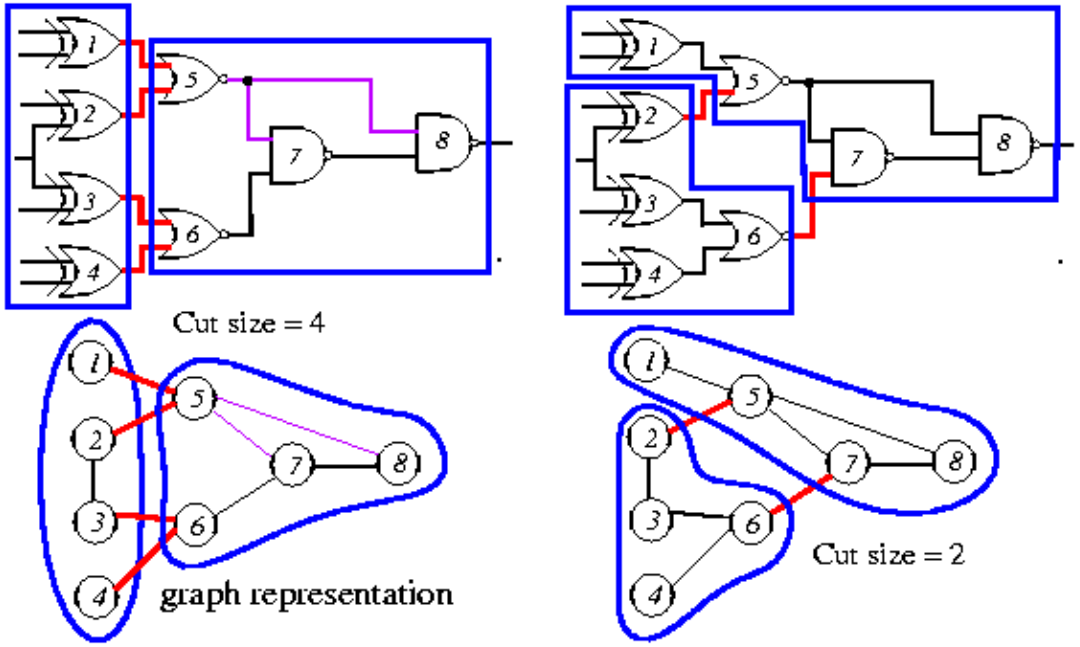
Recap Of Compaction

- From design rules, we create constraint graphs
  - Forward edges, backward edges, cycles
- Find longest paths → use breadth-1st search
- Liao & Wong's algorithm
  - forward to get longest paths, backward edges to update cost, …
- Bellman-and-Ford
  - Using two sets
- Applications for building block layout, …
  - Interactive compaction to take out empty space
- Topological order
  - Useful for project planning

- This course talks about many algorithms

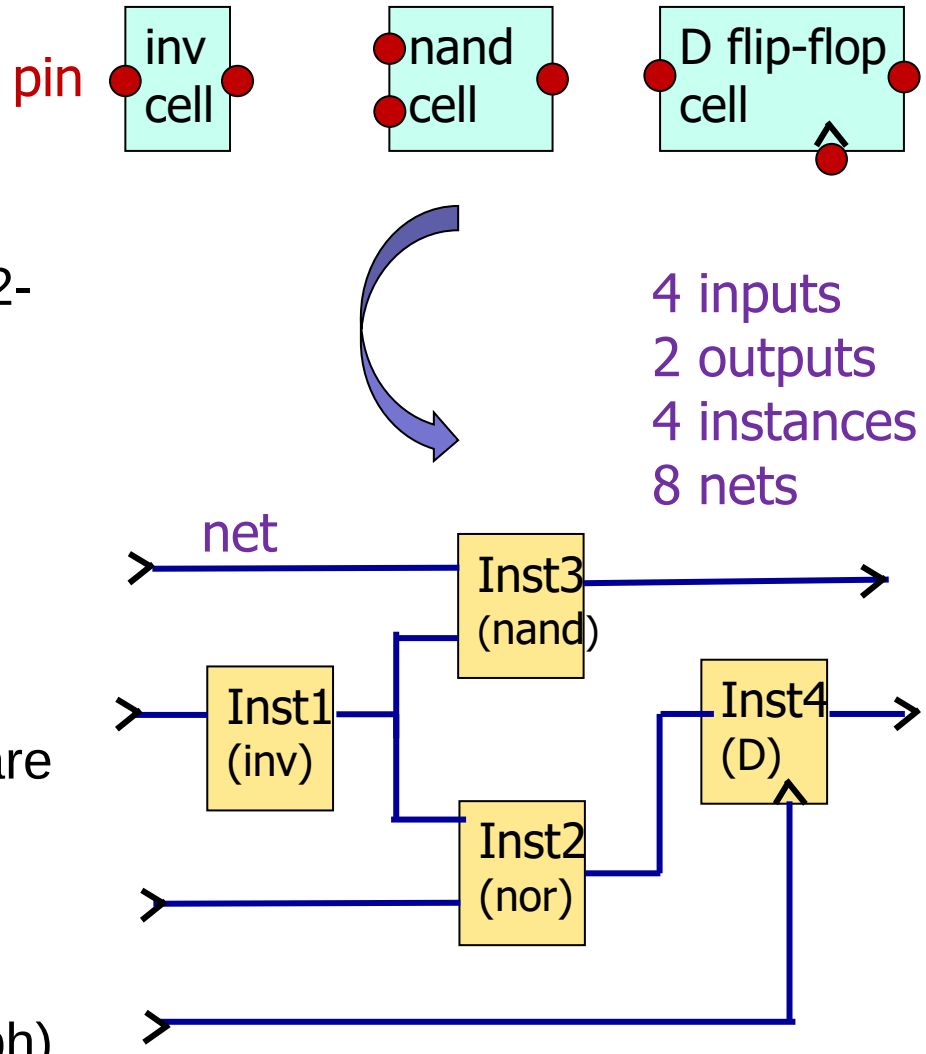
Unit 5A: Circuit Partitioning

- Course contents:
  - Kernighang-Lin partitioning algorithm
  - Simulated annealing based partitioning algorithm
- Readings
  - Chapter 7.5 and 5.6



Common Terminologies Used In Designs

- Cell: has a basic function, such as NAND, NOR, INV
- Instantiation of cells: instances
- Pin: in each cell, there are connection points, such as for 2-input NAND gates, there are 2 input pins and 1 output pin (usually we skip power and ground pins. However, need to consider multiple VDDs).
- Terminal/port: pins in instances
- (Note: often pin/terminal(port) are interchangeable)
- Net: a connection to multiple terminals
- Netlist: a set of nets (hypergraph)



Popular LEF/DEF For Library Cells & Layout

https://www.ispd.cc/contests/18/lefdefref.pdf
Product Version 5.7 November 2009
(if wanting to do APR, this is important.)

Popular Layout Interface Files

LEF – Library Exchange Format

```
# DEMO4 CHIP - 1280
ARRAY NAMESCASESENSITIVE ON
# PLACEMENT SITE SECTION
SITE CORE1 SIZE 67.2 BY 6 ; # GCD of all Y sizes of Macros END CORE1
SITE IOX SIZE 37.8 BY 444 ; # 151.2 / 4 = 37.8 , 4 sites per pad END IOX
SITE IOY SIZE 436.8 BY 30 ; # 150 / 5 = 30 , 5 sites per pad END IOY
SITE SQUAREBLOCK SIZE 268.8 BY 252 ; END SQUAREBLOCK
SITE I2BLOCK SIZE 672 BY 504 ; END I2BLOCK
SITE LBLOCK SIZE 201.6 BY 168 ; END LBLOCK
SITE CORNER SIZE 436.8 BY 444 ; END CORNER
...
LAYER M1 TYPE ROUTING ; DIRECTION VERTICAL ;
    PITCH 5.6 ; WIDTH2.6 ; SPACING 1.5 ;
END M1
LAYER CUT12 TYPE CUT ; END CUT12
LAYER M2 TYPE ROUTING ; DIRECTION HORIZONTAL ;
    PITCH 6.0 ; WIDTH 3.2 ;SPACING 1.6 ;
END M2
LAYER CUT23 TYPE CUT ; END CUT23
LAYER M3 TYPE ROUTING ; DIRECTION VERTICAL ;
    PITCH 5.6 ; WIDTH 3.6; SPACING 1.6 ;
END M3
```

```
MACRO IN1X class pad ;
  FOREIGN IN1X ; SIZE 151.2 BY 444 ;
  SYMMETRY X ;  SITE IOX ;
PIN Z DIRECTION OUTPUT ;
   PORT LAYER M1 ; PATH 61.6 444 72.8
444 ; END
END Z
PIN PO DIRECTION OUTPUT ;
  PORT LAYER M1 ; PATH 78.4 444 84.0
444 ; END
END PO
PIN A DIRECTION INPUT ;
  PORT LAYER M1 ; PATH 95.2 444 100.8
444 ; END
END A
```

FinFet Coloring Rule Impacting Cell Placement

(will re-visit when covering placement)

It is possible, but the current cell row placement rule is defined in a more general way.

LEF can define standard cell edge spacing rules, and therefore foundry can define a minimum spacing between two cell edges.

An example in https://iccad-contest.org/2017/Problem_C/default.html

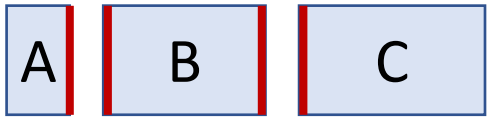
```
PROPERTYDEFINITIONS
   MACRO LEF58_EDGETYPE STRING ;
   LIBRARY LEF58_CELLEDGESPACINGTABLE STRING
"CELLEDGESPACINGTABLE
   EDGETYPE 1 2 0.400
   EDGETYPE 1 1 0.400
   EDGETYPE 2 2 0.000
;" ;
END PROPERTYDEFINITIONS
```

For each cell, two properties are defined for the left and right edges in file "cells_modified.lef". For instance, the following statement is the properties defined for **MACRO ms00f80** of design "fft_2_md2":

```
   PROPERTY LEF58_EDGETYPE "
      EDGETYPE LEFT 2 ;
      EDGETYPE RIGHT 2 ;
" ;
```

So if there is the situation you mentioned, the tech file will include **edgetype** and **celledgespacingtable** to have

1. A_RIGHT to B_LEFT a_non_zero_spacing
2. B_RIGHT to C_LEFT a_non_zero_spacing



During legalization, the spacing rule needs to be followed to prevent any design rule violations.

Tung-Chieh

DEF – Design Exchange Format

```
DESIGN DEMO4CHIP ;
  TECHNOLOGY DEMO4CHIP ;
  ARRAY DEMO4 ;
  UNITS DISTANCE MICRONS 100 ;
COMPONENTS 243 ;
- CORNER1 CORNER ; - CORNER2 CORNER ;
- C01 IN1X ; - C02 IN1Y ; - C04 IN1X ; - C05 IN1X ;
- C13 BIDIR1Y ; - C14 INV ; - C15 BUF ; - C16 BUF ;
- C17 BUF ; - C19 BIDIR1Y ; - C20 INV ; - C21 BUF ;

NETS 222 ;
- VDD ( Z216 B ) ( Z22 Z ) ( Z21 Z ) ( Z16 Z ) ( Z15 Z ) (
D45 PO ) ( D14 PO ) ( C01 PI ) ( D39 TN ) ( D33 TN ) (
D14 TN ) ( D08 TN ) ( D02 TN ) ( C56 TN ) ( C50 TN ) ;
- Z38QN1 ( Z38A01 QN ) ( Z207 B ) ;
- COZ101 ( Z38A01 Q ) ( Z207 A ) ;
- XX901 ( Z236 A ) ( Z234 A ) ( Z231 A ) ( Z229 A ) ( Z227
A ) ( Z226 Z ) ( Z193 A ) ;
```

For nets, there are more data, such as wires, vias, … (describing the wiring)

https://www.ispd.cc/contests/18/lefdefref.pdf

Product Version 5.7 November 2009

Example 4-11 Multi-mask Patterns for Routing Points

The following example shows a routing statement that specifies three-mask layers M1 and VIA1, and a two-mask layer M2:

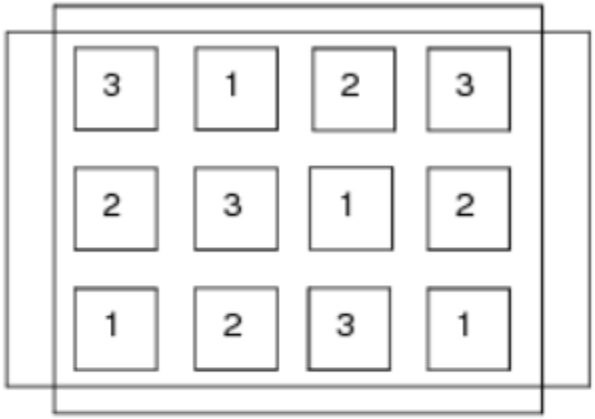
```
+ ROUTED M1 (10 0 ) MASK 3 (10 20 ) VIA1_1

  NEW M2 ( 10 10 ) (20 10) MASK 1 ( 20 20 ) MASK 031 VIA1_2 ;
```

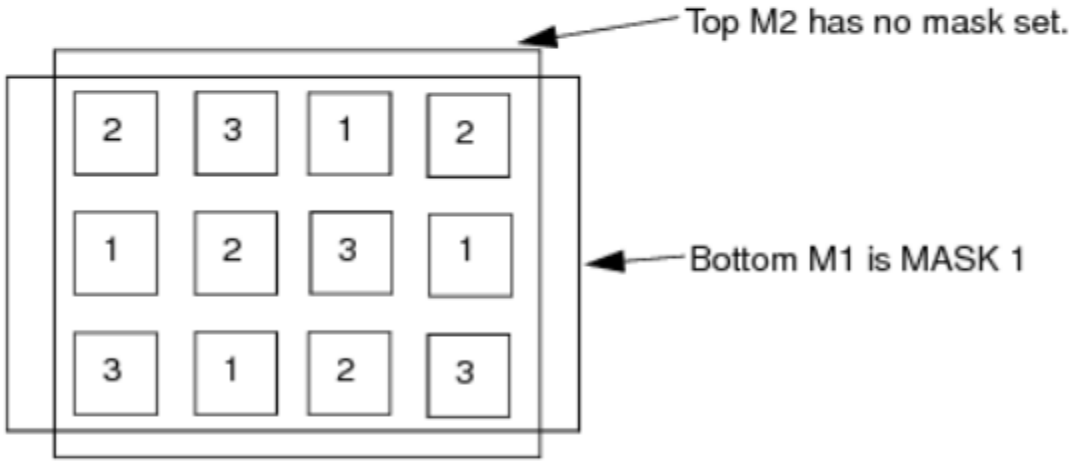
This indicates that the:

- M1 wire shape (10 0) to (10 20) belongs to mask 3
- VIA1_1 via has no preceding MASK statement so all the metal and cut shapes have no mask and are uncolored
- first NEW M2 wire shape (10 10) to (20 10) has no mask set and is uncolored
- second M2 wire shape (20 10) to (20 20) is on mask 1
- VIA1_2 via has a MASK 031 (it can be MASK 31 also) so:
  - *topMaskNum* is 0 in the 031 value, so no mask is set for the top metal (M2) shape
  - *bottomMaskNum* is 1 in the 031 value, so mask 1 is used for the bottom metal (M1) shape
  - *cutMaskNum* is 3 in the 031 value, so the bottom-most, and then the left-most cut of the via-instance is mask 3. The mask for the other cuts of the via-instance are derived from the via-master by "shifting" the via-master's cut masks to match. So if the via-master's bottom-left cut is mask 1, then the via-master cuts on mask 1 become mask 3 for the via-instance, and similarly cuts on 2 shift to 1, and cuts on 3 shift to 2, as shown in Figure 4-4.

Figure 4-4 Via-master multi-mask patterns

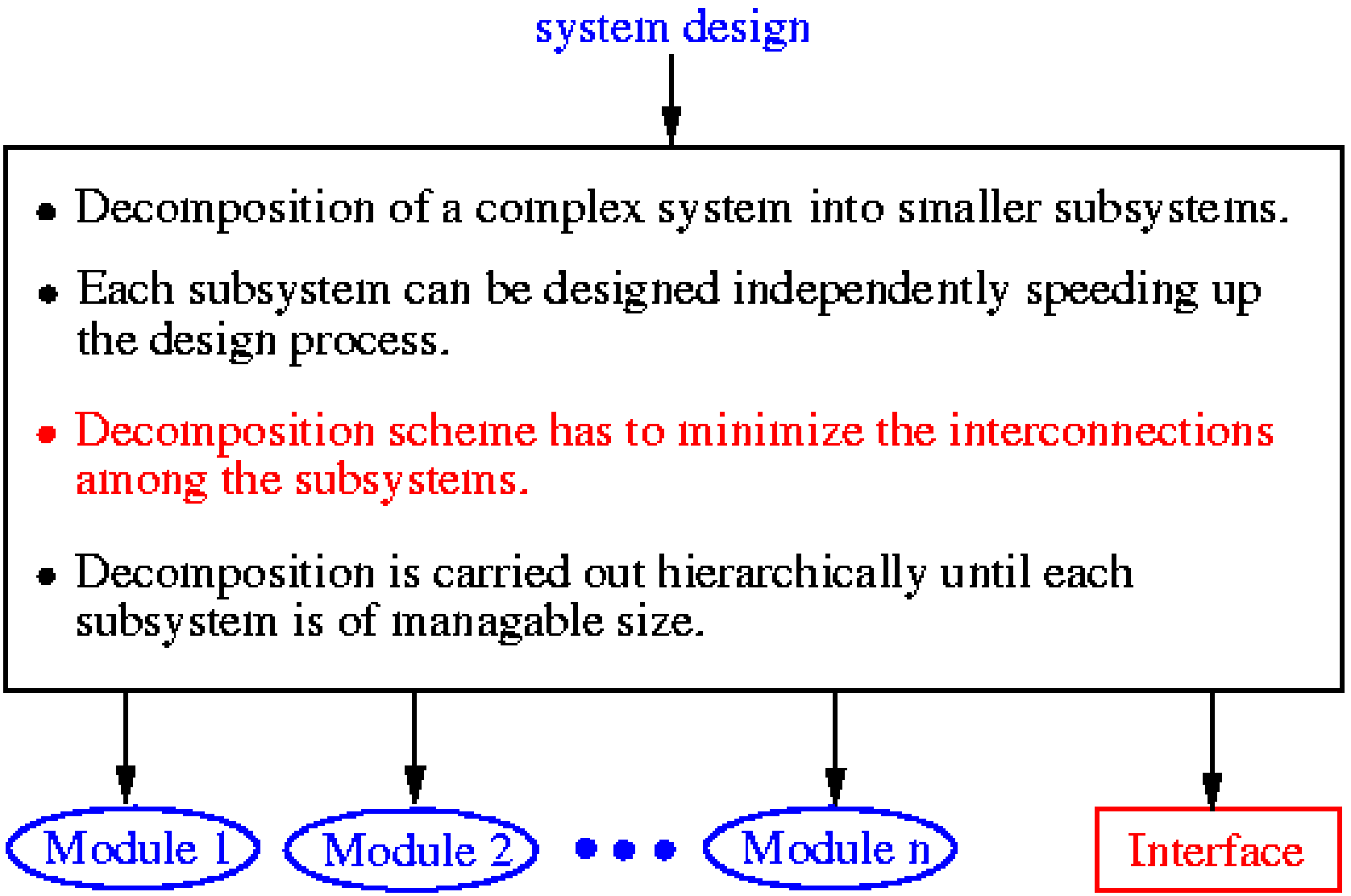


Via-master cut masks for VIA1_2.



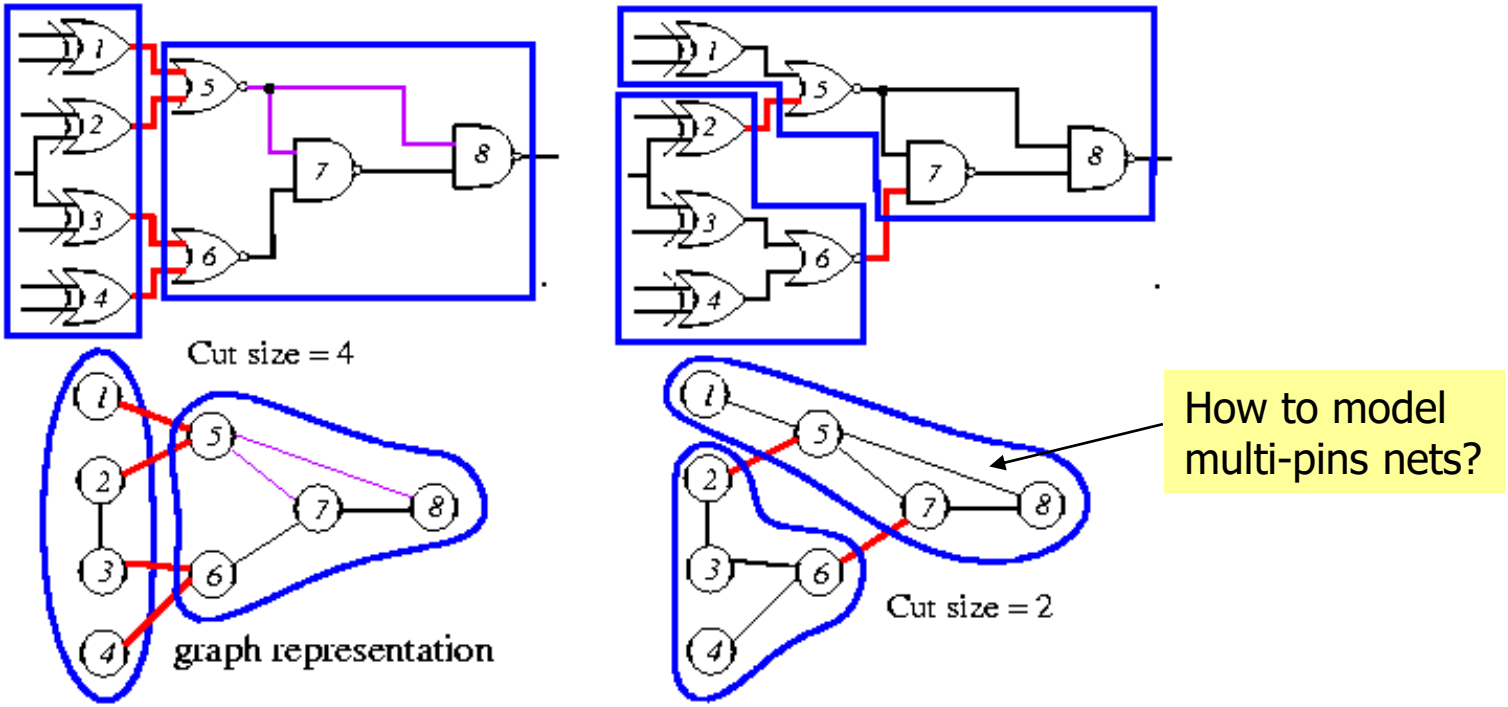
Masks for via-instance: ... (20 20) MASK 031 VIA1_2;
Bottom M1 is MASK 1.Top M2 has no mask set. Lower-left cut is MASK 3, and other via-instance cut shape masks are derived from via-master cut-masks as shown above by "shifting" the via-master masks to match: 1->3, 2->1, 3->2.

Partitioning

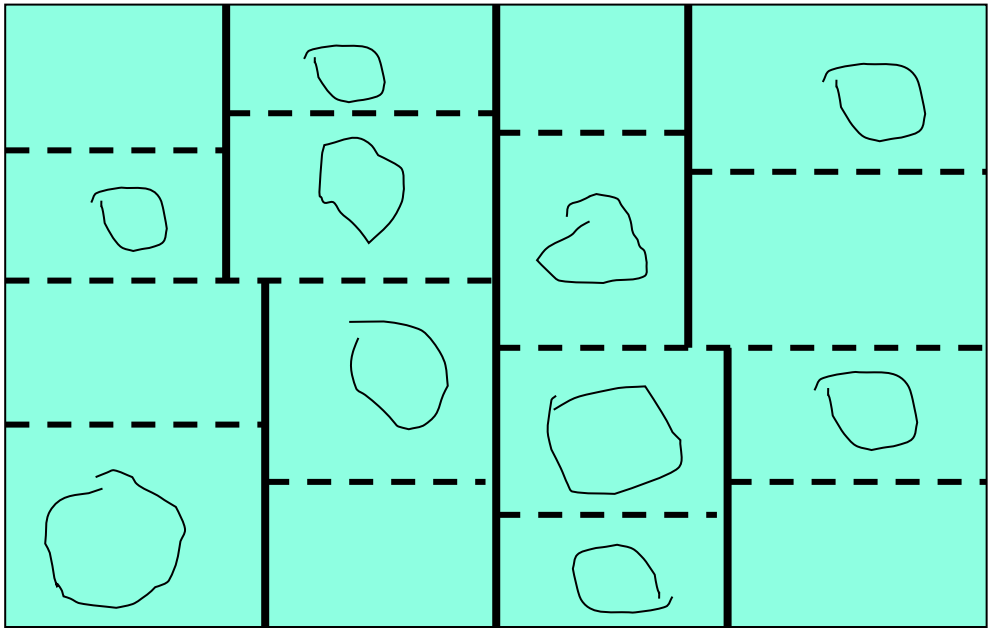


Circuit Partitioning

- **Objective:** Partition a circuit into parts such that every component is within a prescribed range and the # of connections among the components is minimized.
  - More constraints are possible for some applications.
- Cutset? Cut size? Size of a component?



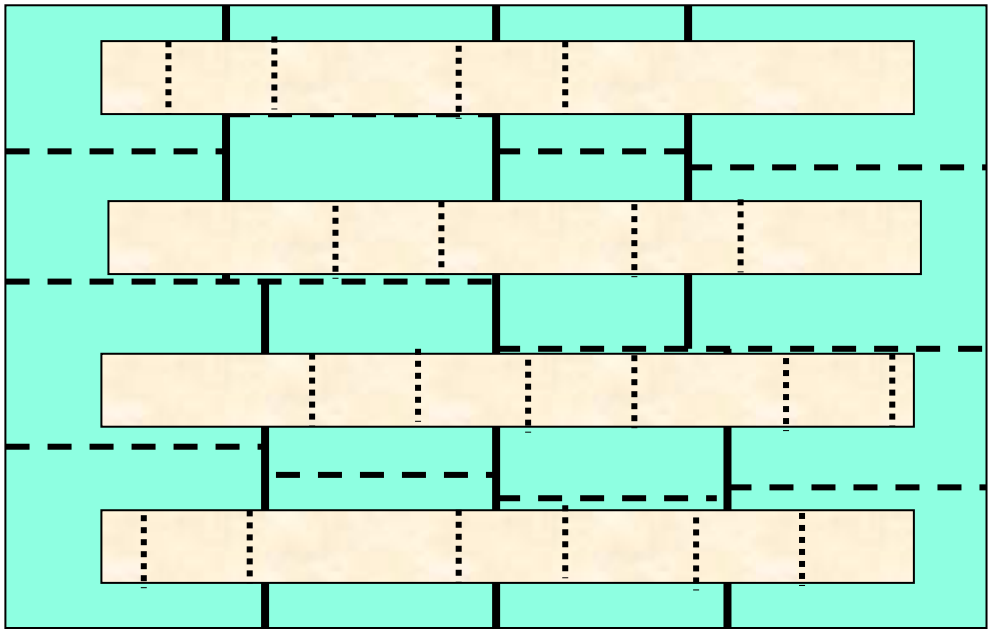
What Can Partitioning Achieve?



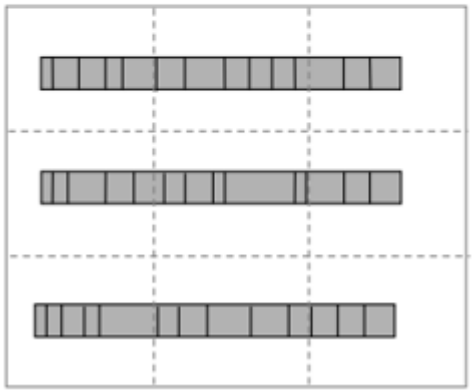
Then, in the bottom level boxes, perform detailed placement

Then Cell Rows Are Considered

From partitioning results, we map instances to rows without overlapping



Detailed Placement



Problem Definition: Partitioning

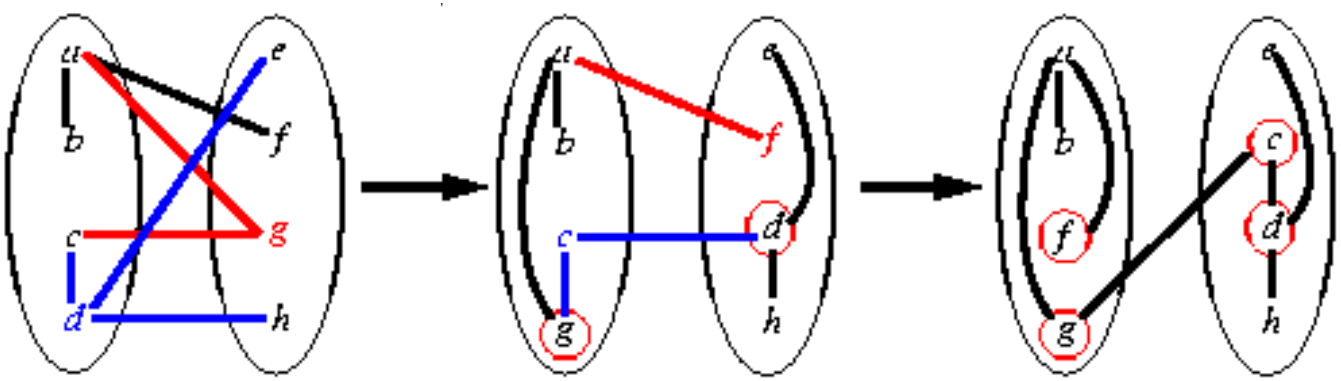
- ***k*-way partitioning:** Given a graph $G(V, E)$, where each vertex $v \in V$ has a **size** $s(v)$ and each edge $e \in E$ has a **weight** $w(e)$, the problem is to divide the set V into k disjoint subsets $V_1, V_2, \ldots, V_k$, such that an objective function is optimized, subject to certain constraints.
- **Bounded size constraint:** The size of the i-th subset is bounded by B_i ($\sum_{v \in V_i} s(v) \leq B_i$).
  - Is the partition balanced?
- **Min-cut cost between two subsets:**
  Minimize $\sum_{\forall e=(u,v) \wedge p(u) \neq p(v)} w(e)$, where $p(u)$ is the partition # of node u.
- The 2-way, balanced partitioning problem is NP-complete, even in its simple form with identical vertex sizes and unit edge weights.

Kernighan-Lin Algorithm

- Kernighan and Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, Feb. 1970.
- An **iterative**, **2-way**, **balanced** partitioning (bi-sectioning) heuristic.
- Till the cut size keeps decreasing
  - Given initial two subsets; find an initial # cuts
  - Vertex pairs, which give the largest decrease or the smallest increase in cut size, are exchanged.
  - These vertices are then **locked** (and thus are prohibited from participating in any further exchanges).
  - This process continues until all the vertices are locked.
  - Find the set with the largest partial sum for swapping.
  - Unlock all vertices.

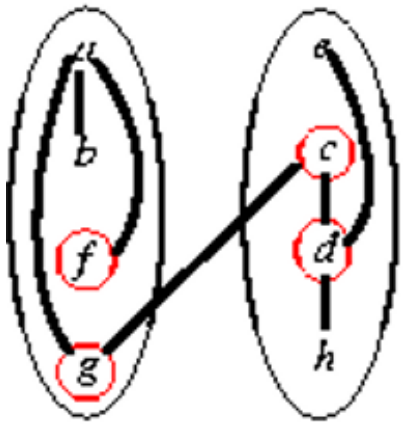
K-L Algorithm: A Simple Example

- Each edge has a unit weight.

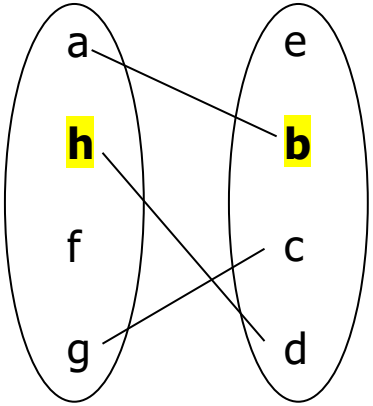


Step #	Vertex pair	Cost reduction	Cut cost
0	-	0	5
1	{d, g}	3	2
2	{c, f}	1	1
3	{b, h}	-2	3
4	{a, e}	-2	5

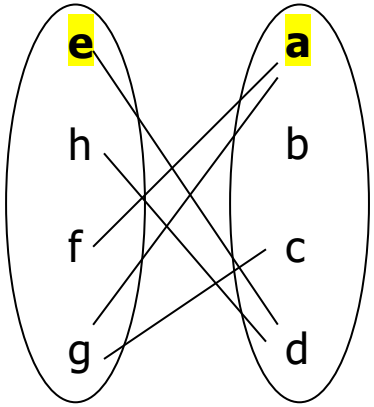
- Questions: How to compute cost reduction? What pairs to be swapped?
  - Consider the change of internal & external connections.



swap b, h



swap a, e



Step #	Vertex pair	Cost reduction	Cut cost
0	-	0	5
1	{d, g}	3	2
2	{c, f}	1	1
3	{b, h}	-2	3
4	{a, e}	-2	5

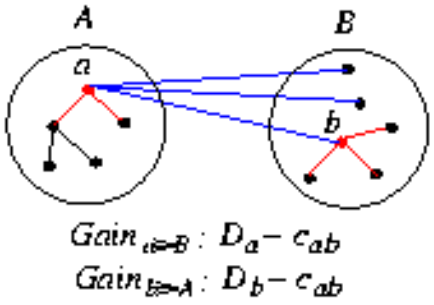
This process continues until all the vertices are locked, since we want to avoid ping-pong situation

Properties

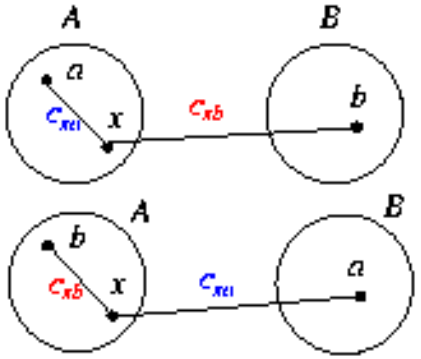
- Two sets A and B such that $|A| = n = |B|$ and $A \cap B = \emptyset$.
- **External cost** of $a \in A$: $E_a = \sum_{v \in B} c_{av}$.
- **Internal cost** of $a \in A$: $I_a = \sum_{v \in A} c_{av}$.
- D-value of a vertex a: $D_a = E_a - I_a$ (cost reduction for moving a).
- Cost reduction (gain) for swapping a and b: $g_{ab} = D_a + D_b - 2c_{ab}$.
- If $a \in$ A and $b \in B$ are interchanged, then the new D-values, D', are given by

$$\left\{ \begin{array}{lcl} D'_x & = & D_x + 2c_{xa} - 2c_{xb}, \forall x \in A - \{a\} \\ D'_y & = & D_y + 2c_{yb} - 2c_{ya}, \forall y \in B - \{b\}. \end{array} \right\}$$

It contains *Cab*



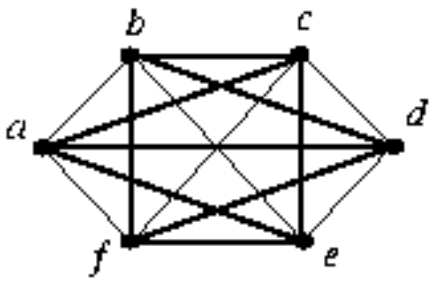
Internal cost vs. External cost



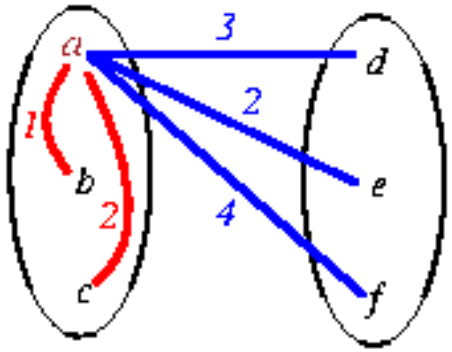
before swap	*after swap*	$\triangle C$
$-c_{xa}$	$+c_{xa}$	$+2c_{xa}$
$+c_{xb}$	$-c_{xb}$	$-2c_{xb}$

updating D−values

K-L Algorithm: A Weighted Example



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



costs associated with a

Initial cut cost = (3+2+4)+(4+2+1)+(3+2+1) = 22

from b **from c**

- Iteration 1:

$$I_a = 1+2 = 3; \quad E_a = 3+2+4 = 9; \quad D_a = E_a - I_a = 9-3 = 6$$
$$I_b = 1+1 = 2; \quad E_b = 4+2+1 = 7; \quad D_b = E_b - I_b = 7-2 = 5$$
$$I_c = 2+1 = 3; \quad E_c = 3+2+1 = 6; \quad D_c = E_c - I_c = 6-3 = 3$$
$$I_d = 4+3 = 7; \quad E_d = 3+4+3 = 10; \quad D_d = E_d - I_d = 10-7 = 3$$
$$I_e = 4+2 = 6; \quad E_e = 2+2+2 = 6; \quad D_e = E_e - I_e = 6-6 = 0$$
$$I_f = 3+2 = 5; \quad E_f = 4+1+1 = 6; \quad D_f = E_f - I_f = 6-5 = 1$$

Computing the g Value

- Iteration 1:

$$I_a = 1+2 = 3; \quad E_a = 3+2+4 = 9; \quad D_a = E_a - I_a = 9-3 = 6$$
$$I_b = 1+1 = 2; \quad E_b = 4+2+1 = 7; \quad D_b = E_b - I_b = 7-2 = 5$$
$$I_c = 2+1 = 3; \quad E_c = 3+2+1 = 6; \quad D_c = E_c - I_c = 6-3 = 3$$
$$I_d = 4+3 = 7; \quad E_d = 3+4+3 = 10; \quad D_d = E_d - I_d = 10-7 = 3$$
$$I_e = 4+2 = 6; \quad E_e = 2+2+2 = 6; \quad D_e = E_e - I_e = 6-6 = 0$$
$$I_f = 3+2 = 5; \quad E_f = 4+1+1 = 6; \quad D_f = E_f - I_f = 6-5 = 1$$

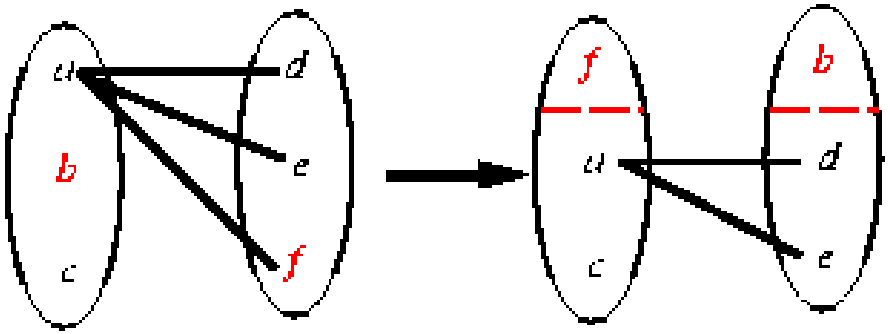
- $g_{xy} = D_x + D_y - 2c_{xy}$.

a with all the nodes on the other side

$$\begin{aligned}
g_{ad} &= D_a + D_d - 2c_{ad} = 6+3-2\times 3 = 3\\
g_{ae} &= 6+0-2\times 2 = 2\\
g_{af} &= 6+1-2\times 4 = -1\\
g_{bd} &= 5+3-2\times 4 = 0\\
g_{be} &= 5+0-2\times 2 = 1\\
g_{bf} &= 5+1-2\times 1 = 4 \ (maximum)\\
g_{cd} &= 3+3-2\times 3 = 0\\
g_{ce} &= 3+0-2\times 2 = -1\\
g_{cf} &= 3+1-2\times 1 = 2
\end{aligned}$$

- Swap *b* and *f*. $(\hat{g_1} = 4)$

Updating the D Value (After Swap)



- $D'_x = D_x + 2\,c_{xp} - 2\,c_{xq}$, $\forall\, x \in A - \{p\}$ (swap p and q, $p \in A$, $q \in B$)

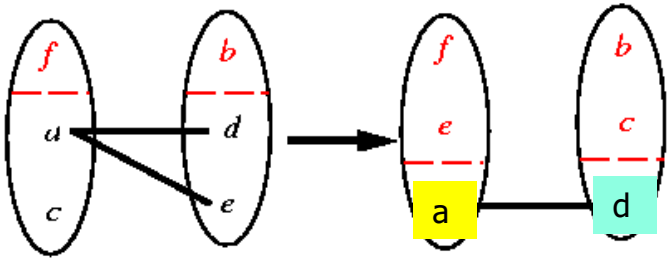
$$
\begin{aligned}
D'_a &= D_a + 2c_{ab} - 2c_{af} = 6 + 2 \times 1 - 2 \times 4 = 0 \\
D'_c &= D_c + 2c_{cb} - 2c_{cf} = 3 + 2 \times 1 - 2 \times 1 = 3 \\
D'_d &= D_d + 2c_{df} - 2c_{db} = 3 + 2 \times 3 - 2 \times 4 = 1 \\
D'_e &= D_e + 2c_{ef} - 2c_{eb} = 0 + 2 \times 2 - 2 \times 2 = 0
\end{aligned}
$$

- $g_{xy} = D'_x + D'_y - 2c_{xy}$.

$$
\begin{aligned}
g_{ad} &= D'_a + D'_d - 2c_{ad} = 0 + 1 - 2 \times 3 = -5 \\
g_{ae} &= D'_a + D'_e - 2c_{ae} = 0 + 0 - 2 \times 2 = -4 \\
g_{cd} &= D'_c + D'_d - 2c_{cd} = 3 + 1 - 2 \times 3 = -2 \\
g_{ce} &= D'_c + D'_e - 2c_{ce} = 3 + 0 - 2 \times 2 = -1 \ (maximum)
\end{aligned}
$$

- Swap c and e. $(\hat{g_2} = -1)$

Determining Swapping Pairs



- $D''_x = D'_x + 2\,c_{xp} - 2\,c_{xq}, \forall\, x \in A - \{p\}$

$$D''_a = D'_a + 2c_{ac} - 2c_{ae} = 0 + 2 \times 2 - 2 \times 2 = 0$$
$$D''_d = D'_d + 2c_{de} - 2c_{dc} = 1 + 2 \times 4 - 2 \times 3 = 3$$

- $g_{xy} = D''_x + D''_y - 2c_{xy}$.

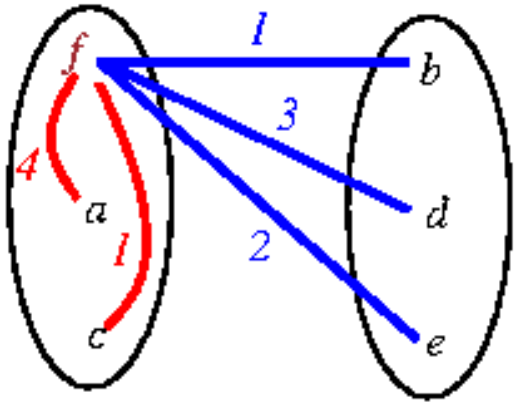
$$g_{ad} = D''_a + D''_d - 2c_{ad} = 0 + 3 - 2 \times 3 = -3 (\hat{g_3} = -3)$$

- Note that this step is redundant $(\sum_{i=1}^{n} \hat{g_i} = 0)$.
- Summary: $\hat{g_1} = g_{bf} = 4,\ \hat{g_2} = g_{ce} = -1,\ \hat{g_3} = g_{ad} = -3.$
- Largest partial sum $\max \sum_{i=1}^{k} \hat{g_i} = 4$ (k = 1) ⇒ Swap *b* and *f*.

Only swap the k=1st pair

Next Iteration – From Previous Result

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



Initial cut cost = (1+3+2)+(1+3+2)+(1+3+2) = 18 (22−4)

From a **From c**

- Iteration 2: Repeat what we did at Iteration 1 (Initial cost = 22 - 4 =18).
- Summary: $\hat{g_1} = g_{ce} = -1,\ \hat{g_2} = g_{ab} = -3,\ \hat{g_3} = g_{fd} = 4.$
- Largest partial sum = $\max \sum_{i=1}^{k} \hat{g_i} = 0$ (k = 3) ⇒ Stop!

Kernighan-Lin Algorithm

Algorithm: Kernighan-Lin(G)
Input: $G = (V, E), |V| = 2n$.
Output: Balanced bi-partition A and B with "small" cut cost.

```
1  begin
2  Bipartition G into A and B such that |V_A| = |V_B|, V_A ∩ V_B = ∅,
       and V_A ∪ V_B = V.
3  repeat
4      Compute D_v, ∀v ∈ V.
5      for i = 1 to n do
6          Find a pair of unlocked vertices v_ai ∈ V_A and v_bi ∈ V_B whose
           exchange makes the largest decrease or smallest increase in
           cut cost;
7          Mark v_ai and v_bi as locked, store the gain ĝ_i, and compute
           the new D_v, for all unlocked v ∈ V;
8      Find k, such that G_k = Σ_{i=1}^{k} ĝ_i is maximized;
9      if G_k > 0 then
10         Move v_a1,...,v_ak from V_A to V_B and v_b1,...,v_bk from V_B to V_A;
11     Unlock v, ∀v ∈ V.
12 until G_k ≤ 0;
13 end
```

Time Complexity

- Line 4: Initial computation of D: $O(n^2)$
- Line 5: The **for**-loop: $O(n)$
- The body of the loop: $O(n^2)$.
  - Lines 6--7: Step i takes $(n - i + 1)^2$ time.
- Lines 4--11: Each pass of the repeat loop: $O(n^3)$.
- Suppose the repeat loop terminates after r passes.
- The total running time: $O(rn^3)$.
  - Polynomial-time algorithm?

Extensions of K-L Algorithm

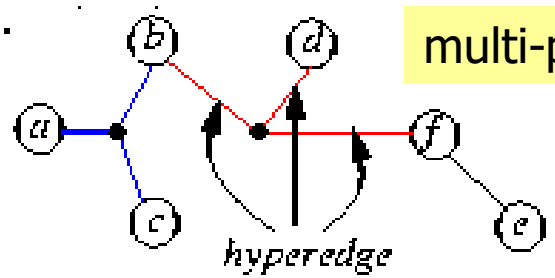
- **Unequal sized subsets** (assume $n_1 < n_2$)
  1. Partition: $|A| = n_1$ and $|B| = n_2$.
  2. Add $n_2 - n_1$ dummy vertices to set A. Dummy vertices have no connections to the original graph.
  3. Apply the Kernighan-Lin algorithm.
  4. Remove all dummy vertices.
- **Unequal sized "vertices"**
  1. Assume that the smallest "vertex'' has unit size. (or largest common denominator)
  2. Replace each vertex of size s with s vertices which are fully connected with edges of infinite weight.
  3. Apply the Kernighan-Lin algorithm.
- ***k*-way partition**
  1. Partition the graph into k equal-sized sets.
  2. Apply the Kernighan-Lin algorithm for each pair of subsets.
  3. Time complexity? Can be reduced by recursive bi-partition.
- How to **handle hypergraphs**?
  - Need to handle multi-terminal nets directly.

Next Homework Is Coming

- There is a homework to go through steps in K-L algorithm

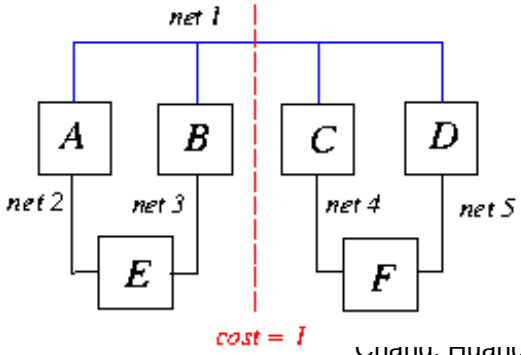
Coping With Hypergraph

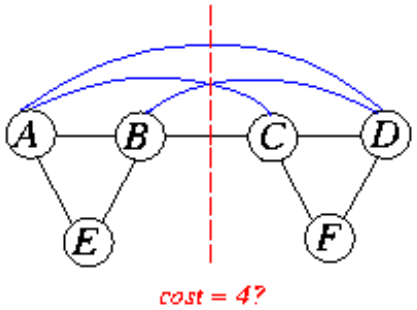
- A hypergraph $H = (N, L)$ consists of a set N of vertices and a set L of hyperedges, where each hyperedge corresponds to a **subset** N_i of distinct vertices with $|N_i| \geq 2$.



multi-pins nets

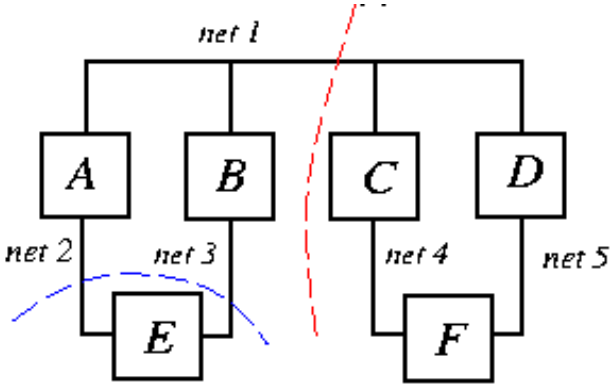
- Schweikert and Kernighan, "A proper model for the partitioning of electrical circuits," 9th Design Automation Workshop, 1972.
- For multi-terminal nets, **net cut** is a more accurate measurement for cut cost (i.e., deal with hyperedges).
  - {A, B, E}, {C, D, F} is a good partition.
  - Should not assign the same weight for all edges.

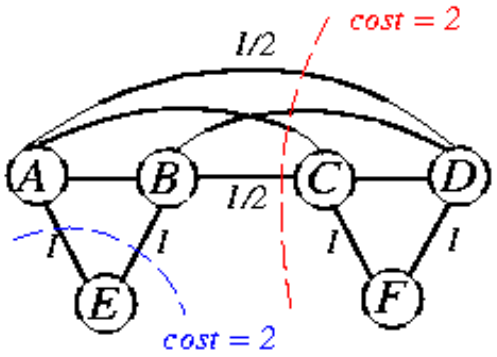




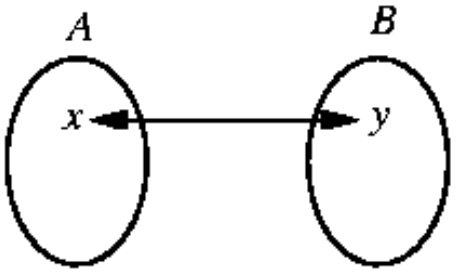
Net-Cut Model

- Let $n(i)$ = # of cells associated with Net i.
- Edge weight $w_{xy} = \frac{2}{n(i)}$ for an edge connecting cells x and y.





- Easy modification of the K-L heuristic.



D_x: gain in moving x to B

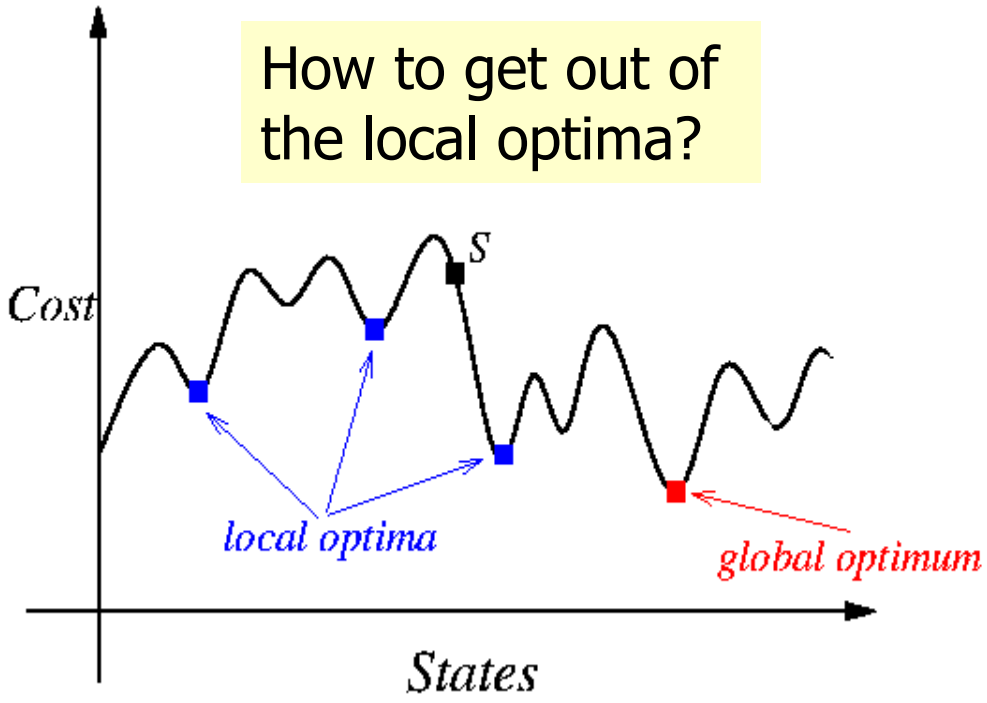
D_y: gain in moving y to A

$g_{xy} = D_x + D_y - Correction(x, y)$

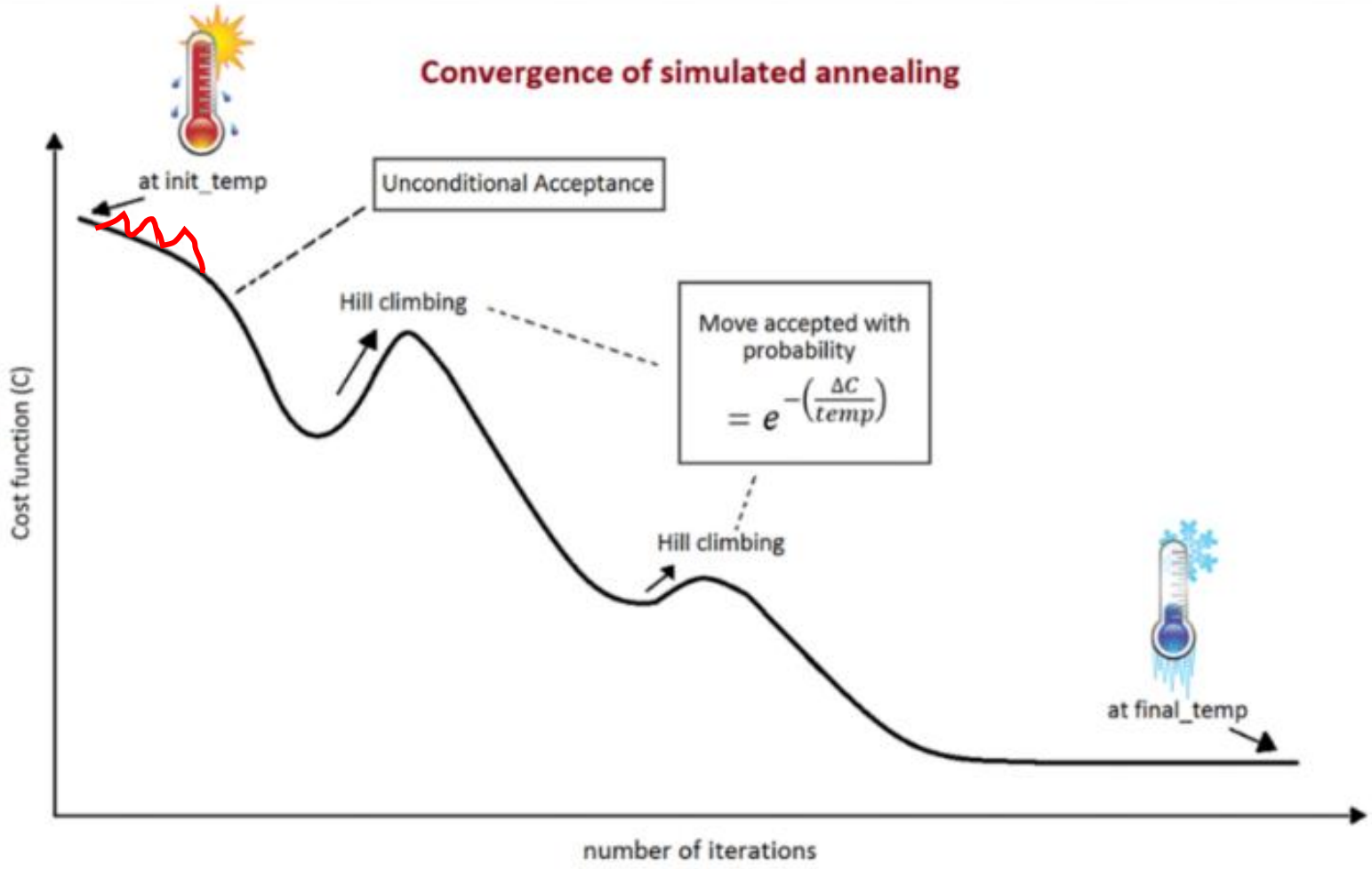
Note: some signal nets, such as **reset**, connect to many pins. Usually no need to include them

Simulated Annealing

- Kirkpatrick, Gelatt, and Vecchi, “Optimization by simulated annealing,” *Science*, May 1983.
- Greene and Supowit, “Simulated annealing without rejected moves,” ICCD-84.



Solving NP Complete Timetable Problem Using Monte Carlo and Simulated Annealing Algorithms



Simulated Annealing Basics

- Non-zero probability for “up-hill” moves.
- Probability depends on
  1) magnitude of the “up-hill” movement
  2) total search time

Do you see anything missing?

Printing not clear

$$Prob(S \rightarrow S') = \begin{cases} 1 & \text{if } \Delta C \leq 0 \quad /* \text{“down-hill” moves} */ \\ e^{-\frac{\Delta C}{T}} & \text{if } \Delta C > 0 \quad /* \text{“up-hill” moves} */ \end{cases}$$

- $\Delta C = cost(S') - Cost(S)$
- T: Control parameter (temperature)
  - So, for high values of T nearly all bad moves are accepted, while hardly any bad move is accepted when T is low.
- Annealing schedule: $T=T_0, T_1, T_2, \ldots$, where $T_i = r^i T_0$, $r < 1$.

Generic Simulated Annealing Algorithm

```
1 begin
2 Get an initial solution S;
3 Get an initial temperature T > 0;
4 while not yet "frozen" do
5    for 1 ≤ i ≤ P do
6        Pick a random neighbor S' of S;
7        Δ ← cost(S') - cost(S);
         /* downhill move */
8        if  Δ ≤ 0 then S ← S'
         /* uphill move */
9        if  Δ > 0 then S ← S' with probability e^(-Δ/T) ;
10 T ← rT;  /* reduce temperature */
11 return S
12 end
```

Random number generation

Basic Ingredients for Simulated Annealing

- Analogy:

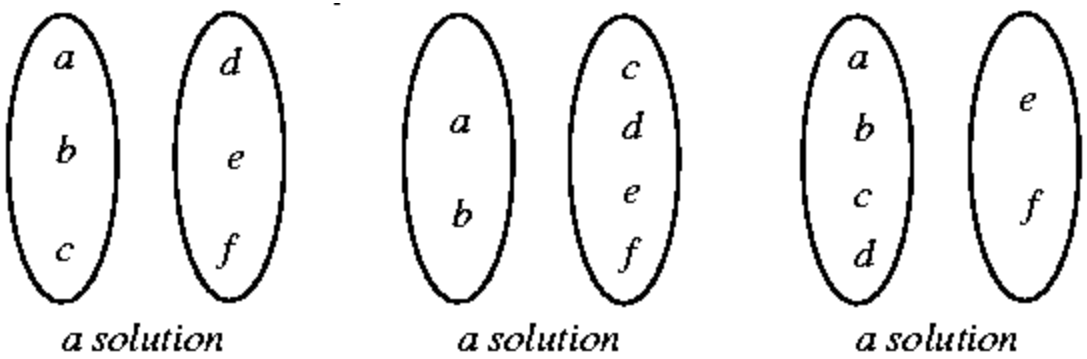
Physical system	Optimization problem
state	configuration
energy	cost function
ground state	optimal solution
quenching	iterative improvement
careful annealing	simulated annealing

- Basic Ingredients for Simulated Annealing:
  - **Solution space**
  - **Neighborhood structure**
  - **Cost function**
  - **Annealing schedule**

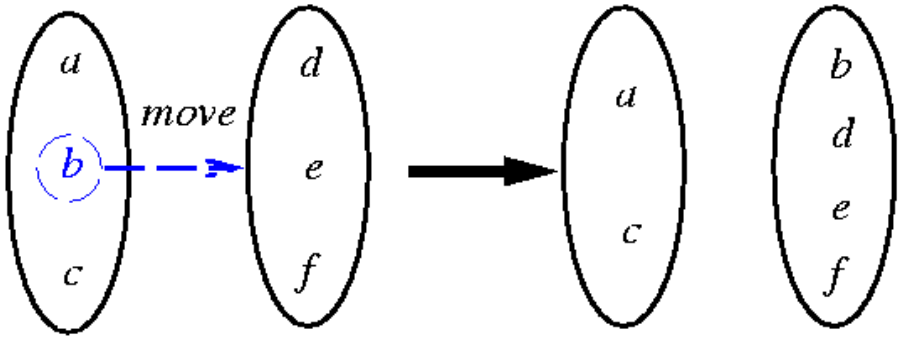
https://slideplayer.com/slide/8038378/

Partition by Simulated Annealing

- Kirkpatrick, Gelatt, and Vecchi, “Optimization by simulated annealing,” *Science*, May 1983.
- **Solution space:** set of all partitions



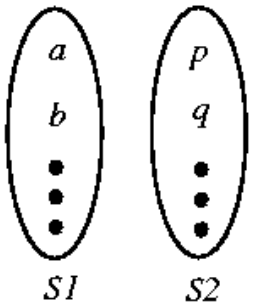
- **Neighborhood structure:**



Randomly move one cell to the other side

Partition by Simulated Annealing (cont'd)

- **Cost function:** $f = C + \lambda B$
  - C: the partition cost as used before.
  - B: a measure of how balance the partition is
  - λ: a constant

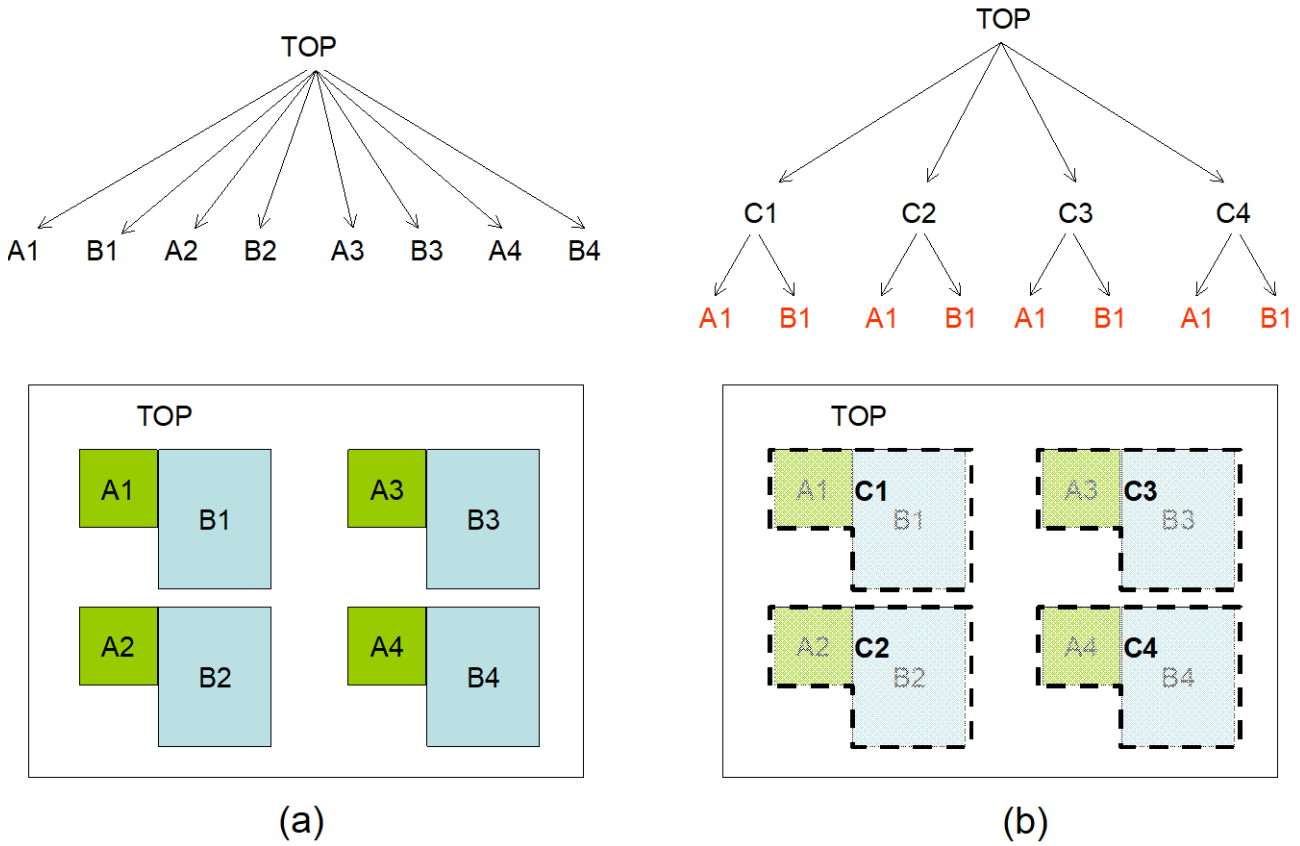


$$B = (|S1| - |S2|)^2$$

- **Annealing schedule:**
  - $T_n = r^n T_0$, $r = 0.9$.
  - At each temperature, either
    1. there are 10 accepted moves/cell on the average, or
    2. # of attempts $\geq$ 100 x total # of cells.
  - The system is "frozen" if very low acceptances at 3 consecutive temperatures.

Clustering

- **Clustering:** Reduce the problem size by grouping highly connected (related) components and treat them as a super node.
  - 2003 MOE IC/CAD Contest: Problem 7



Any Other Constraints You Can Think Of?

For partitioning

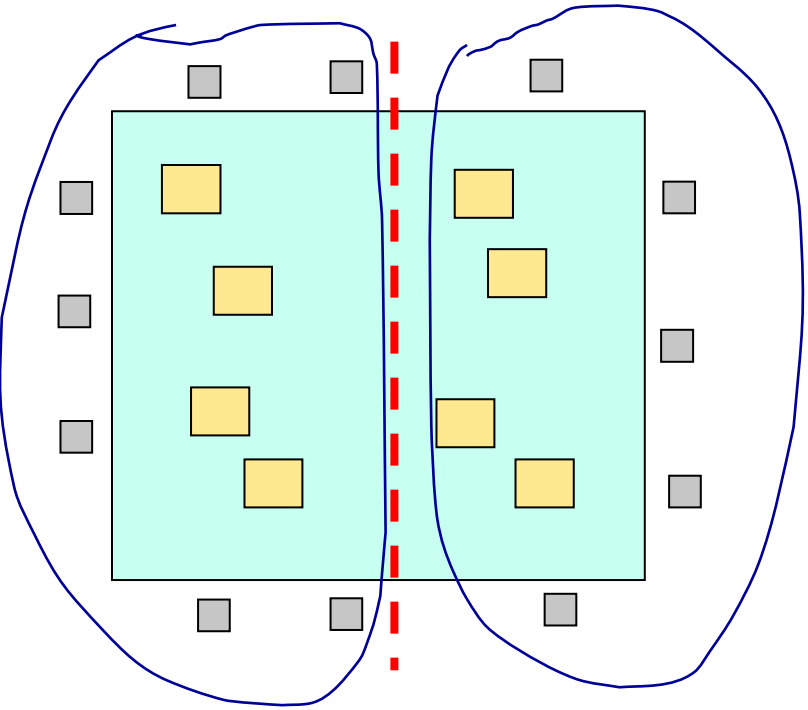
- What if there are some cells are fixed at certain locations?

Summary So Far

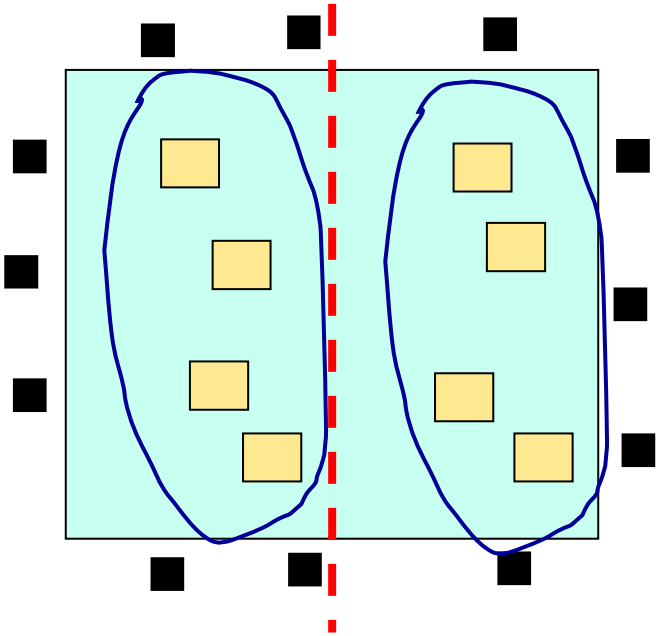
- Bi-partitioning
- Kernighan-and-Lin
- Multi-pin nets (hypergraph)
- Simulated annealing
- Clustering first to reduce the problem size

How To Apply Partitioning To Layout?

- All I/O pads free



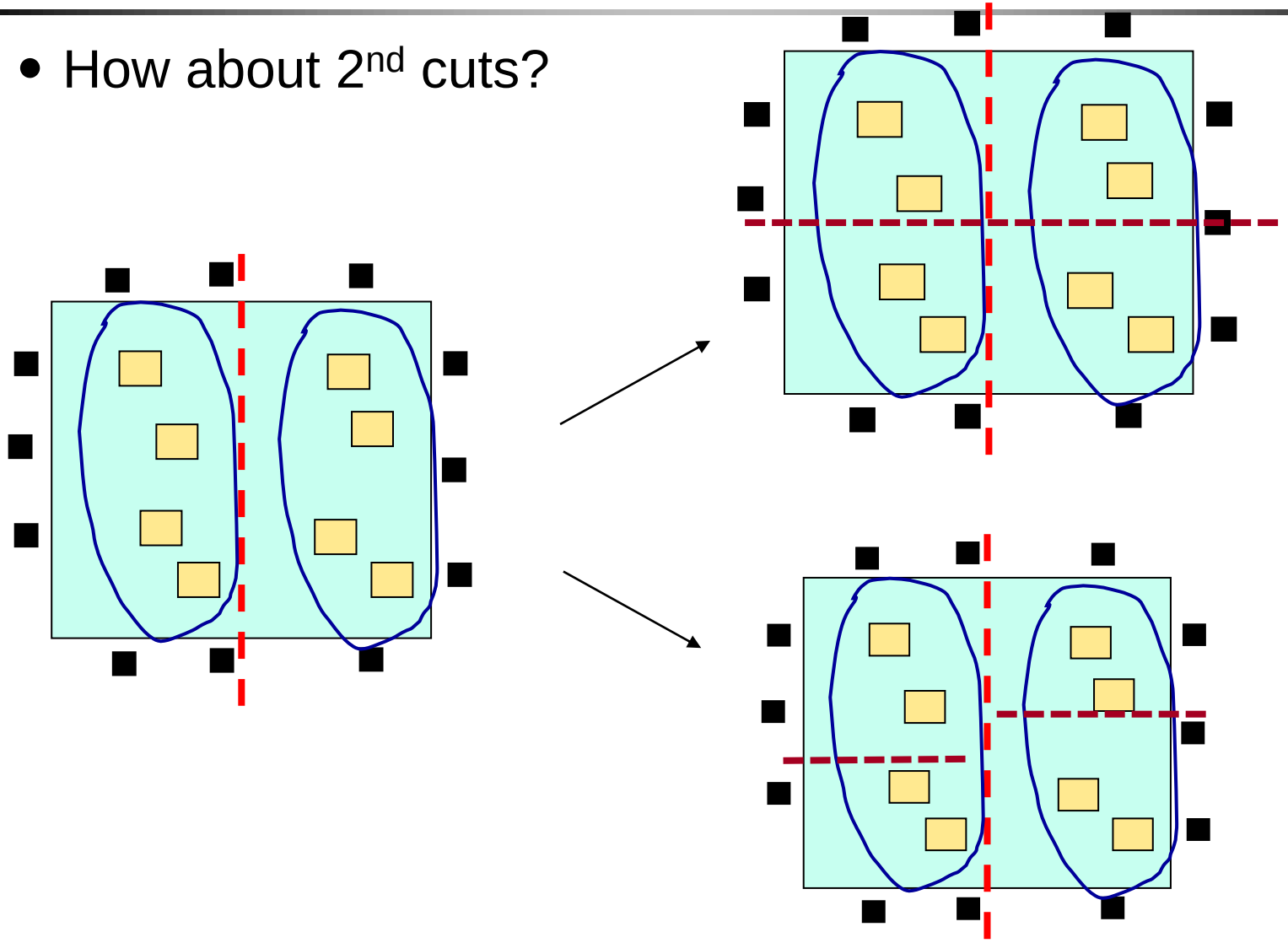
- All I/O pads fixed



Fixed nodes certainly will influence the partitioning. How?

How To Apply Partitioning To Layout? (2)

- How about 2nd cuts?



Ordering Of The Major Topics

- Digital design flow
- CMOS logic gates and Boolean equations
- Algorithms and complexity
- Basic logic synthesis/optimization, technology mapping
- Compaction
- Partitioning
- Floorplanning (Next)
- Placement
- Routing
- Clock tree synthesis
- High level synthesis
- Simulation