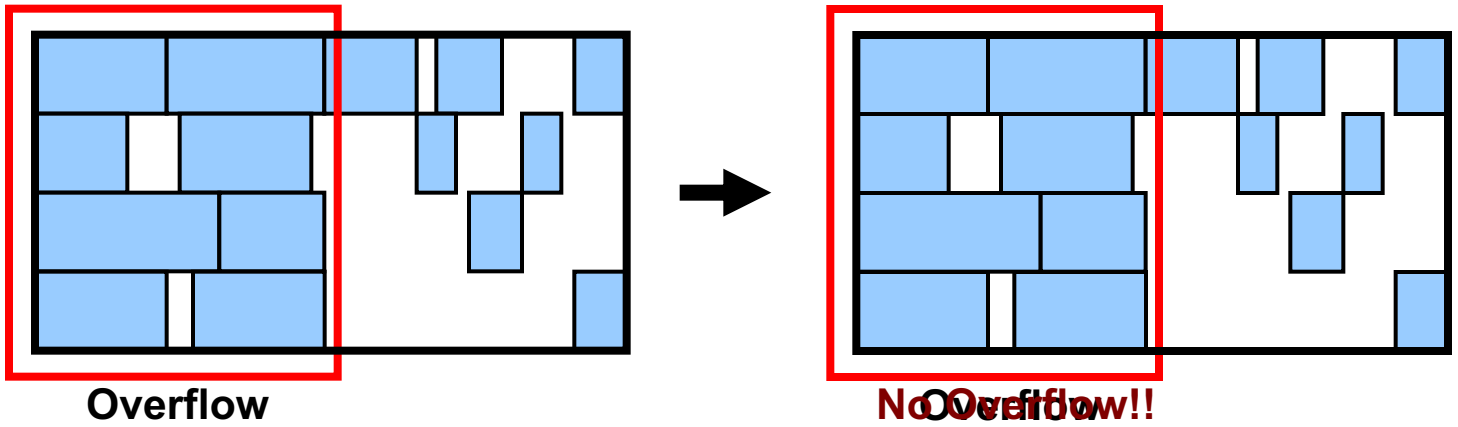


Assign cells {1,2,3} to locations {A,B,C}

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Cell Sliding

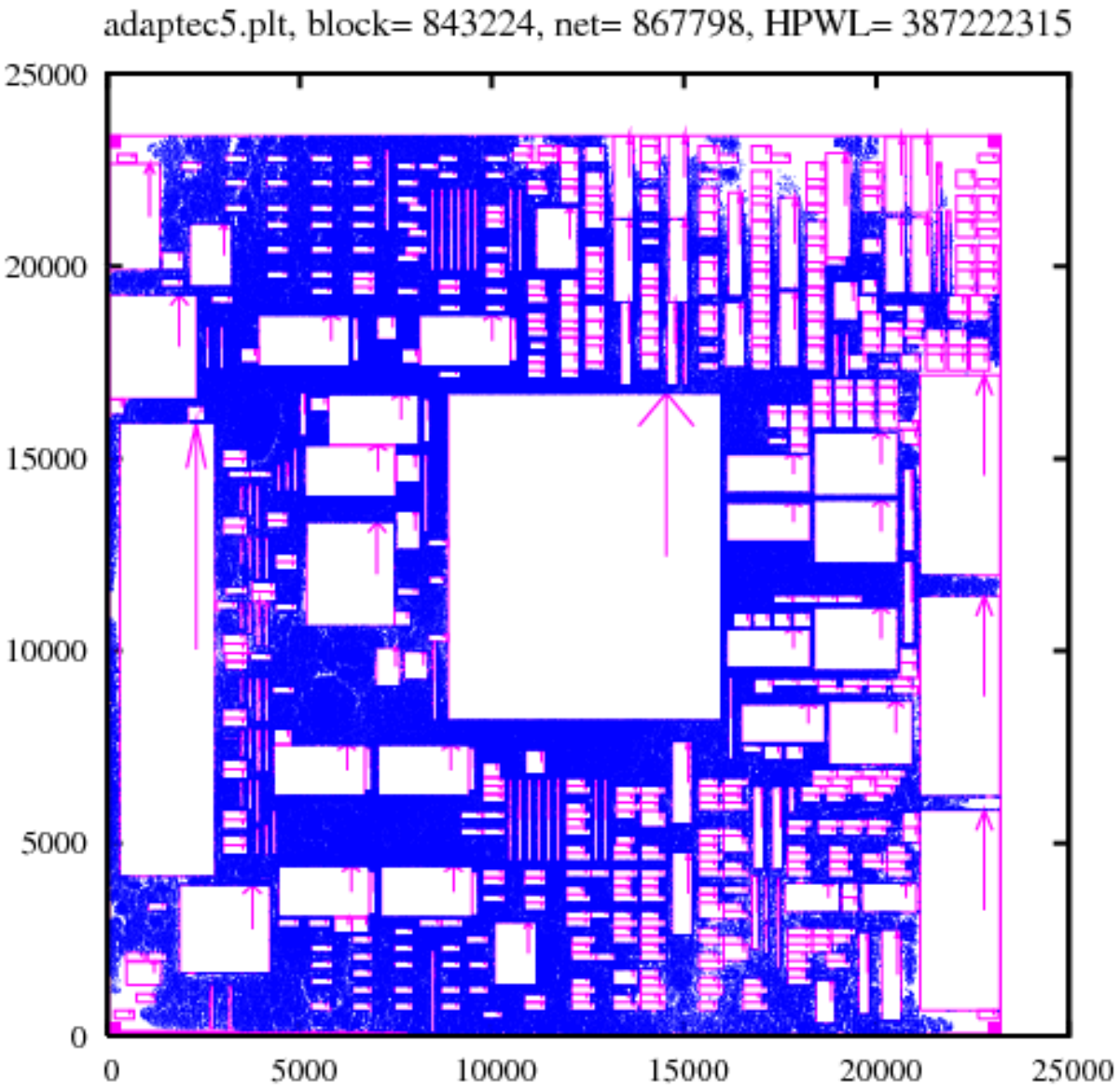
- For density optimization
- Steps
  1. Select an overflow window
  2. Calculate the amount of area to shift
  3. Move cells left/right to reduce the density
- Is fast and effective



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Demo (adaptec5)

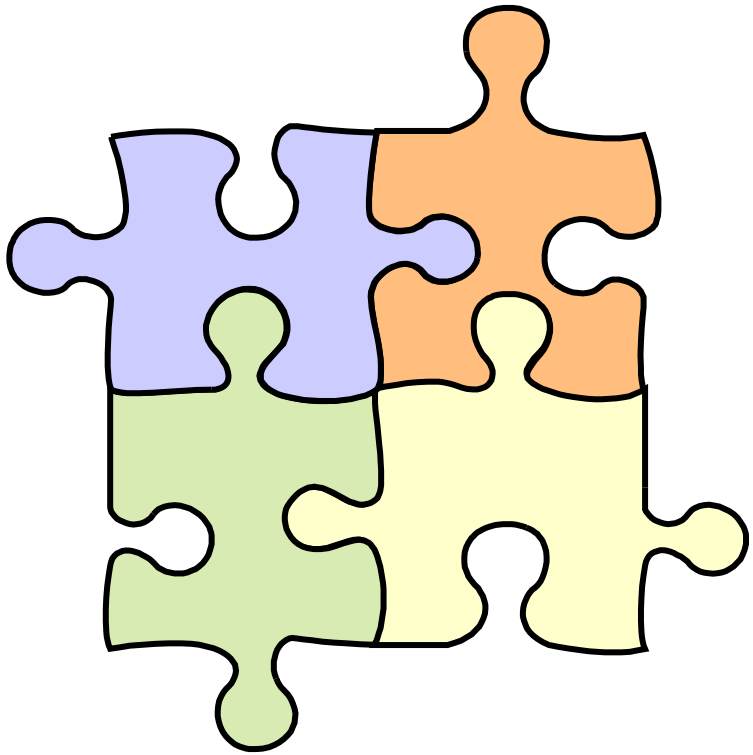
- GP: 5 levels
- #Movable obj. = 842K
- #Fixed obj. = 646
- #Nets = 868K



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

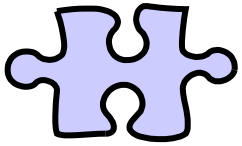
Essential Issues in Analytical Placement

- Chang, Jiang & Chen, "Essential issues in analytical placement algorithms," IPSJ Trans. System LSI Design Methodology, August 2009

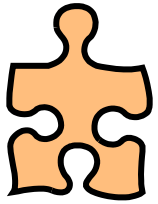


Slides: courtesy T.-C. Chen

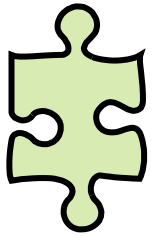
NTUplace3 Example



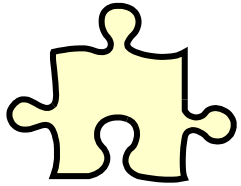
Wirelength Model: Log-Sum-Exp (LSE) function



Overlap Reduction: Bell-shaped density model



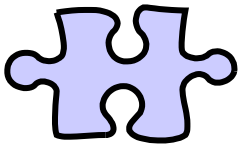
Integration: Penalty method



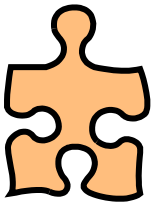
Optimization: Nonlinear

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

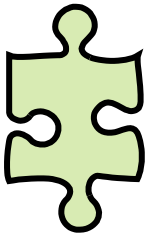
GODIAN Example



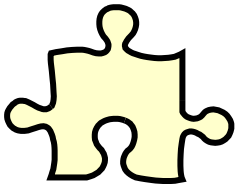
Wirelength Model: Quadratic function



Overlap Reduction: Partitioning



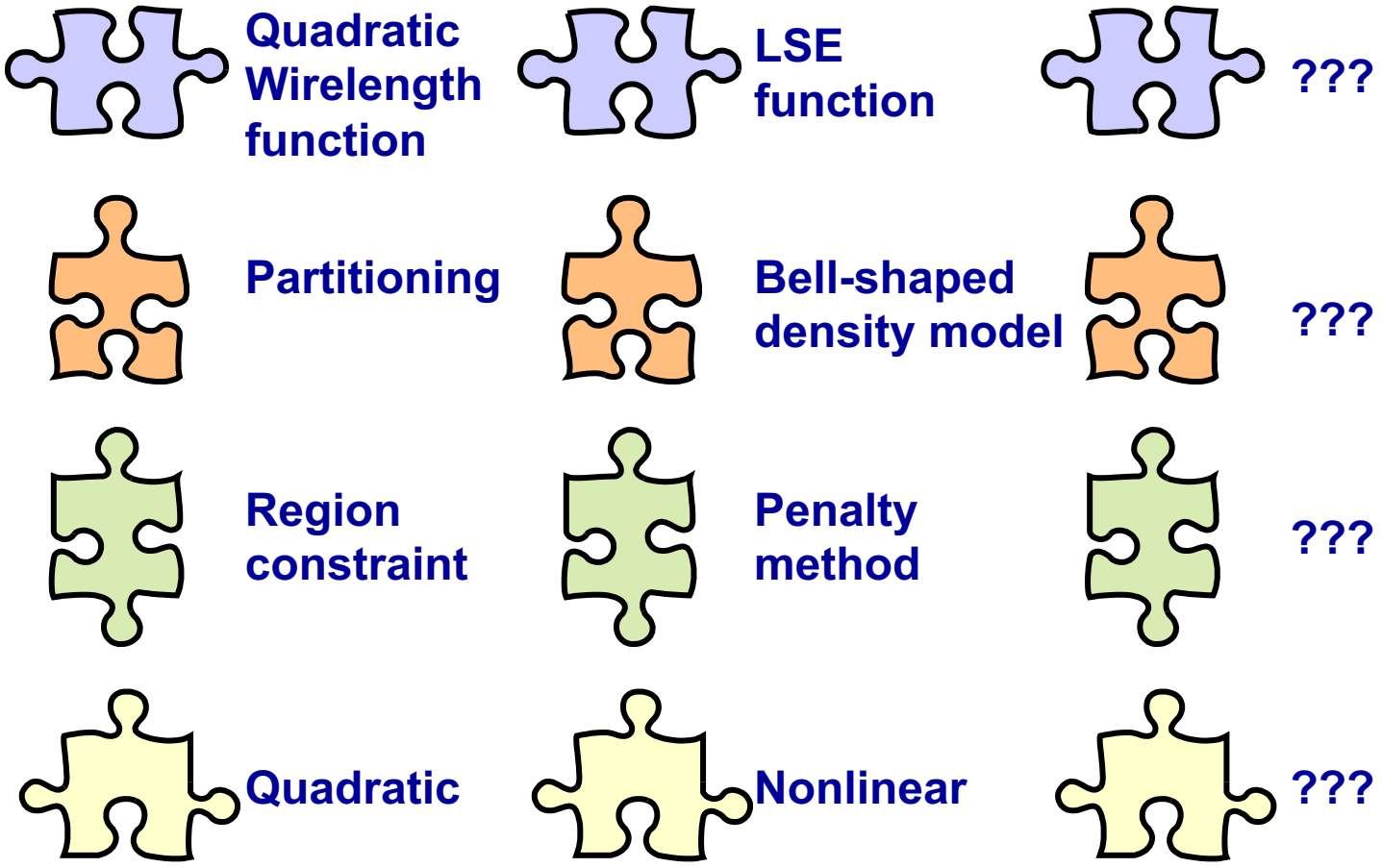
Integration: Region Constraint



Optimization: Quadratic

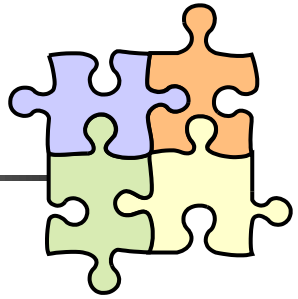
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Other Combinations?



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Modern Academic Analytical Placers

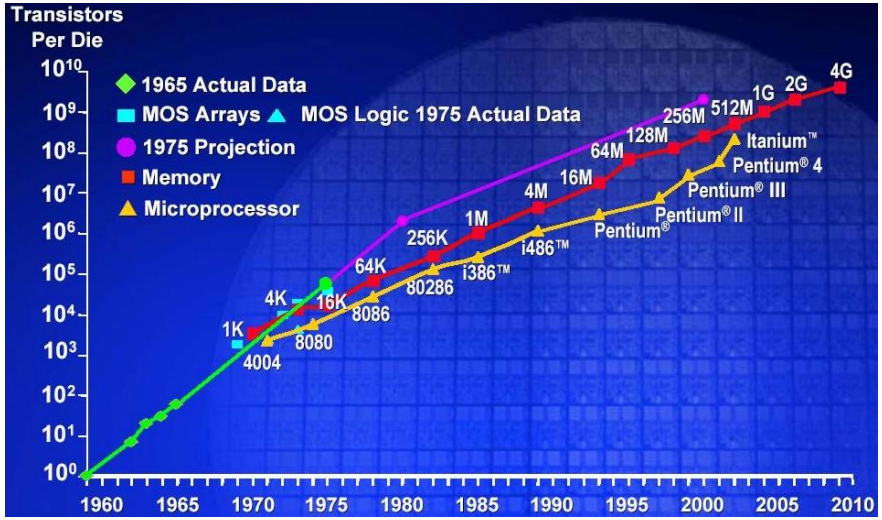


Placer	Wirelength Model	Overlap Reduction	Integration	Optimization
APlace	LSE	Density	Penalty Method	Nonlinear
BonnPlace	Quadratic	Partitioning	Region Constraint	Quadratic
DPlace	Quadratic	Diffusion	Fixed Point	Quadratic
FastPlace	Quadratic	Cell Shifting	Fixed Point	Quadratic
FDP	Quadratic	Density	Fixed Point	Quadratic
Gordian	Quadratic	Partitioning	Region Constraint	Quadratic
hATP	Quadratic	Partitioning	Region Constraint	Quadratic
Kraftwerk2	Bound2Bound	Density	Fixed Point	Quadratic
mFAR	Quadratic	Density	Fixed Point	Quadratic
mPL6	LSE	Density	Penalty Method	Nonlinear
NTUplace3	LSE	Density	Penalty Method	Nonlinear
RQL	Quadratic	Cell Shifting	Fixed Point	Quadratic
Vassu	LSE	Assignment	Fixed Point	Nonlinear

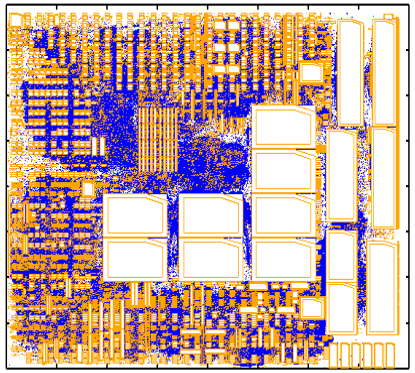
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

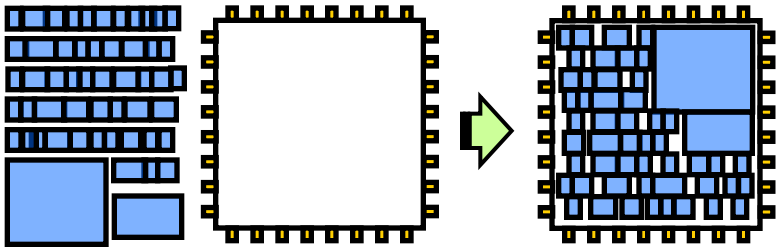
Modern Placement Challenges Revisited

- High complexity
  - Millions of objects to be placed
- Mixed-size placement
  - Hundreds of movable macro blocks with millions of small standard cells
- Placement constraints
  - Preplaced blocks
  - Chip density
  - etc.



2.5M placeable objects





Courtesy of Prof. Y.-W. Chang & H.-M. Chen

"Hard Macros Will Revolutionize SoC Design"

- Wein & Benkoski, *EE Times*, Aug. 20, 2004
- An SoC often contains hundreds of pre-designed macros
  - Embedded memories, analog circuits, IP blocks, etc.
- The number of macros grows dramatically.
- Existing layout tools are having problems with macros

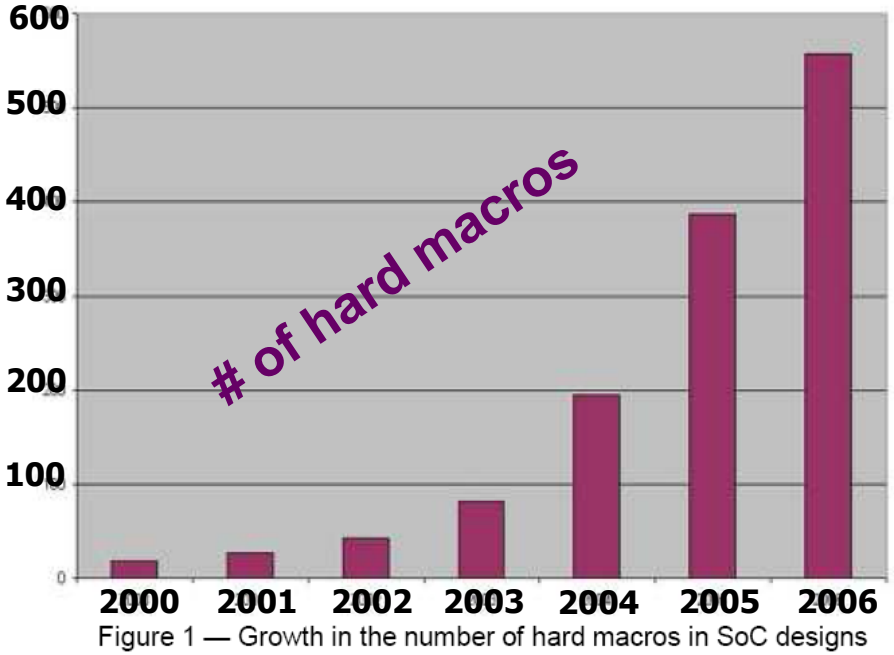


Figure 1 — Growth in the number of hard macros in SoC designs

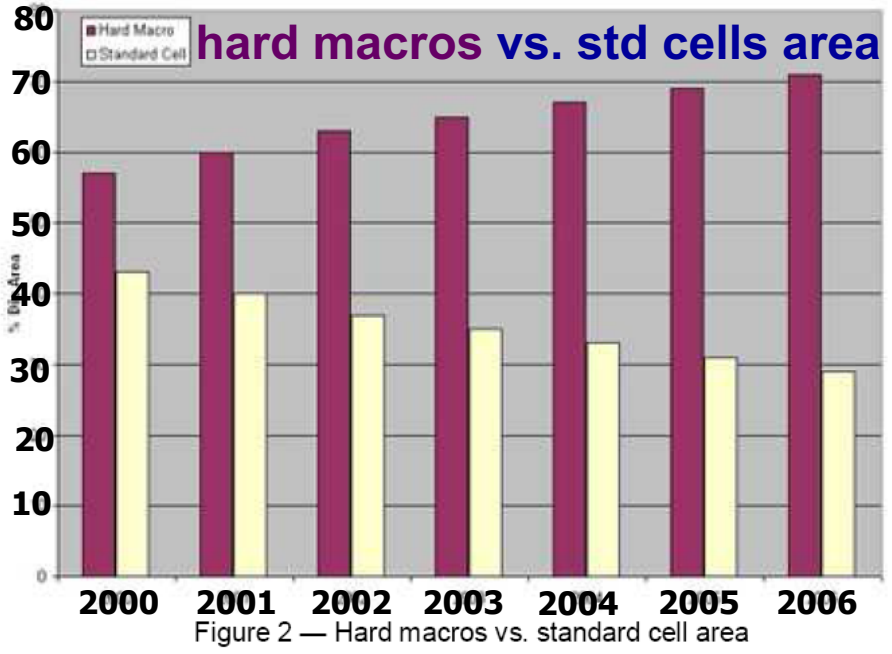


Figure 2 — Hard macros vs. standard cell area

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

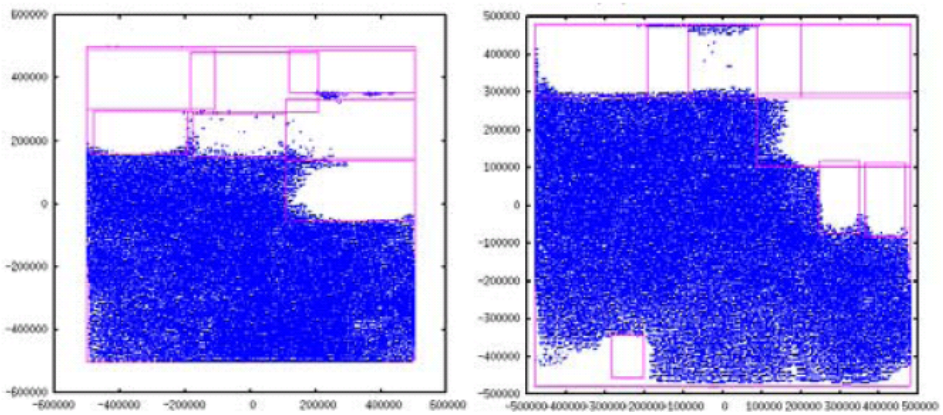
Existing Works on Mixed-Size Designs

- Type 1: Simultaneous macro and cell placement
  - Allow macro overlaps during the placement process
  - Need a robust legalizer
  - Include most mixed-size placers
    - APlace, Dragon, FengShui, Kraftwerk, mPL, NTUplace3, etc.
- Type 2: Constructive macro placement
  - Keep macro overlap-free during the placement process
  - Include Capo, PATOMA, FLOP
  - It is harder to achieve a high-quality solution
- Type 3: Two-stage macro placement
  - Consist of (1) macro placement and then (2) cell placement
  - Is robust in finding legal placement
  - Is widely used in the industry
  - Include MP-tree, CG
  - **Need a good macro placer!!**

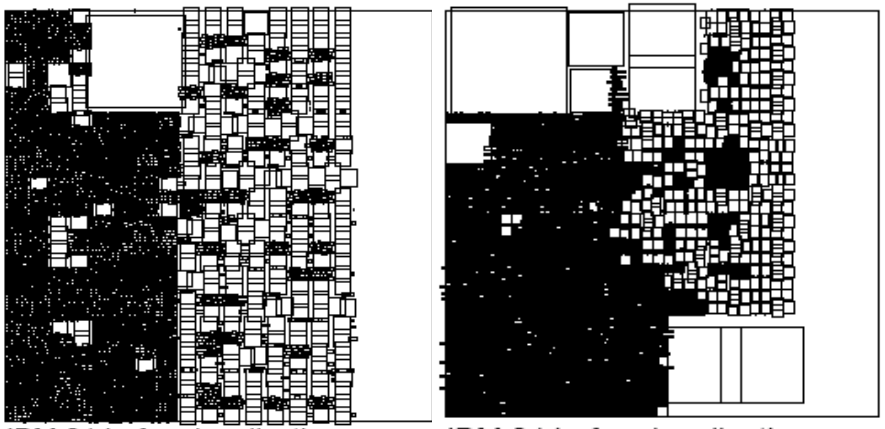
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Problem with Simultaneous Macro and Cell Placement

- Is hard to guarantee overlap-free placement
- Placement generated by APlace 2.0



- Placement generated by FengShui

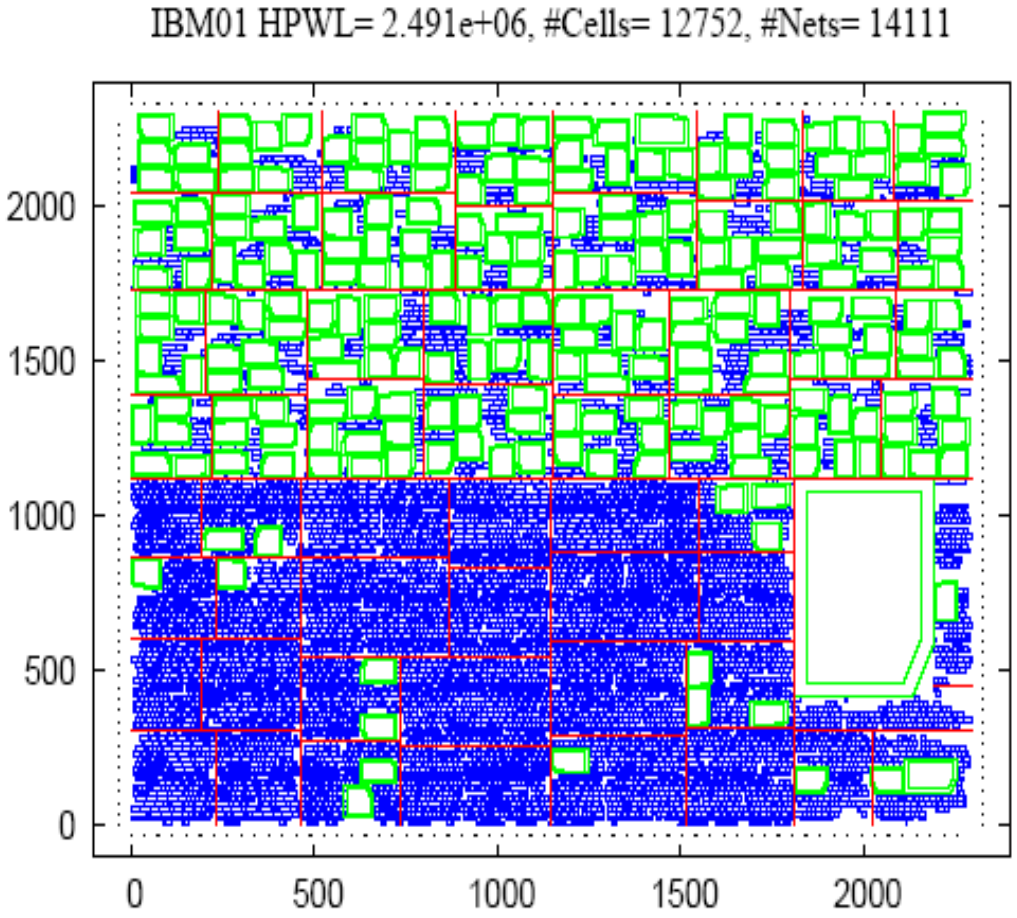


IBM 01 before legalization IBM 04 before legalization

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Constructive Macro Placement

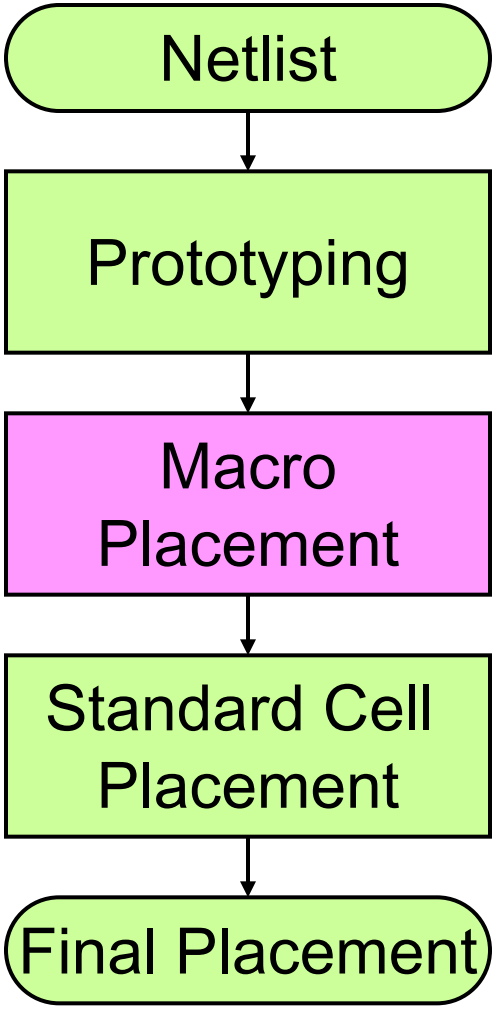
- Capo: recursive min-cut bisection
- Floorplanning is performed when necessary in the sub-region.

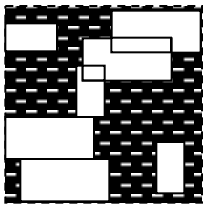


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

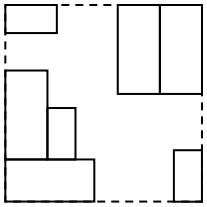
Our Mixed-Size Placement Flow

- Based on two-stage macro placement

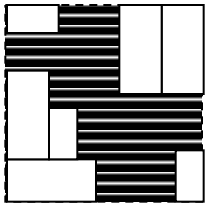




wirelength optimization



macro legalization (displacement minimization, orientation optimization, congestion optimization, etc.)

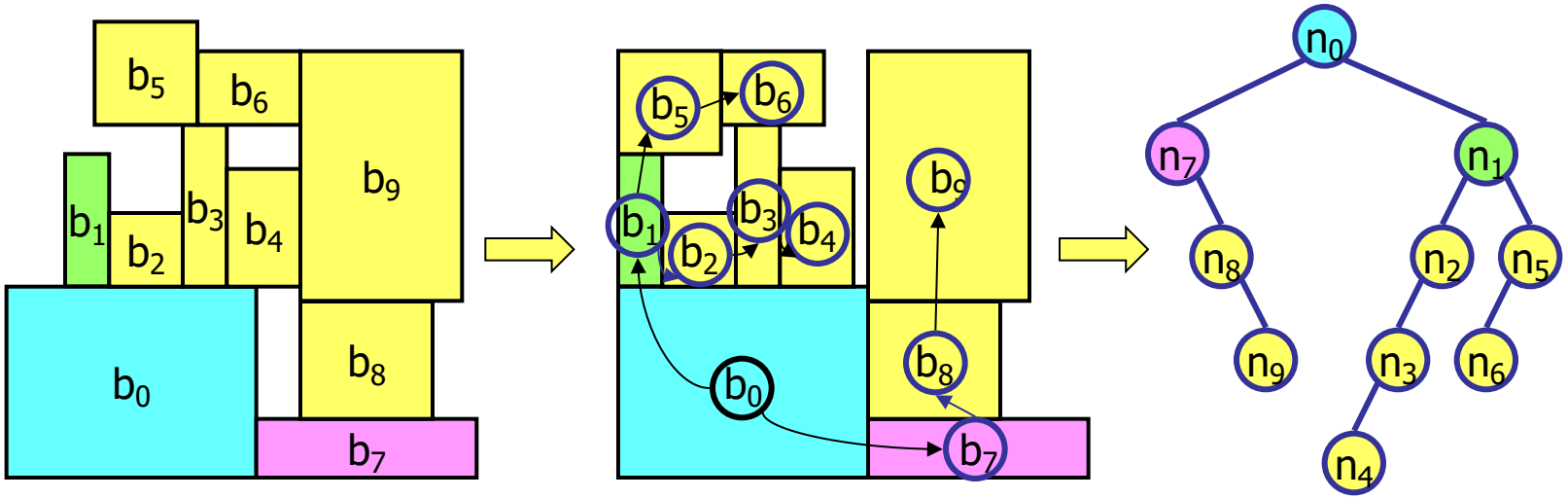


wirelength optimization, congestion optimization

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Macro Placement Using Multiple B*-Trees

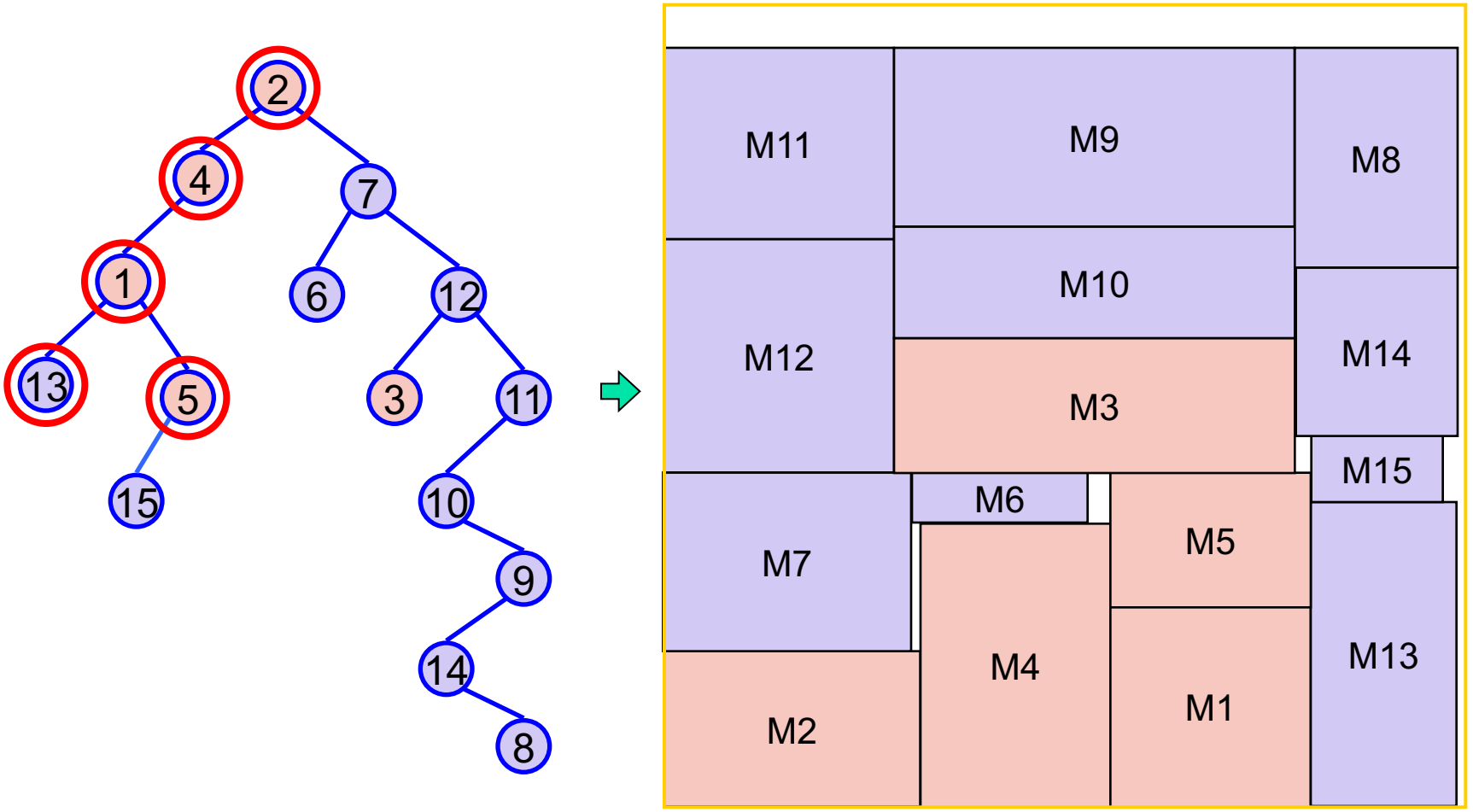
1. Compact macros to left and bottom.
2. Construct an ordered binary tree.
   - Left child: the lowest, adjacent macro on the right ($x_j = x_i + w_i$).
   - Right child: the first macro above, with the same x-coordinate ($x_j = x_i$).



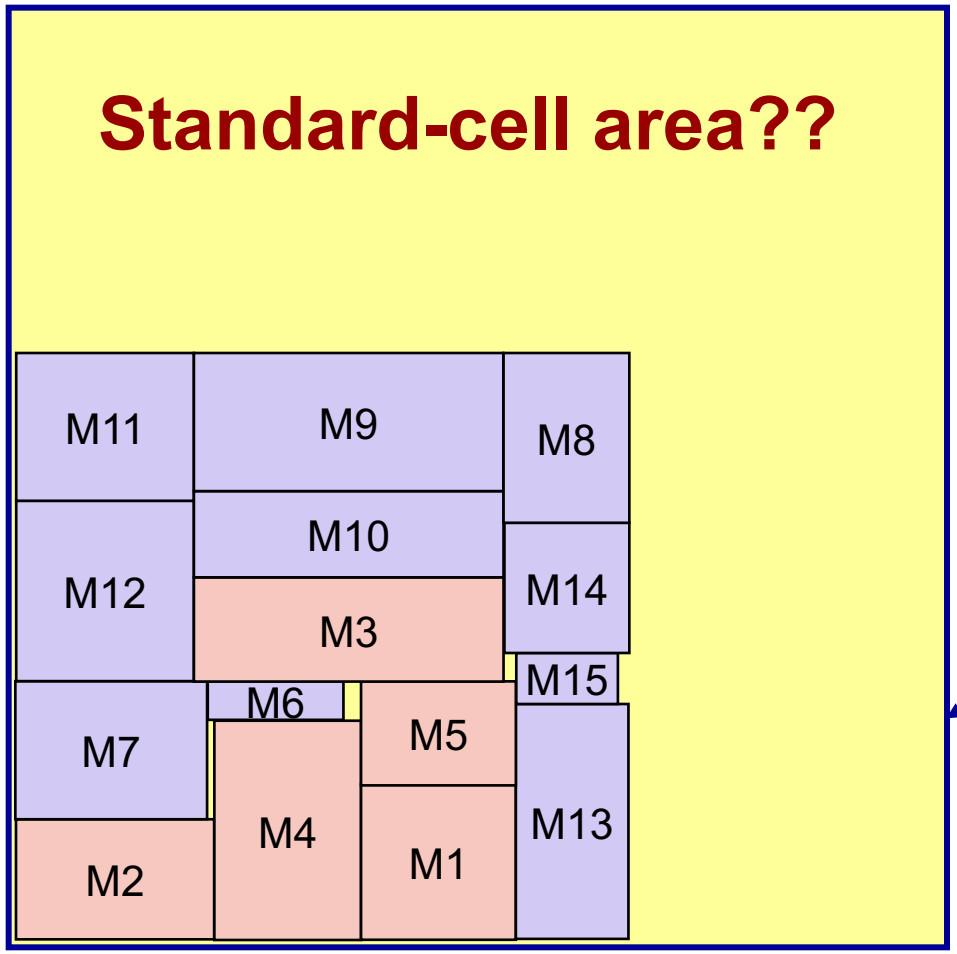
A non-slicing floorplan Compact to left and down B*-tree

B*-Tree Packing Revisited

- Preorder traversal, complexity: O(n)



But What If %Macro Area Is Not High?

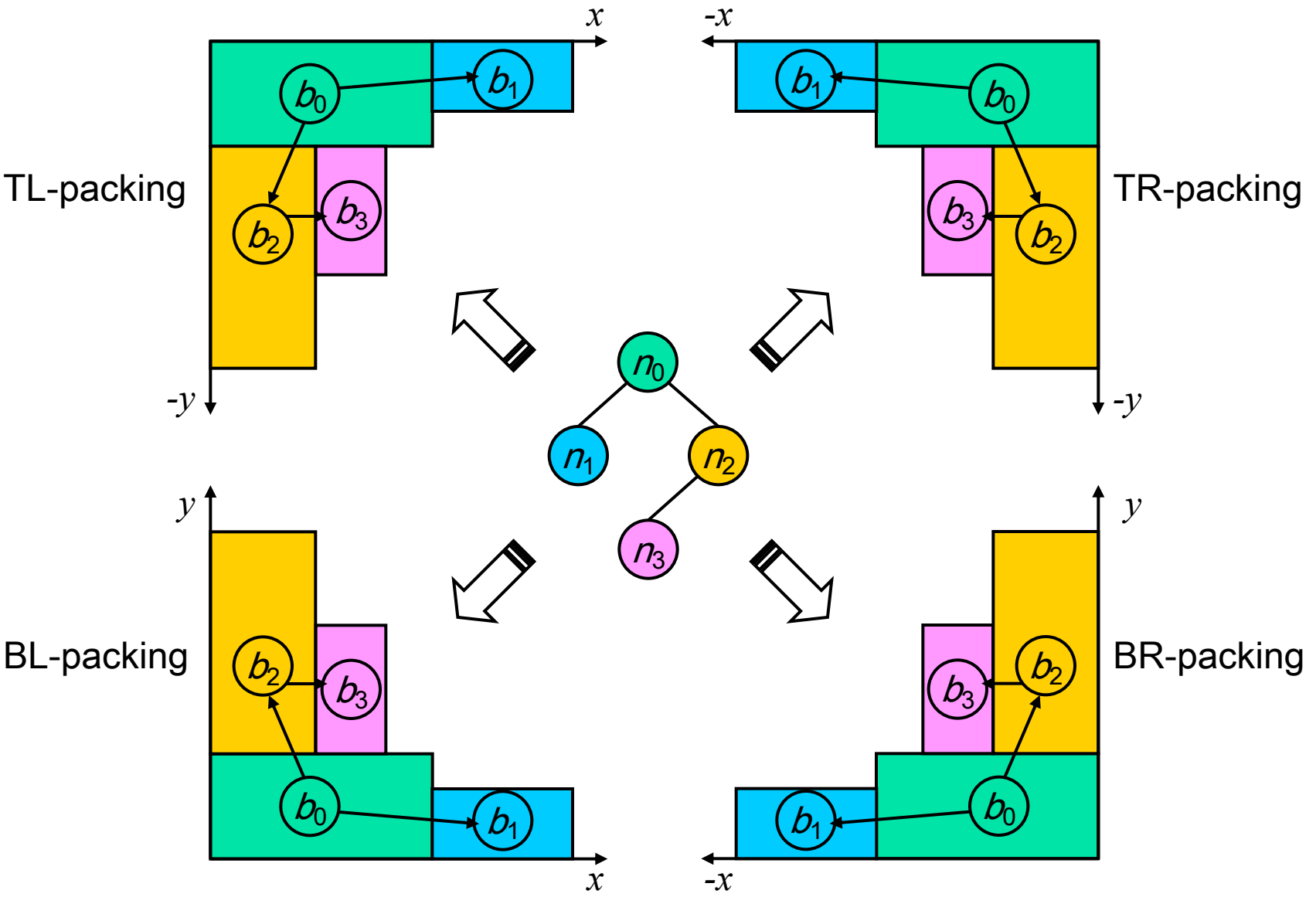


All macros will be packed together!!

Chip outline

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

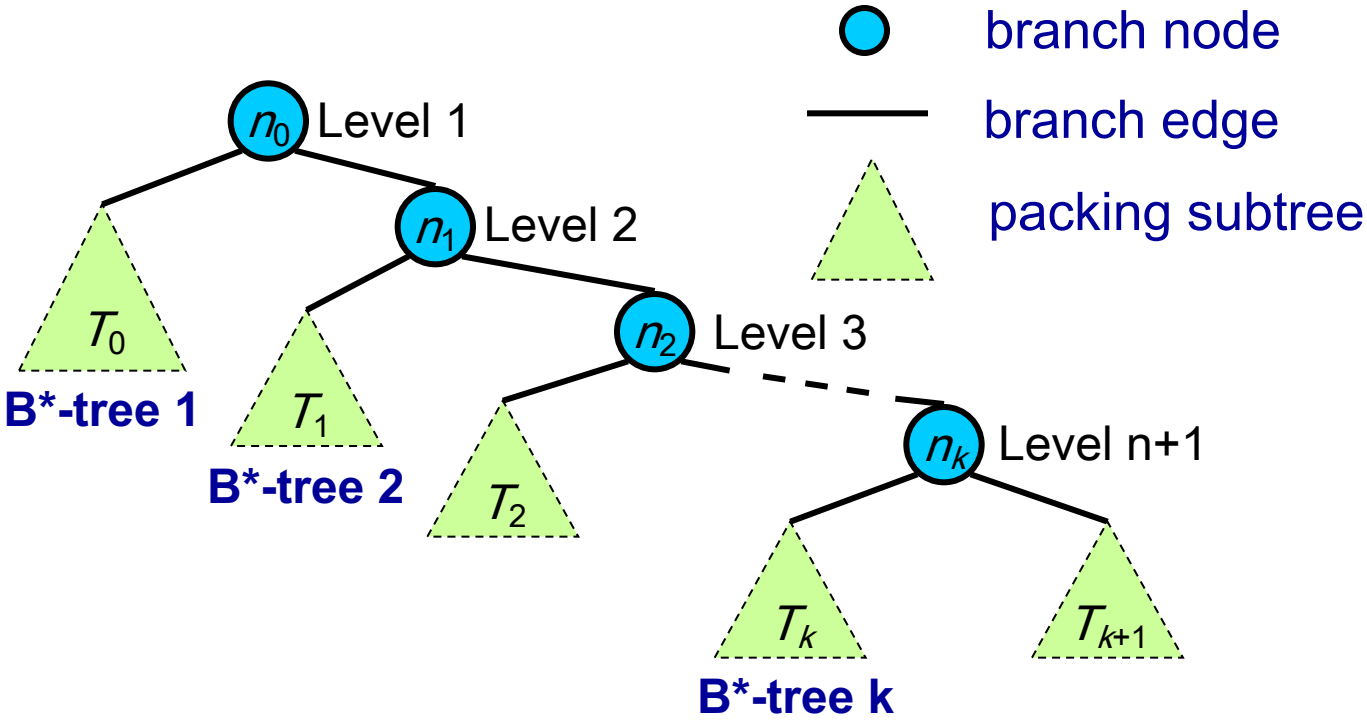
Multi-Packing (MP) Tree Representation



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Generalized MP-Tree Representation

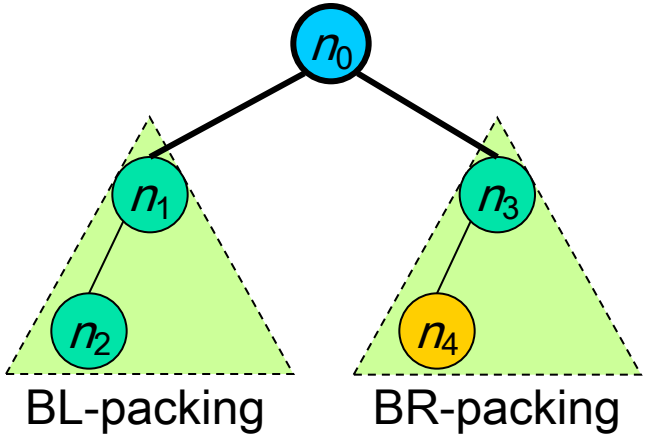
- Working on independent packing trees may not obtain desired solution
  - Lack global interactions among different subproblems
- Key: Combine several packing trees packing to different corners
- Use the right skewed branch for easier implementation

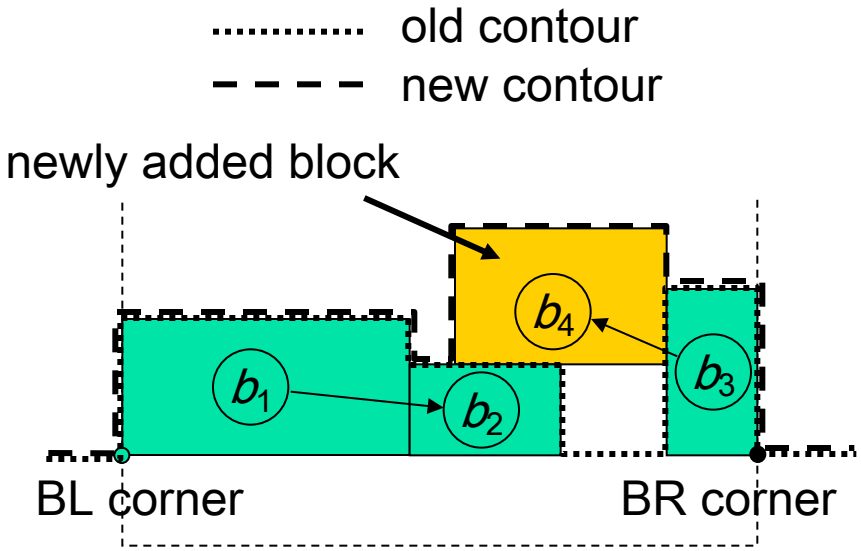


Courtesy of Prof. Y.-W. Chang & H.-M. Chen

MP-tree Packing

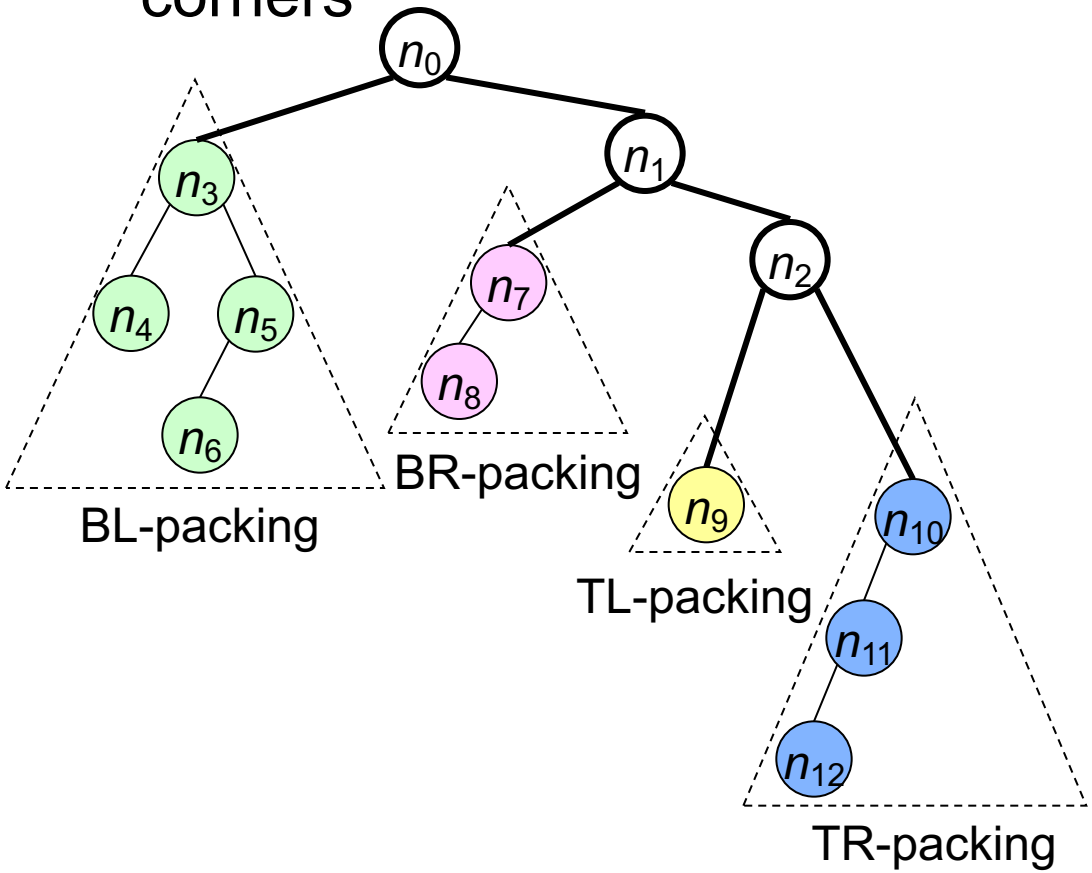
- All BL- and BR-packing subtrees use the bottom-contour data structure
- All TL- and TR-packing subtrees use the top contour data structure
- An example of BL- and BR-packing

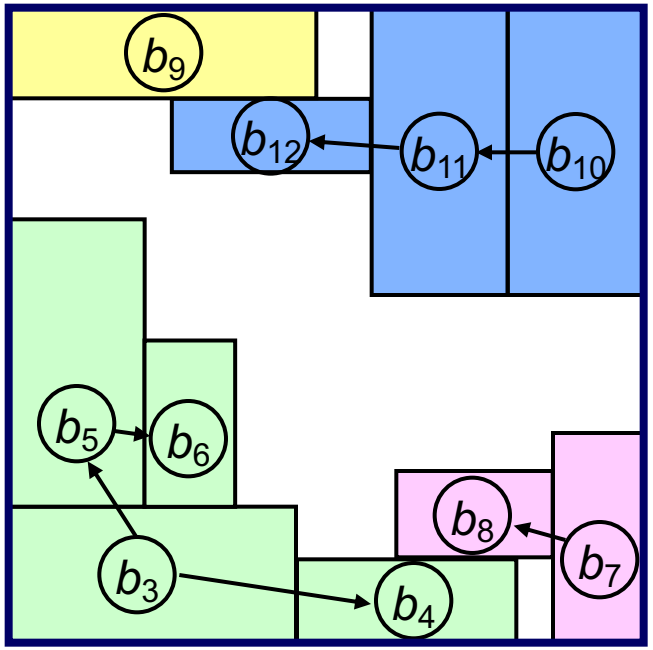




MP-tree Example

- Use four packing subtrees to handle a rectangular chip
- Applies to a floorplan region with an arbitrary number of corners





Courtesy of Prof. Y.-W. Chang & H.-M. Chen

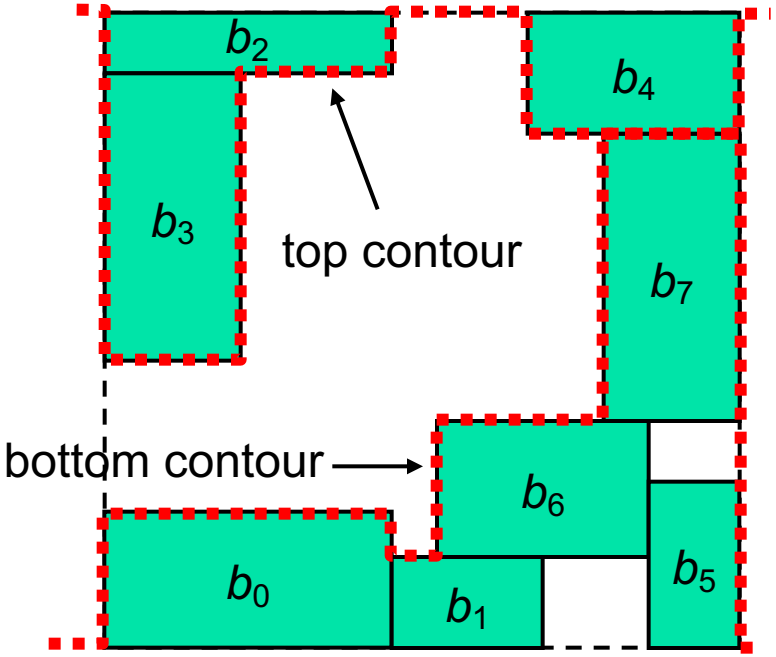
Operations on MP-tree

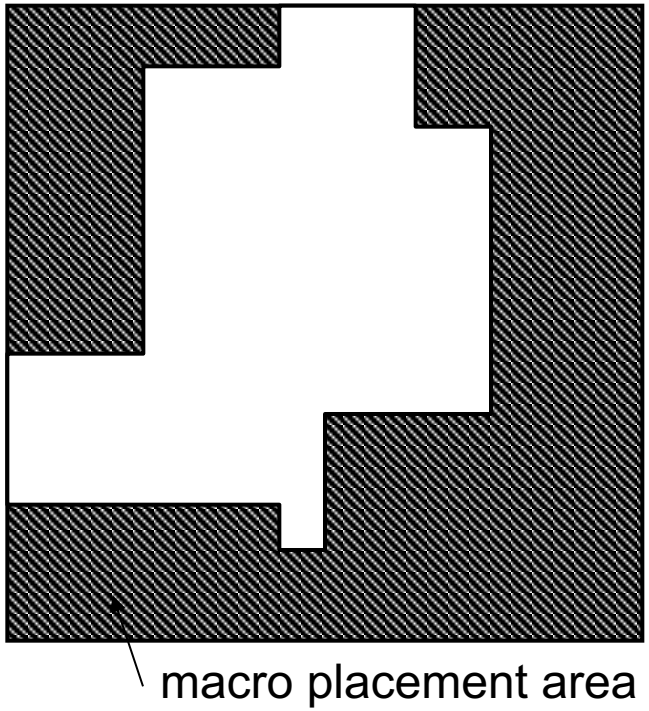
- Operations for simulated annealing to perturb one MP-tree to another

- Op1: Rotate a block or a cluster
- Op2: Resize a cluster
- Op3: Move a node in a packing subtree to another place
- Op4: Swap two nodes within one or two packing subtrees
- Op5: Swap two packing subtrees

Evaluation of a Macro Placement

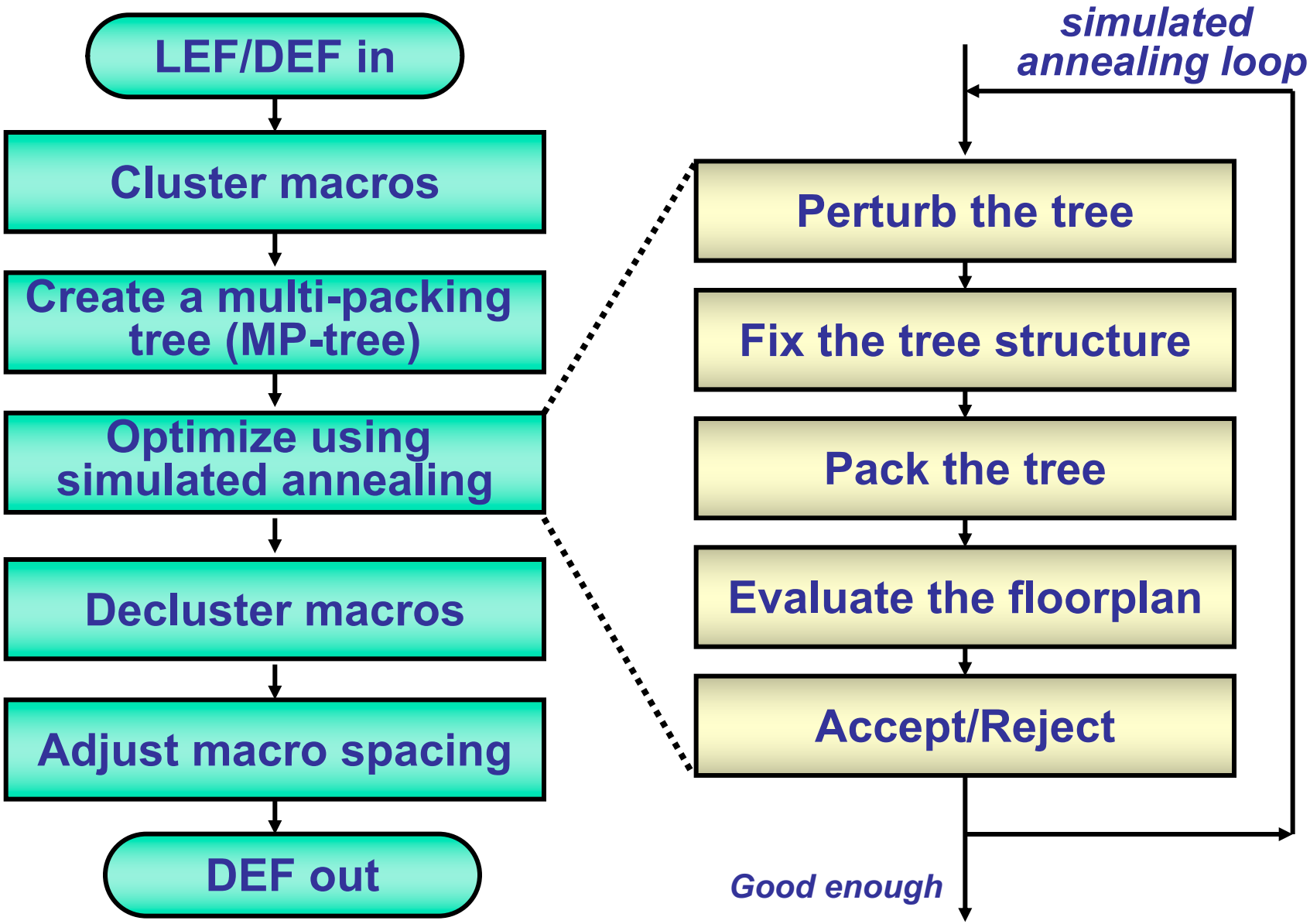
- Macro placement area
- Wirelength
- Macro displacement





Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Macro Placement Flow



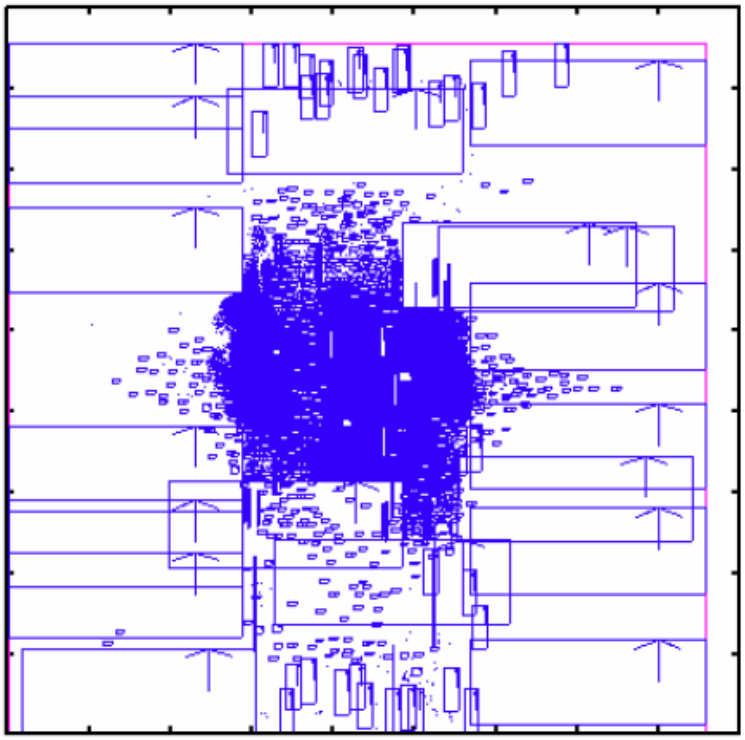
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

MP-tree Macro Placer

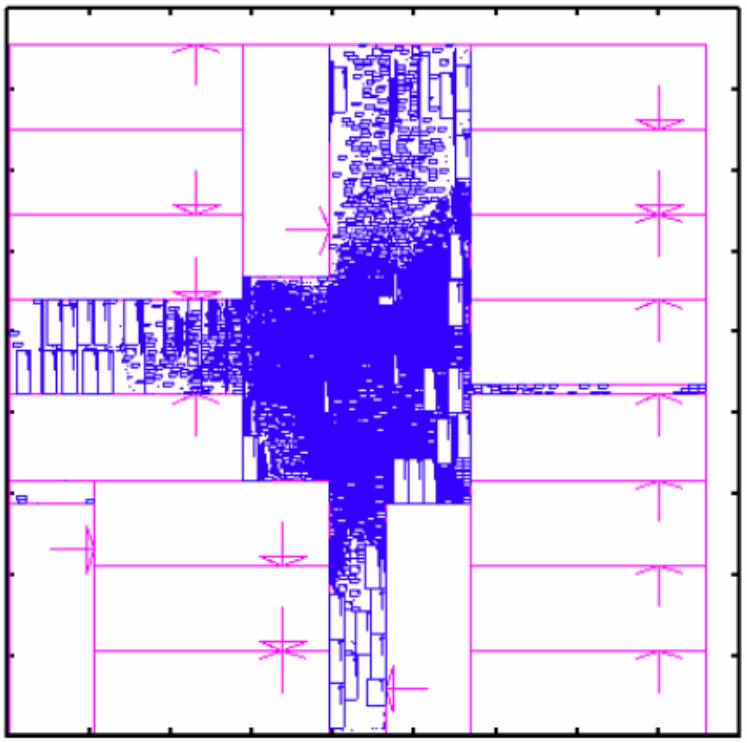
- Is very **fast** for operations and packing based on binary trees (extension of B*-trees)
- Considers **macro orientations** and **spacing** between macros
  - Lead to smaller wirelength and less congestion
- Can handle various **placement constraints**
  - Pre-placed blocks, corner block, and placement blockages
- Can be combined with state-of-the-art standard cell placers to obtain better mixed-size placement

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

ISPD-06 newblue3 Layouts



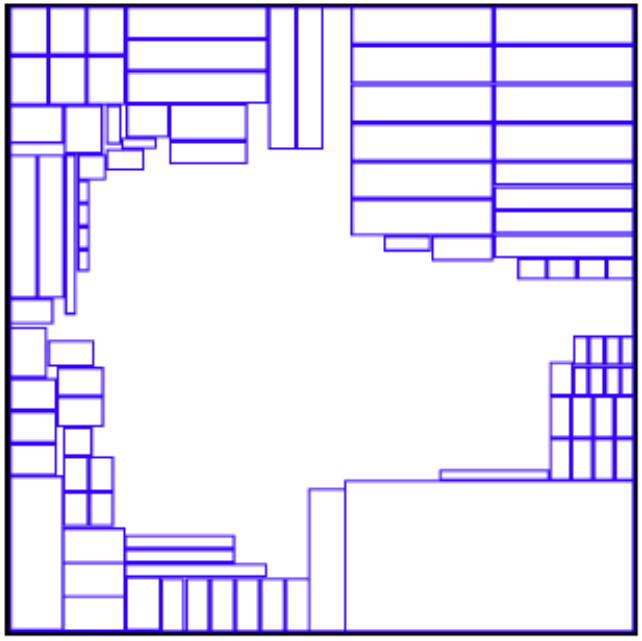
NTUplace3 alone
(failed to find a
legal placement)



MP-tree + NTUplace3

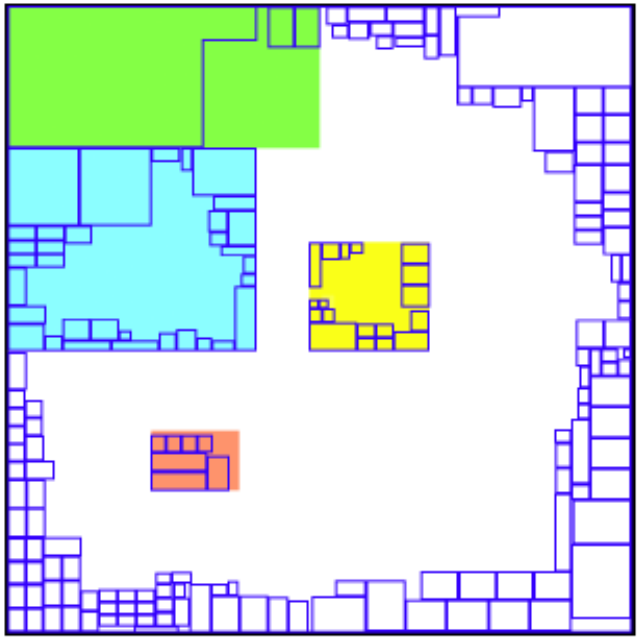
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Mchip Benchmark Results



(a) mchip2

95 Macros



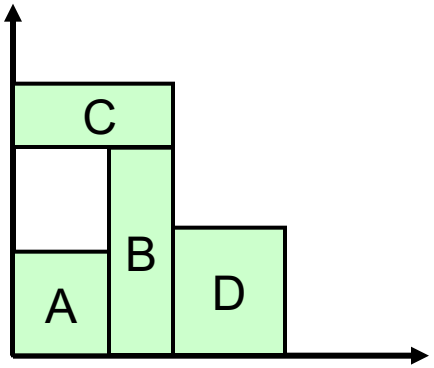
(b) mchip4

380 Macros w/
4 region constraints

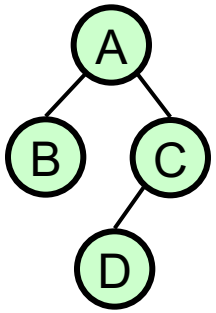
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Limitation of the MP-tree

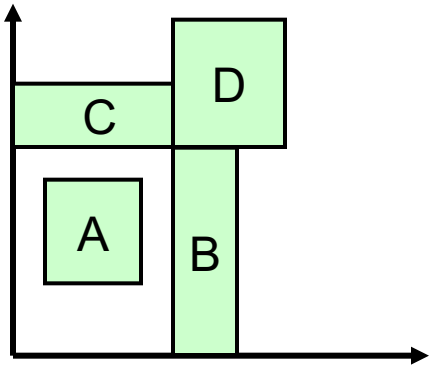
- MP-tree can generate only compacted macro placement results, not desirable for low-density designs



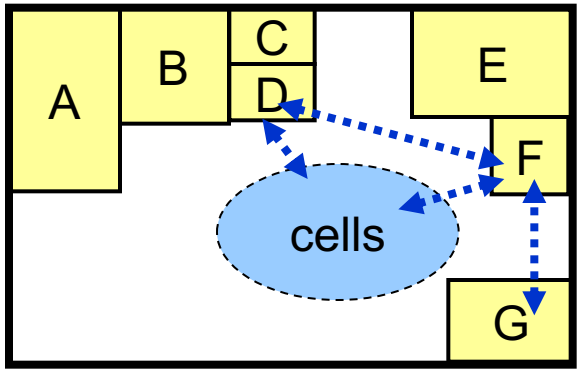
compacted floorplan



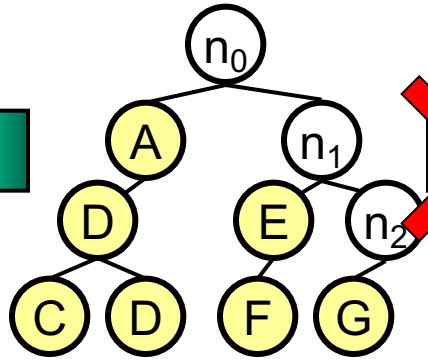
B*-tree



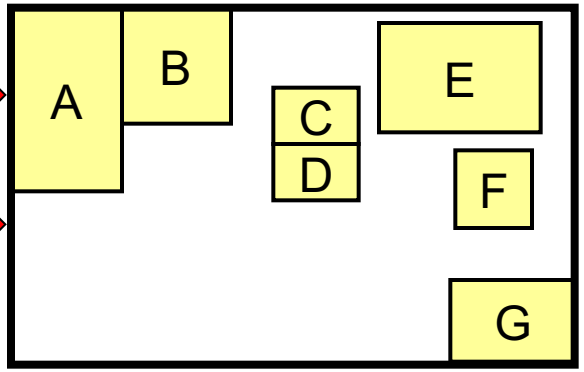
non-compacted floorplan



compacted placement



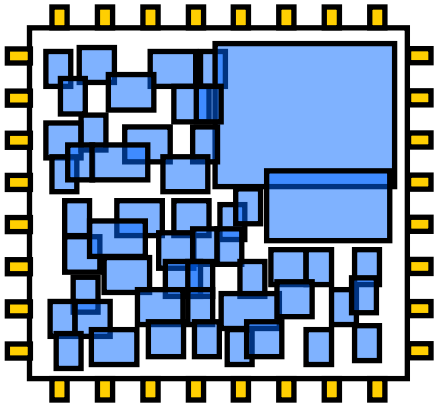
MP-tree



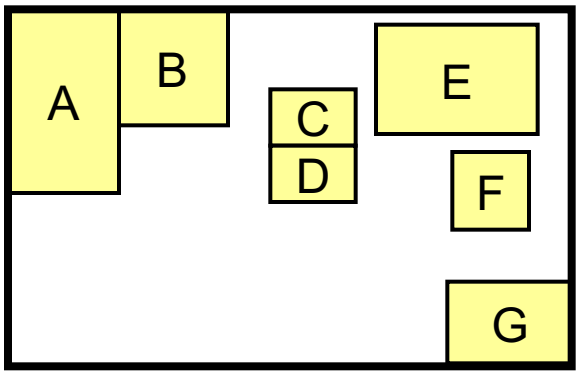
non-compacted placement

Techniques to Overcome the Limitations

. Chen, Chuang, Chang, Chang, ICCAD-08.



Legal placement?



Non-compacted macro placement?

Floorplan techniques

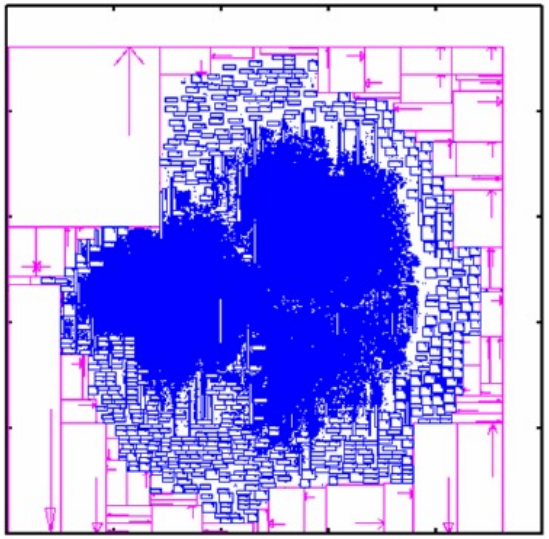
Placement prototype

Constraint Graph (e.g., TCG) + Linear Programming

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

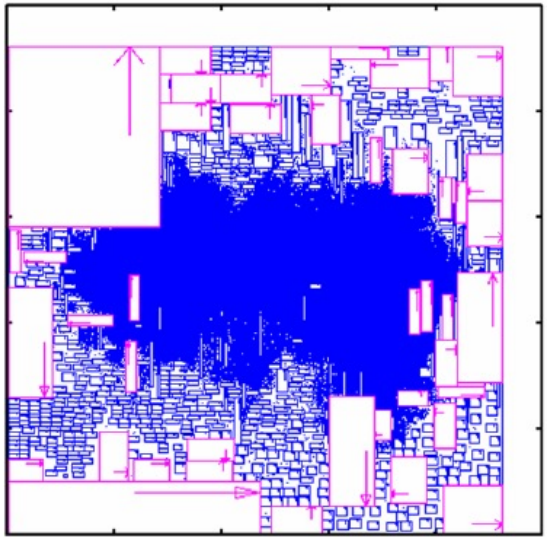
Layout Example: adaptec5

- Integrated with NTUplace3
- 80% utilization rate



MP-tree

HPWL: 32.72×10^7

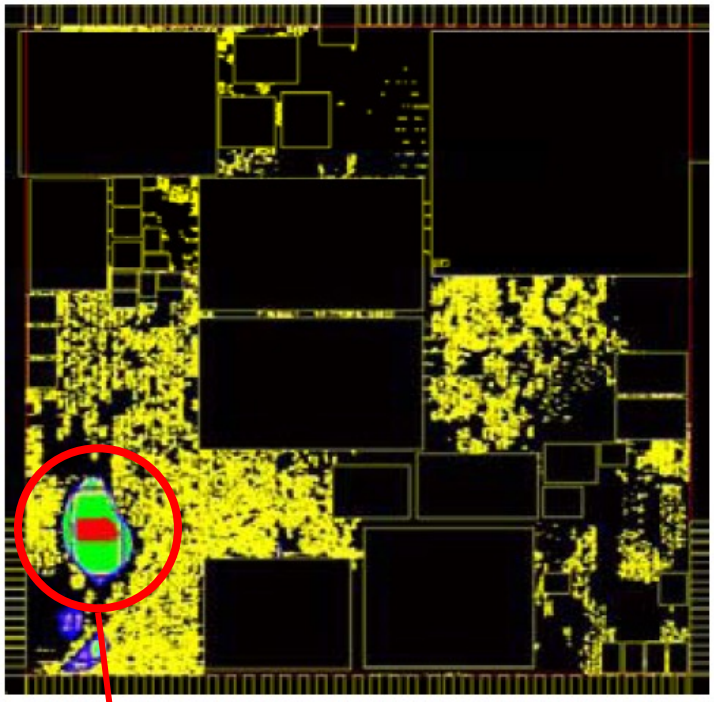


TCG

HPWL: 29.72×10^7

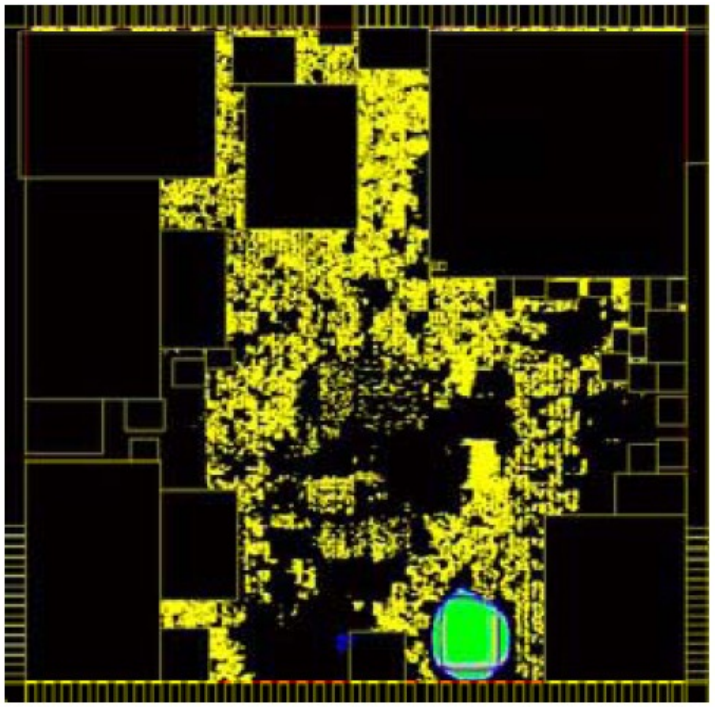
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Congestion Map - gchip



(a) From Tool A

congestion hot spot



(b) From our macro placer

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Summary: Placement

- Popular methods: partitioning-based, simulated annealing, analytical placement
- Major steps: global placement -> legalization -> detailed placement
- Design Styles
  - Gate Array / FPGA
  - Standard cell
  - Macro block
  - Mixed-size (block and cells)
- Integrated with the design flow
  - Placement with clock-tree synthesis, power/ground network synthesis, scan-chain reordering, etc
- Many modern design challenges, e.g.,
  - Mixed-size large-scale placement
  - Other objectives: routability, timing, power (voltage islands), thermal, manufacturability, reliability
  - 3D IC placement (thermal vias?)

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Popular Wirelength Models

HPWL $$W(\mathbf{x},\mathbf{y}) = \sum_{\text{net } e} \left(\max_{v_i, v_j \in e} |x_i - x_j| + \max_{v_i, v_j \in e} |y_i - y_j| \right)$$

quadratic $$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_{ij} [(x_i - x_j)^2 + (y_i - y_j)^2]$$

Log-sum-exp $$\gamma \sum_{e \in E} (\log \sum_{v_k \in e} \exp(x_k/\gamma) + \log \sum_{v_k \in e} \exp(-x_k/\gamma) + \log \sum_{v_k \in e} \exp(y_k/\gamma) + \log \sum_{v_k \in e} \exp(-y_k/\gamma))$$

L_p-norm $$\sum_{e \in E} ((\sum_{v_k \in e} x_k^p)^{\frac{1}{p}} - (\sum_{v_k \in e} x_k^{-p})^{-\frac{1}{p}} + (\sum_{v_k \in e} y_k^p)^{\frac{1}{p}} - (\sum_{v_k \in e} y_k^{-p})^{-\frac{1}{p}})$$

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

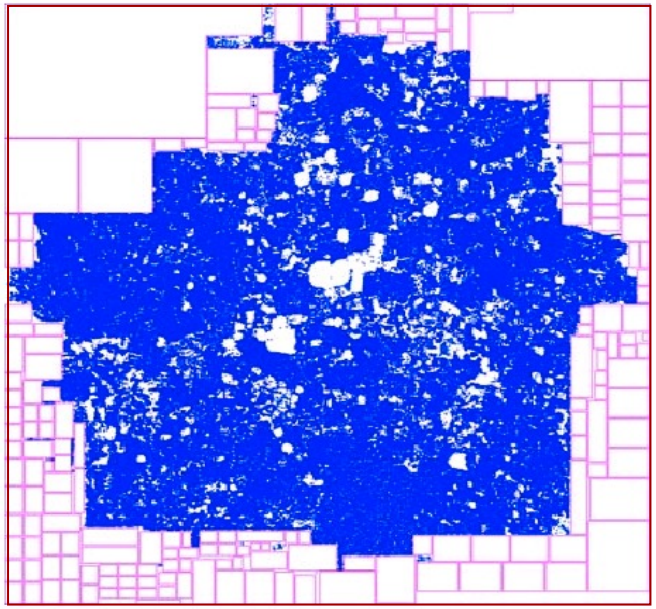
Placement Benchmarks

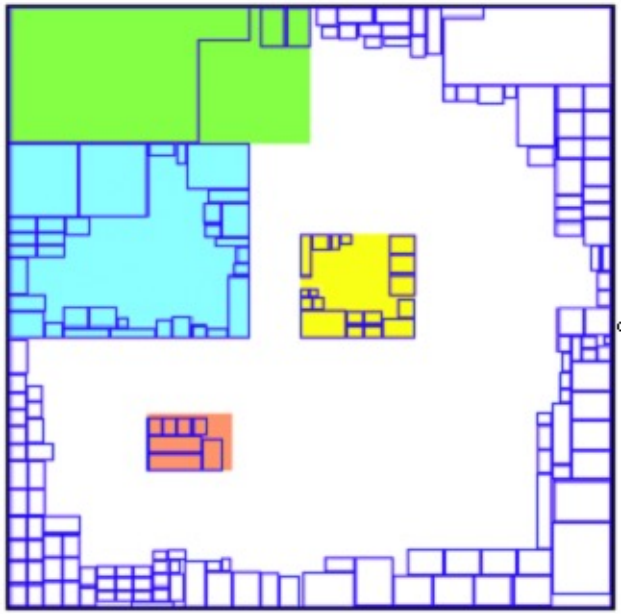
- ISPD 2005 and 2006 benchmark suites
  - Derived from IBM ASIC designs
  - Mixed-size designs with both fixed and movable modules
  - Up to 2.5 million movable modules
- IBM-PLACE 2.0
  - Derived from smaller IBM circuits released in 1998
  - Has standard cells only and no connection to I/O pads
- IBM-MSwPins
  - Derived from the same circuits as IBM-PLACE 2.0
  - Contain large movable macros and fixed pads
- Faraday mixed-sized benchmarks
  - Have sufficient routing information to run industrial routers
- PEKO benchmark suites
  - Synthetic placement benchmarks with known optimal wirelength
  - Many variations

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Large-Scale Mixed-Size Placement

- We still have a long road to go for large-scale mixed-size placement!!
  - Is the two-stage approach the best alternative?
  - Should we pack macros along the chip boundaries?
  - Need to consider many other placement constraints?

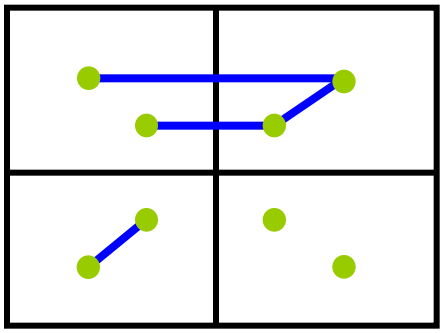




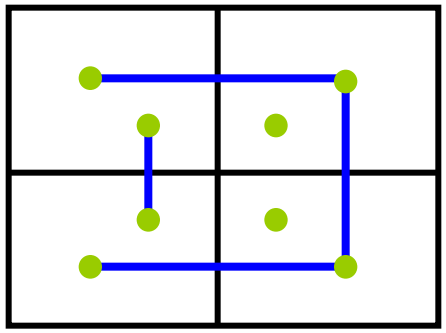
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Routability-Driven Placement

- Most placement algorithms focus on wirelength minimization alone
- Congestion objective differs from the wirelength one
  - Jiang, Su, Chang, DAC-08



Wirelength minimization
Maximum congestion = 2

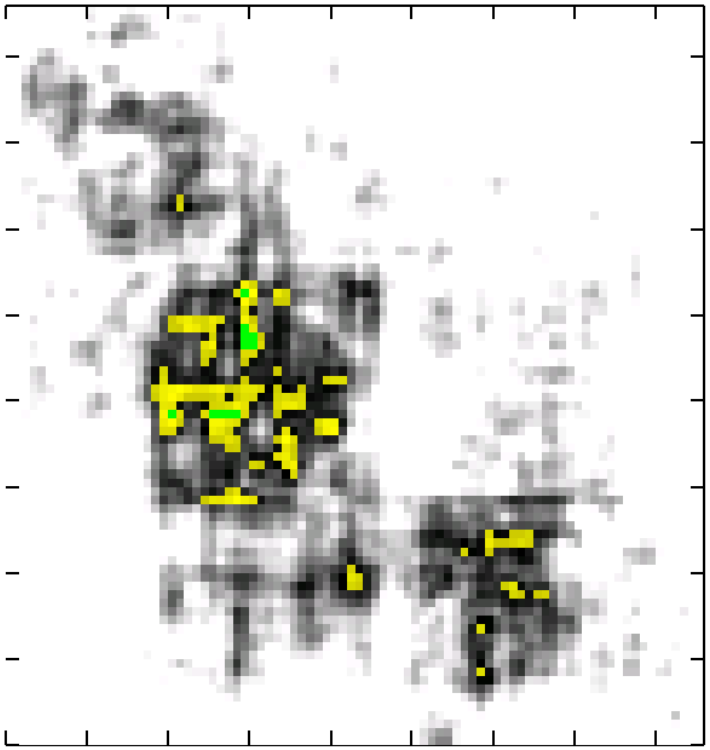


Congestion optimization
Maximum congestion = 1

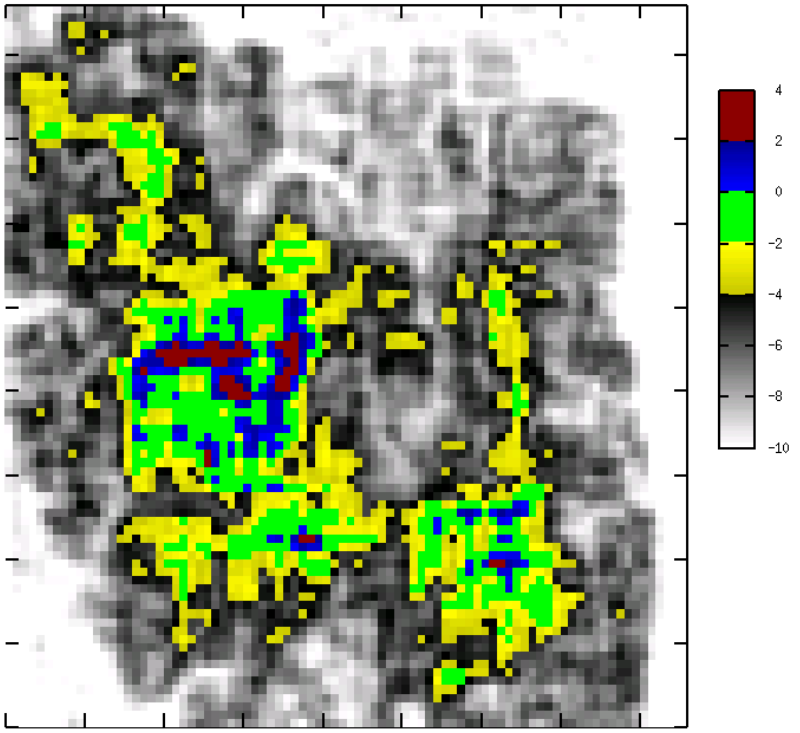
- Further, placement should interact with routing
  - Fast and accurate routing demand estimation for placement?
  - Simultaneous placement and routing?
    - Pan & Chu, DAC-07

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Example Congestion Maps



Congestion optimized result
violations = 0

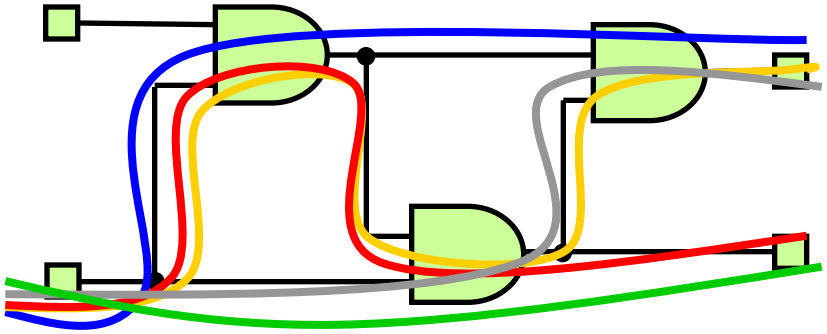


Wirelength minimized result
violations = 6057

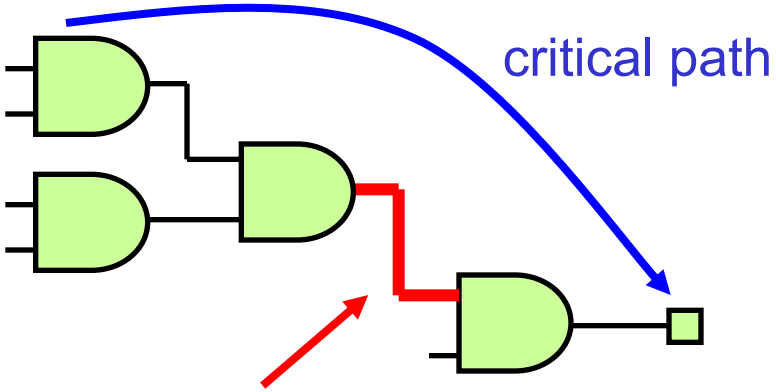
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Timing-Driven Placement

- Two major techniques for timing-driven placement
  - Path-based methods incur prohibitively time complexity due to the exponentially increasing number of paths
  - Net-based methods lack the global view of the full path



timing paths increases exponentially



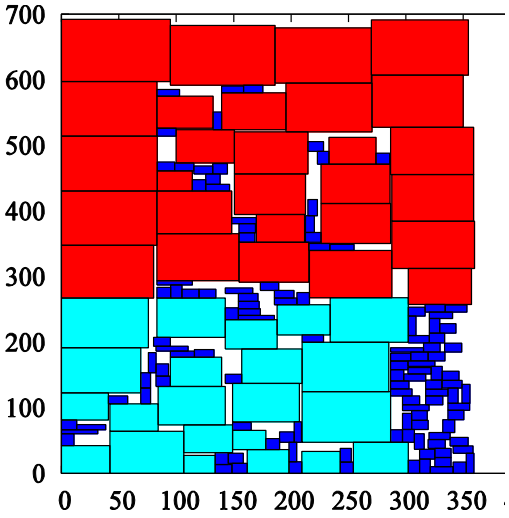
Net based method may not optimize timing effectively

- A timing optimization technique with low complexity and high controllability is desired

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

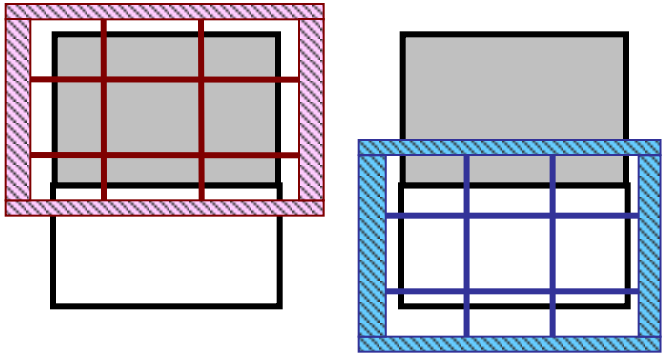
Power-Aware Placement

- Multiple supply voltages are widely used for low-power design
  - Voltages can be assigned during floorplanning/placement
- Integration of voltage assignment and placement can further reduce the power consumption
  - Lee, Liu, Chang, ICCAD-06, ICCAD-07



Example voltage assignment

High voltage block
Low voltage block
Level shifter



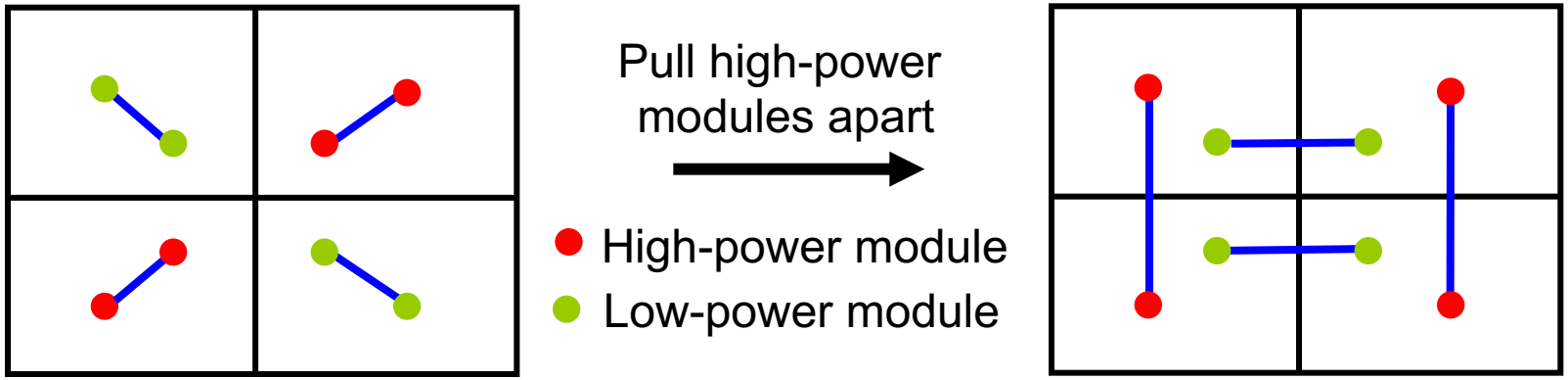
Resulting power ring structure

High voltage power ring
Low voltage power ring

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Thermal-Aware Placement

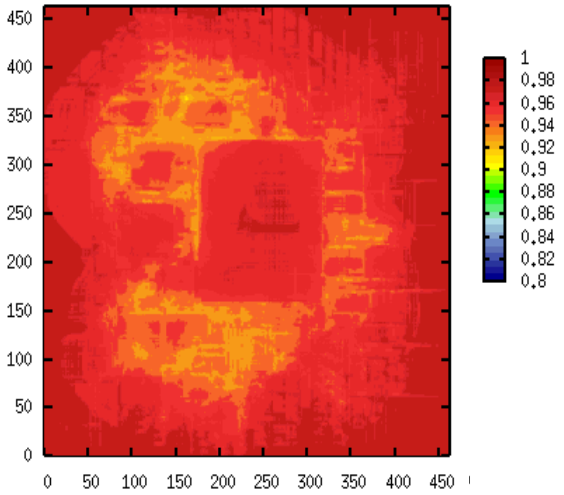
- The increasing integration density and clock frequency raise the chip temperature significantly
- Power density is a dominant factor to reduce the chip temperature
  - Reducing power consumption alone is not sufficient
- Need to spread hot cells to lower chip temperature and thus to increase chip reliability



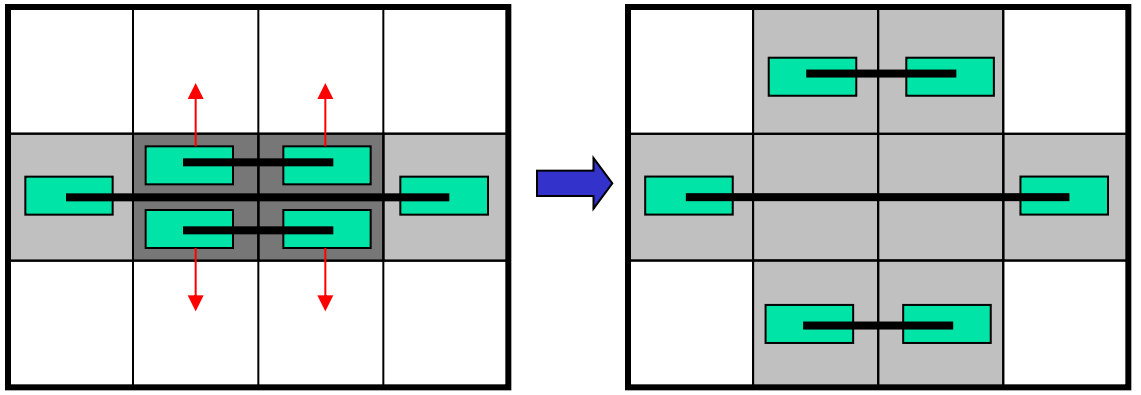
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Manufacturability-Aware Placement

- Predictive Chemical Mechanical Polishing (CMP) model
  - The number of dummy fills and normalized copper thickness are functions of wire density [Cho et al, ICCAD-06]
- Wire density optimization is limited by pin locations
- Shall move cells/pins out of the high wire-density regions during placement
  - Chen, et al., ISPD-08 (TCAD, 2008)



Normalized Copper Thickness Map

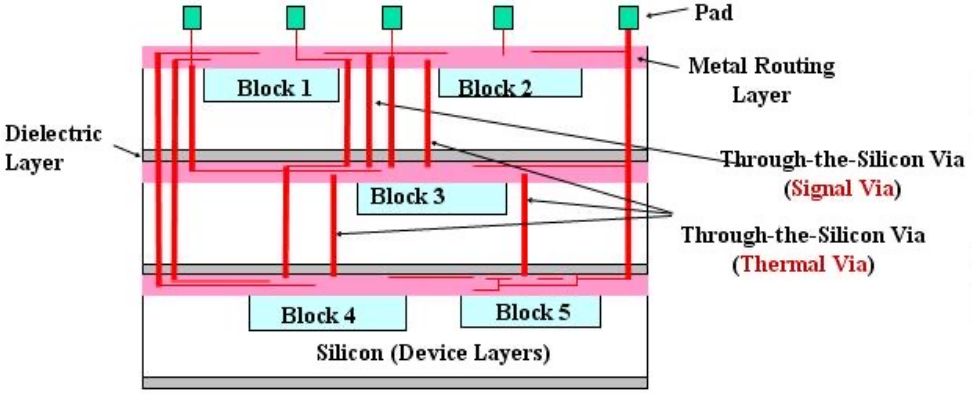


Better wire distribution

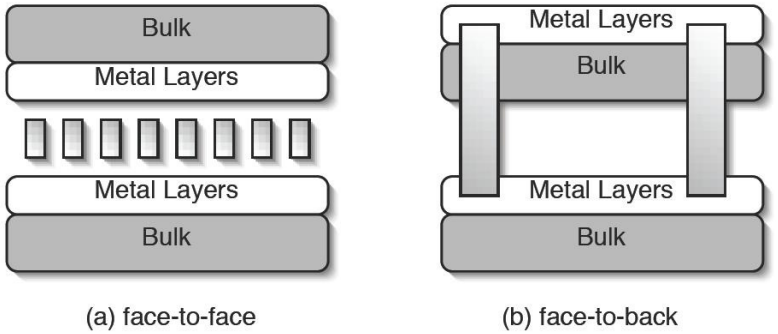
Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement for 3D (2.5D) IC's

- 3D (2.5D) IC technology has been proposed to alleviate interconnect problems
- Thermal dissipation is the major challenge for 2.5D design.
- Dummy thermal vias are often inserted to dissipate heat
- Two integration strategies for 2.5D ICs: face-to-face (F2F) and face-to-back (F2B).
  - F2F is more popular due to its lower cost
- Problem: Place cells to optimize wirelength and thermal



Cross section view of a 2.5D IC

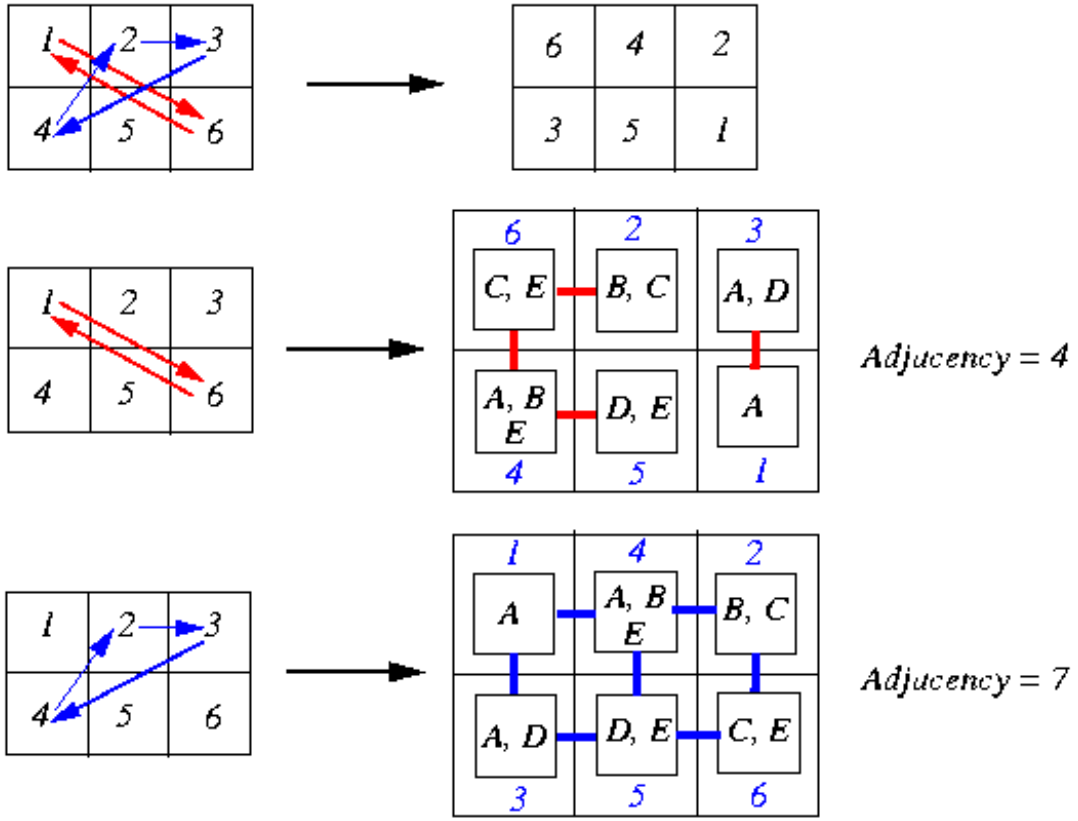


Two types of 2.5D IC's

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

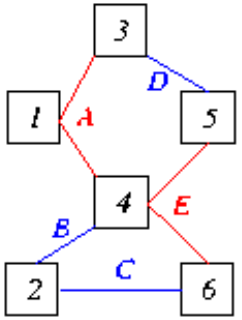
Appendix A:
Placement by Linear Assignment



Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Linear Assignment

- Problem Definition:
  - Instance: An $n \times n$ matrix $D=((d_{ij}))$ with value from $R^+ \cup \{0\}$.
  - Configurations: All permutations $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$
  - Solutions: All configurations
  - Maximize or Minimize $D(\pi) \leftarrow \sum_{i=1}^{n} d_{i,\pi(i)}$
  - *D* can be interpreted as an edge-weighted complete bipartite graph *G* with *n* vertices on each side.
- The linear assignment problem can be solved in $O(n^3)$ time (known fastest algorithm by Fredman & Tarjan: $O(n^2 \log n + mn)$ time).



1	A	A, B E	4
2	B, C	D, E	5
3	A, D	C, E	6

a	b	c
d	e	f

	a	b	c	d	e	f
1	1	1	1	0	0	0
2	1	0	1	1	0	1
3	1	1	1	0	0	0
4	1	0	1	0	0	0
5	1	1	1	1	1	1
6	1	0	1	1	0	1

1. signal A in top row
2. even #'s only in corners

1	3	4
2	5	6

	a	b	c	d	e	f
1	1	1	1	0	0	0
2	1	0	0	1	0	0
3	1	1	1	0	0	0
4	1	0	1	0	0	0
5	1	1	1	1	1	1
6	1	0	0	1	0	0

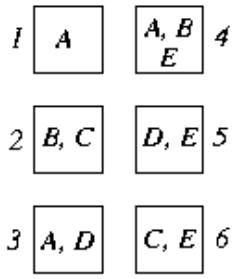
1. signal A in top row
2. even #'s only in corners
3. signal C not on right

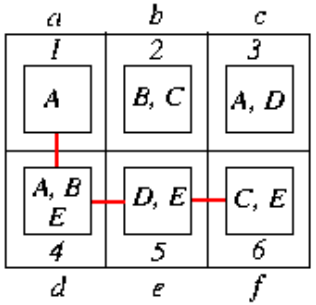
No feasible assignment!

Courtesy of Prof. Y.-W. Chang & H.-M. Chen

Placement by Linear Assignment

- Akers, "On the use of linear assignment algorithm in module placement," DAC-81.
- Total # of signal adjacencies for the placement = 3 (Signal *A*: between modules 1 and 4; Signal *E*: modules 4 and 5; Signal *E*: modules 5 and 6)





J:

	1	2	3	4	5	6
1	0	0	1	1	0	0
2	0	0	0	1	0	1
3	1	0	0	1	1	0
4	1	1	1	0	1	1
5	0	0	1	1	0	1
6	0	1	0	1	1	0

A:

	a	b	c	d	e	f
1	1	1	0	0	1	1
2	1	0	1	0	2	0
3	1	2	0	2	1	1
4	1	3	2	2	2	2
5	1	1	1	0	2	1
6	2	1	1	1	2	1

- J_{ij}: # of common signals between modules *i* and *j*.
- A_{ix}: # of signal adjacencies between module *i* and its neighbors if we move it to location *x*.
- Example: $A_{6a}=J_{62} + J_{64} = 2$; $A_{4e}=J_{42} + J_{44} + J_{46} = 2$.

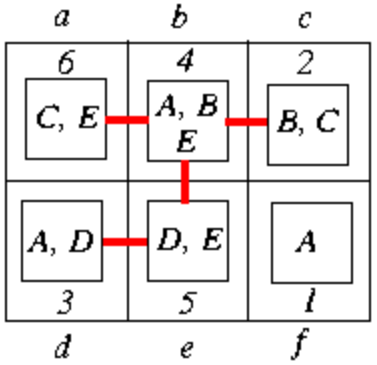
Linear Assignment

A:

	a	b	c	d	e	f
1	1	1	0	0	1	**1**
2	1	0	**1**	0	2	0
3	1	2	0	**2**	1	1
4	1	**3**	2	2	2	2
5	1	1	1	0	**2**	1
6	**2**	1	1	1	2	1

max linear assignment
$= 1+1+2+3+2+2 = 11$

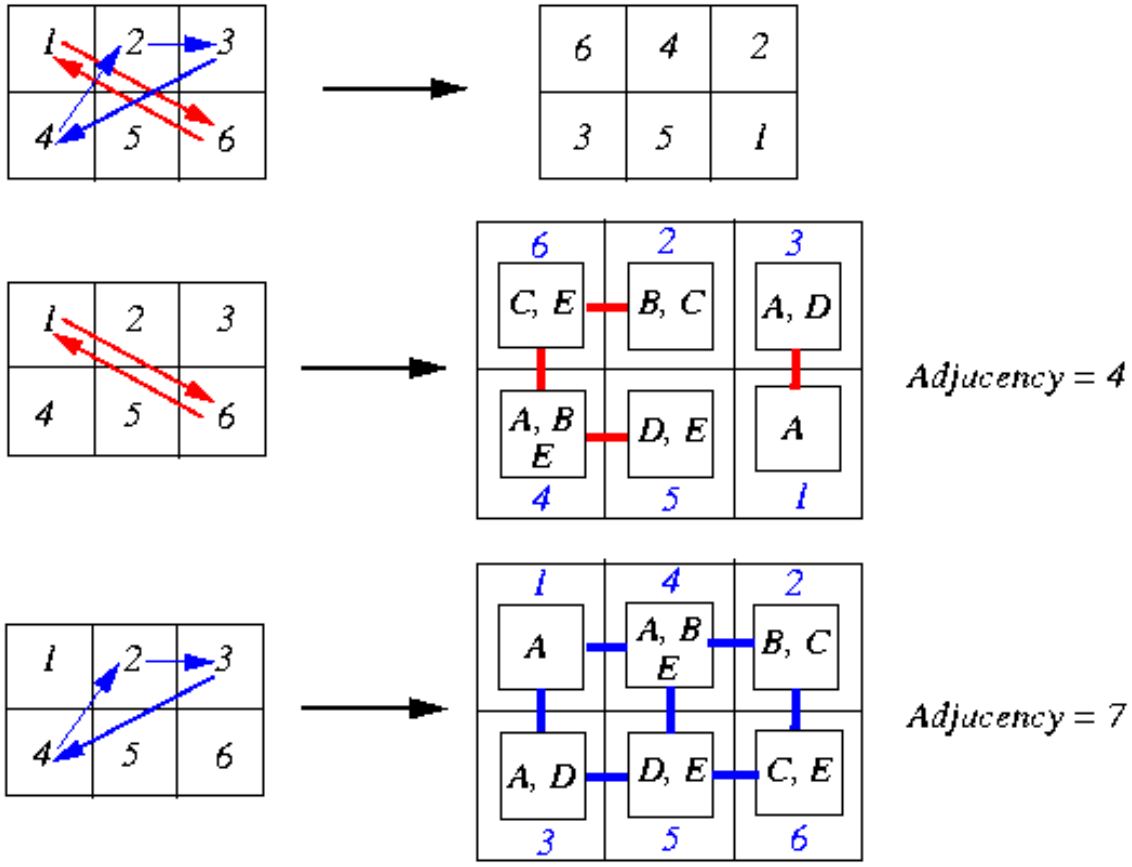
Bad approach: Rearrange the placement according to this solution



Adjacency = 4

- Minor improvement.
  - Reason: When a module reached a “good” position, the module previously adjacent to that position had gone elsewhere!

Better Approach for Linear Assignment



- Modify the placement based on a cycle that yields maximum improvement.
- Repeat the process all over again.

Courtesy of Prof. Y.-W. Chang & H.-M. Chen