Partitioning for Physical Design

Prof. A. R. Newton

Prof. K. Keutzer

Michael Orshansky

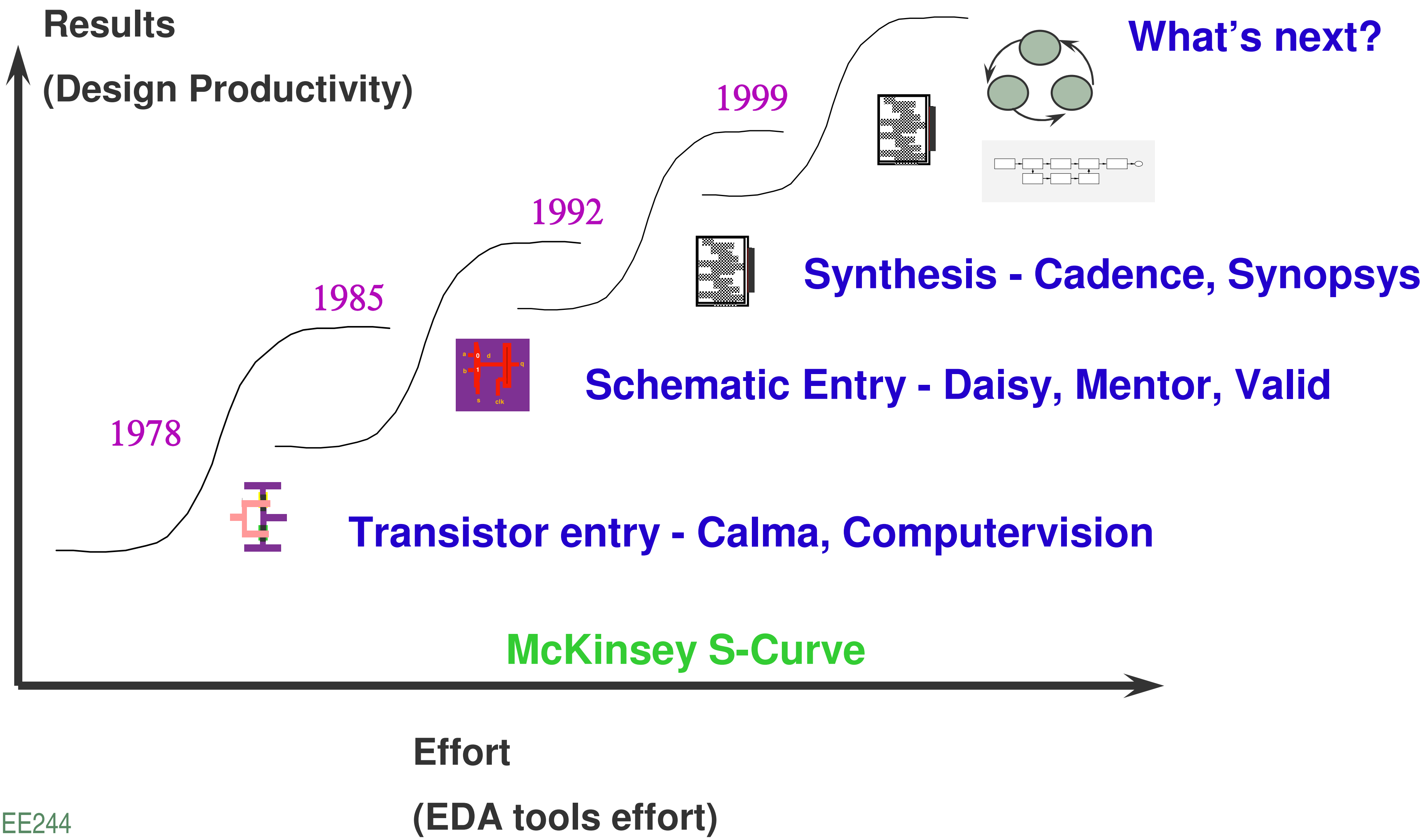
EECS

University of California

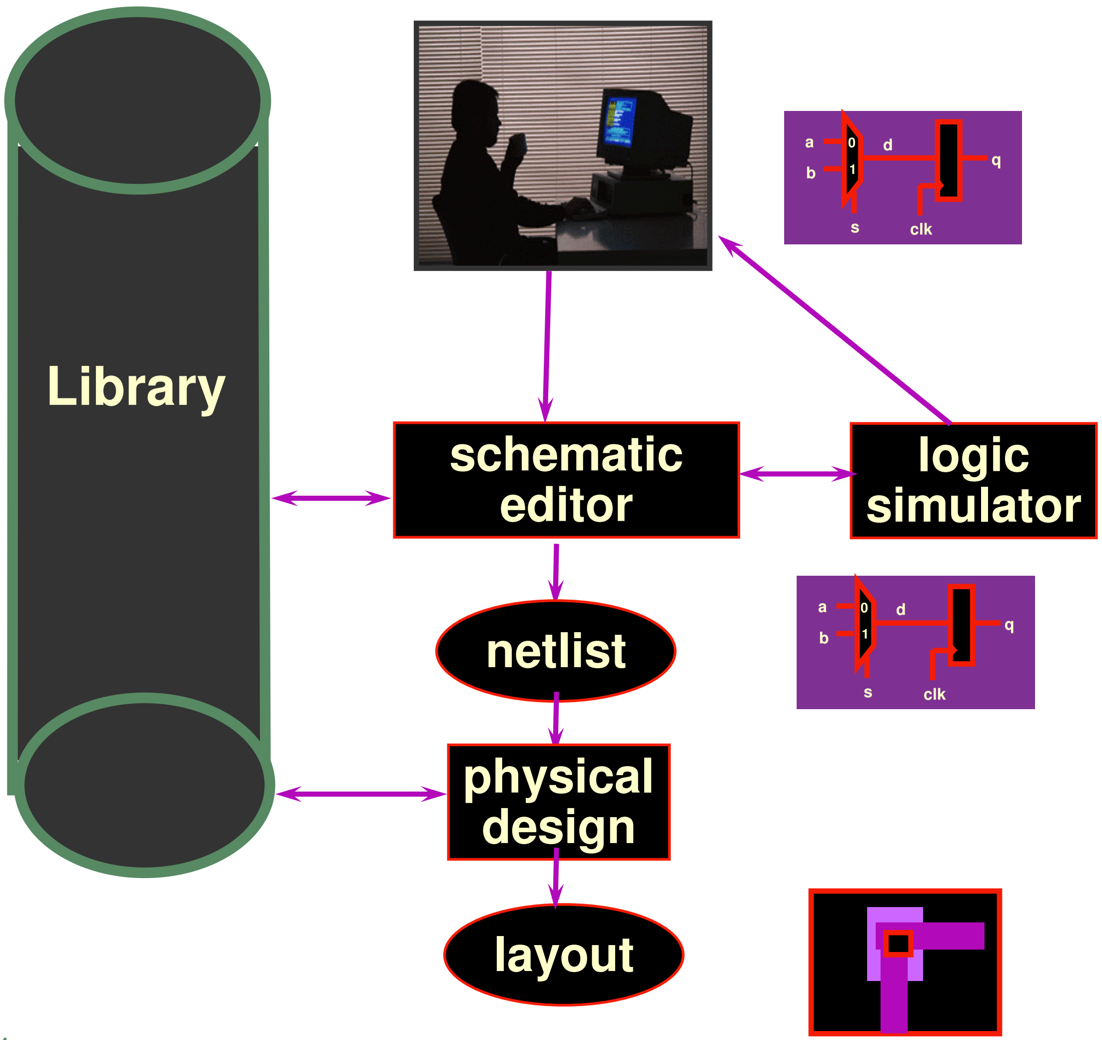
Berkeley, CA

With additional material from Andrew B. Kahng, UCSD, M. Sarrafzadeh, UCLA

Let's take a step back to the 1980's



Schematic Entry Design Flow



Designer designs the circuit on napkins and blackboard

Gate-level details of the circuit are entered in a schematic entry tool

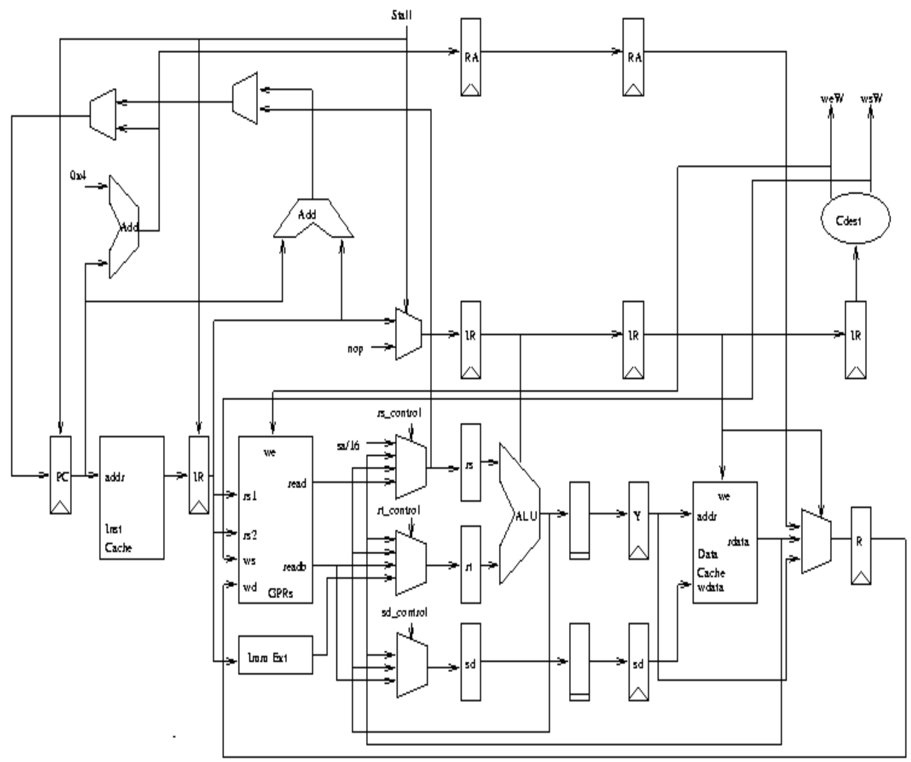
Vectors are generated to verify the circuit

When logic is correct the netlist is passed off to another group to lay out

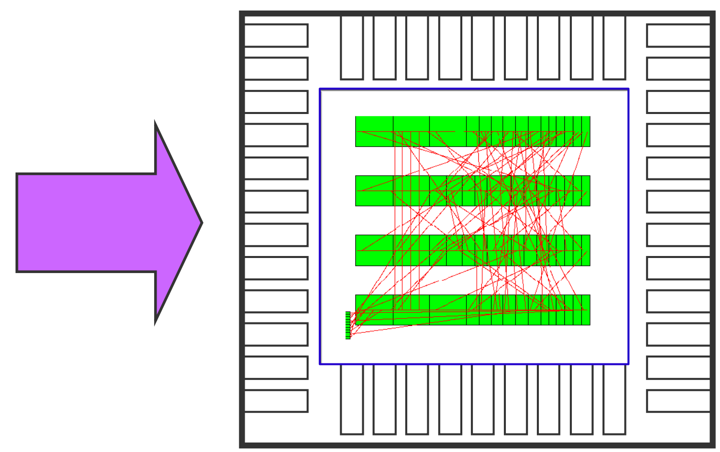
Automated place and route tools create layout

Basic Physical Design Problem

Schematic



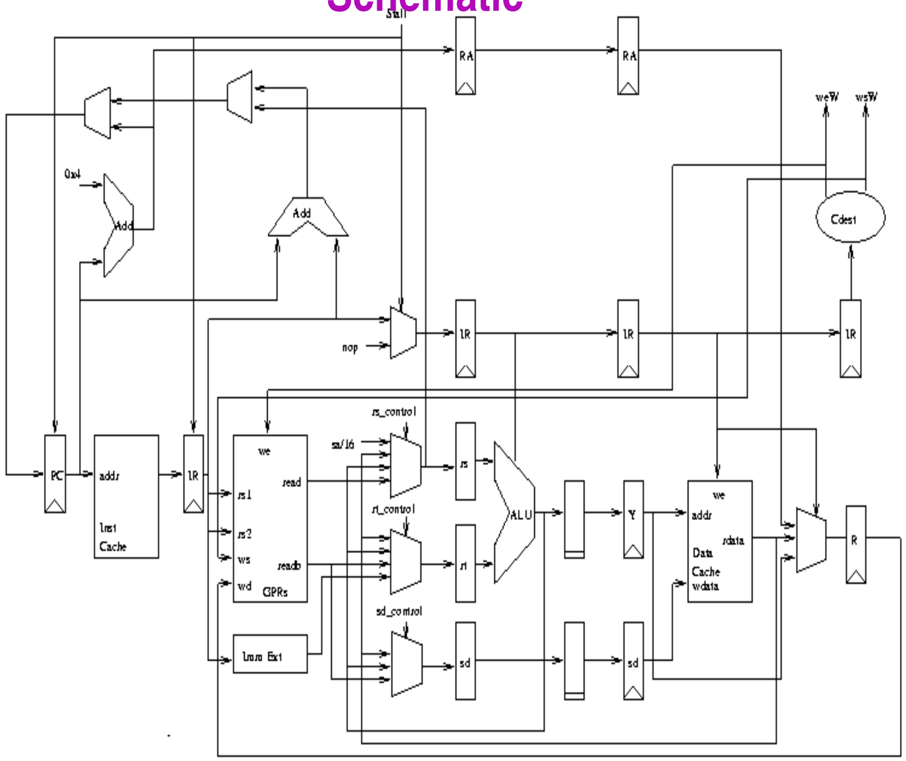
Layout



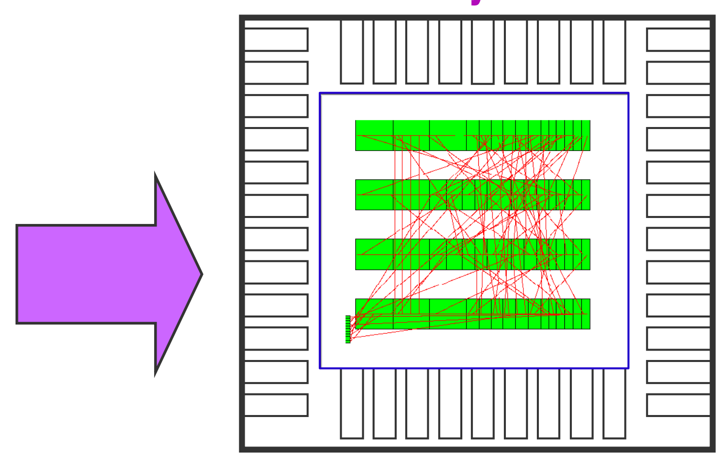
◆ What problems need to be solved in physical design?

Basic Physical Design Problem

Schematic

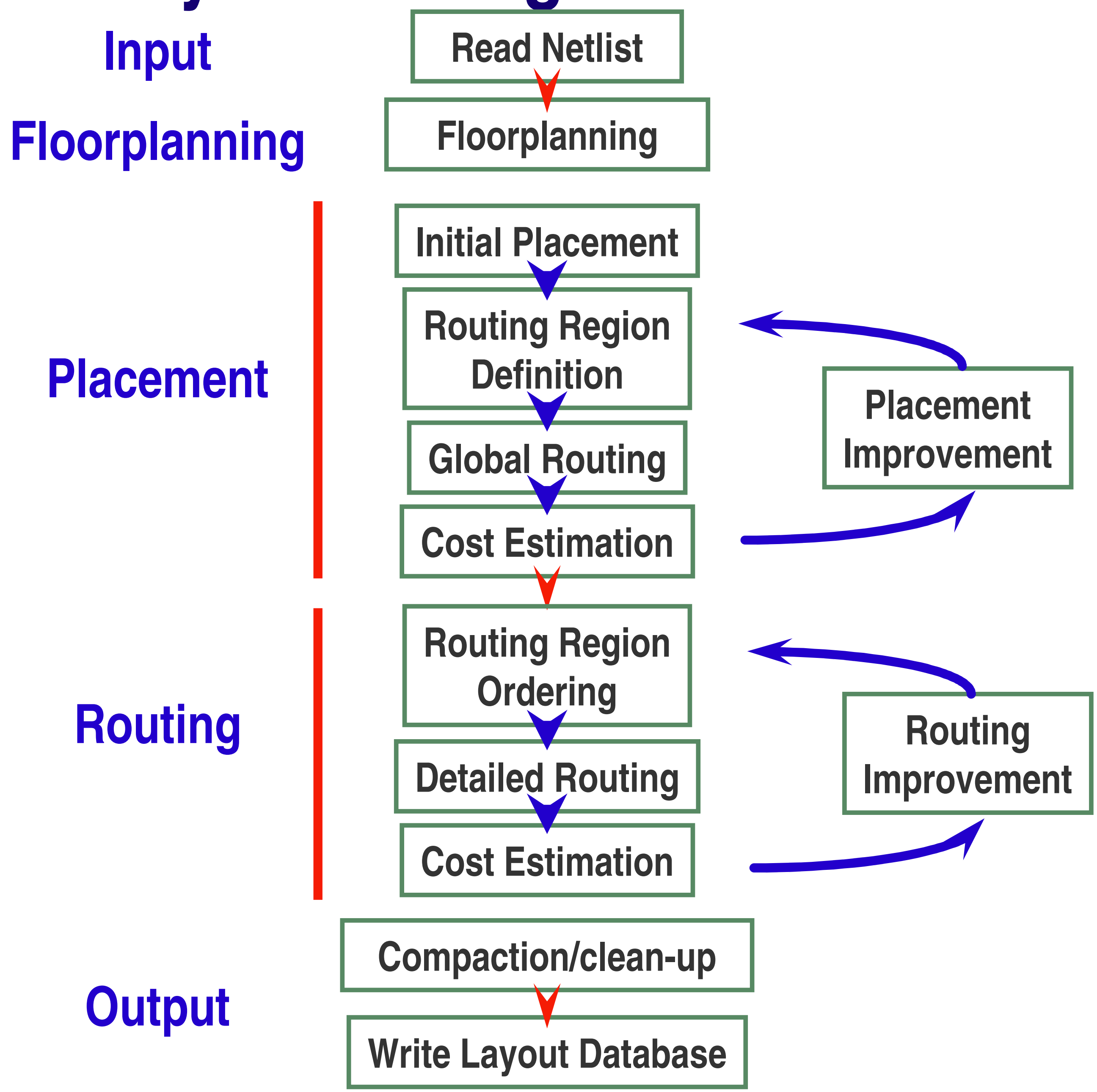


Layout



- What problems need to be solved in physical design?
  - Planarize graph → place gates/cells
  - Route wires to connect cells
  - Route clock
  - Route power and ground
  - Bond I/O's to I/O pads

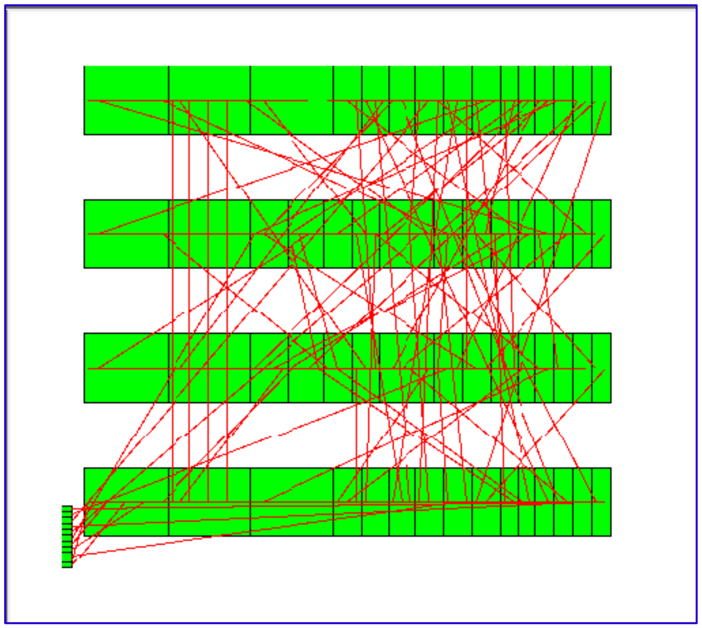
Physical Design: Overall Flow

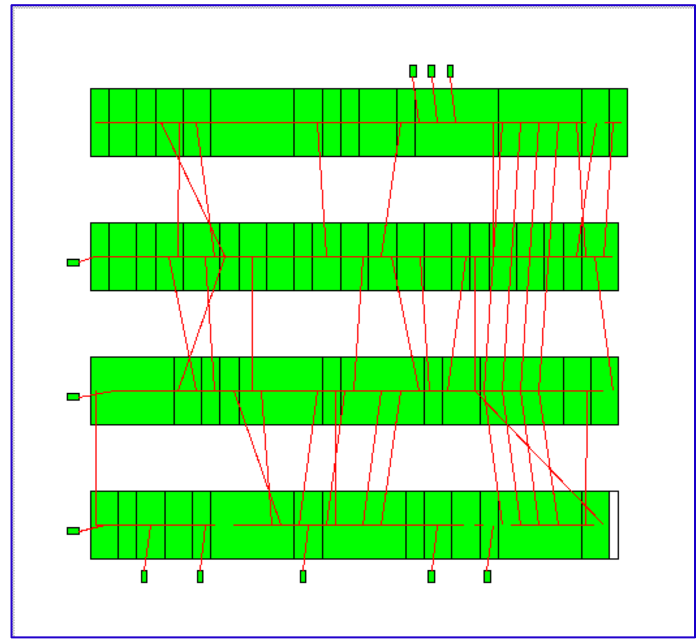


Formulation of the Placement Problem

- **Given:**
  - A netlist of cells from a pre-defined semiconductor library
  - A mathematical expression of that netlist as a vertex-, edge-weighted graph
  - Constraints on pin-locations expressed as constraints on vertex locations / aspect ratio that the placement needs to fit into
  - One or more of the following: chip-level timing constraints, a list of critical nets, chip-level power constraints
- **Find:**
  - Cell/vertex locations to minimize placement objective subject to constraints
- **Typical Objectives:**
  - minimal delay (fastest clock cycle time)
  - minimal area (least die area/cost
  - minimal power (static, dynamic)

Results of Placement





A bad placement

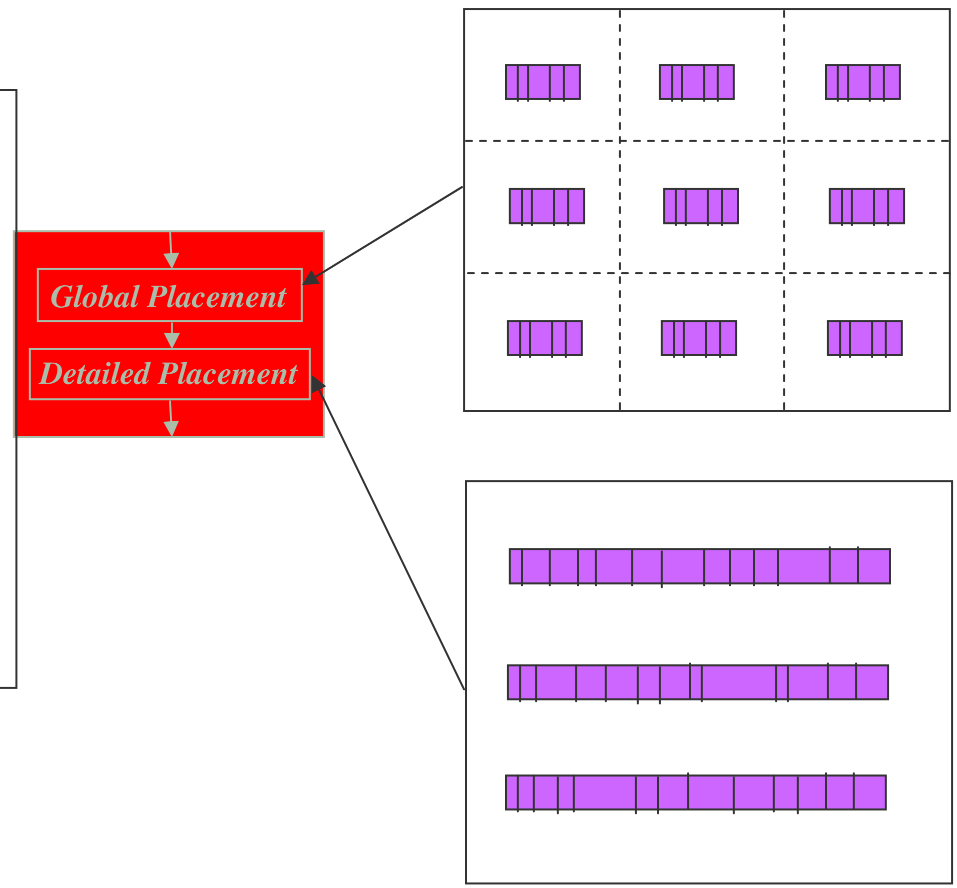
A good placement

A. Kahng

Global and Detailed Placement

In global placement, we decide the approximate locations for cells by placing cells in global bins.

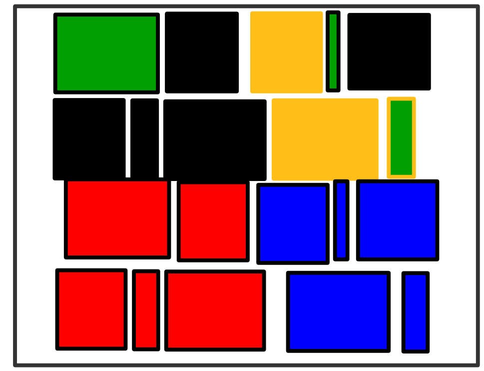
In detailed placement, we make some local adjustment to obtain the final non-overlapping placement.



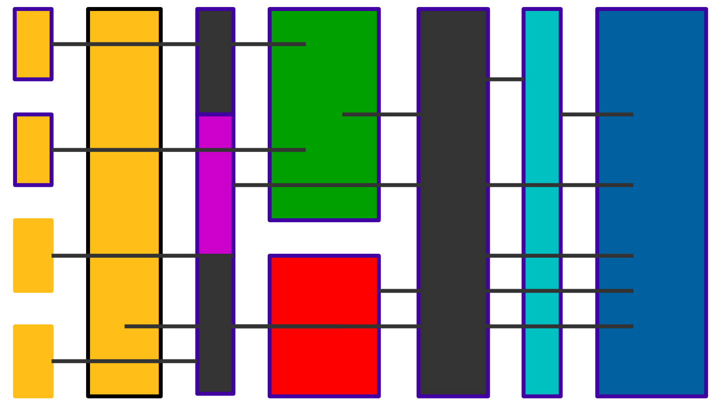
A. Kahng

Placement Footprints:

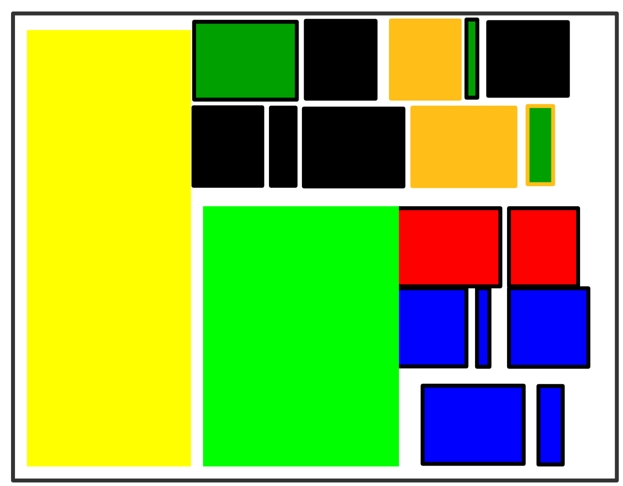
Standard Cell:



Data Path:



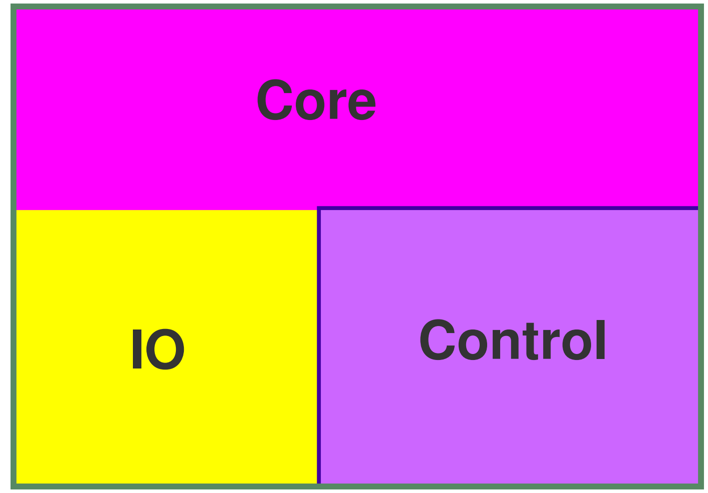
IP block - Floorplanning



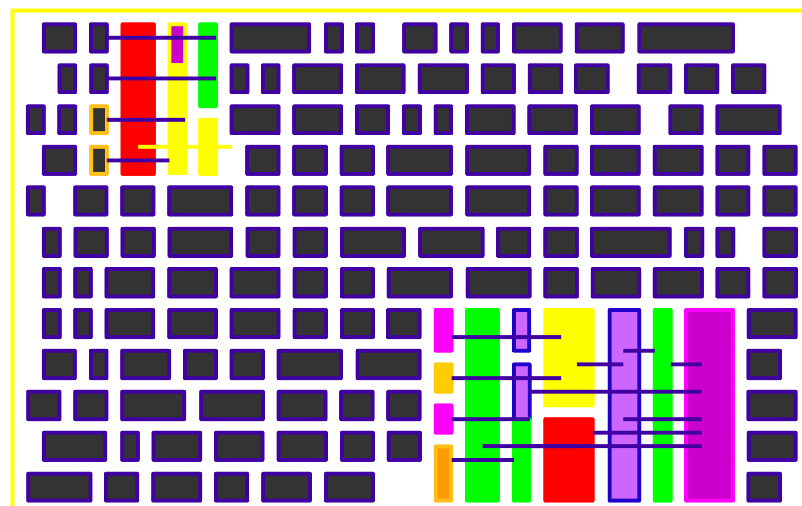
A. Kahng

Placement Footprints:

Reserved areas

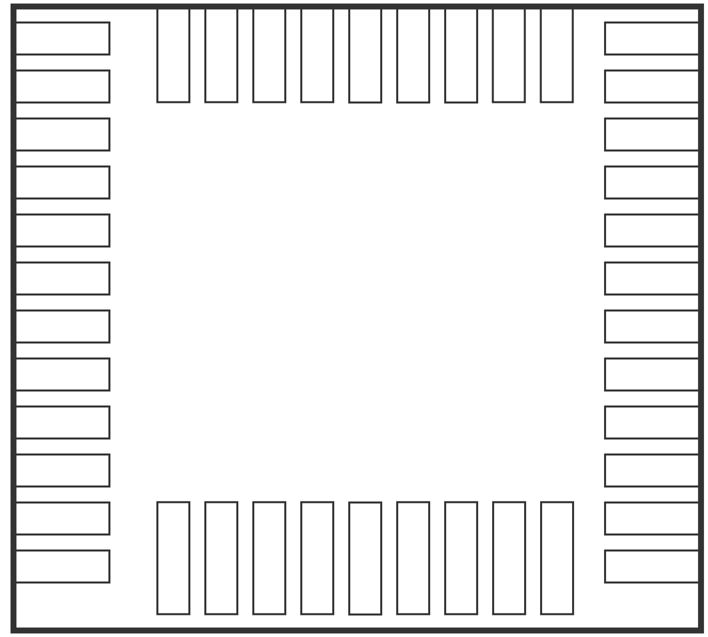


Mixed Data Path & sea of gates:

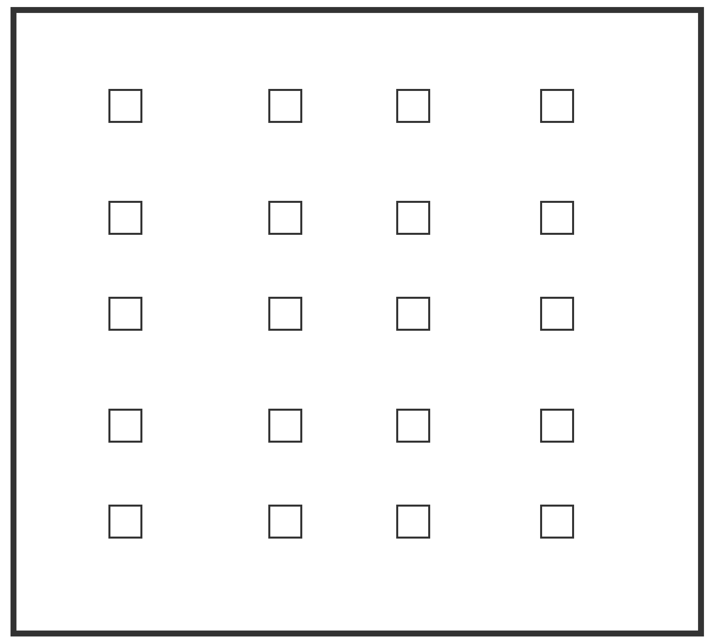


Placement Footprints:

Perimeter IO



Area IO – ball grid array



Approach to Placement: GORDIAN 1

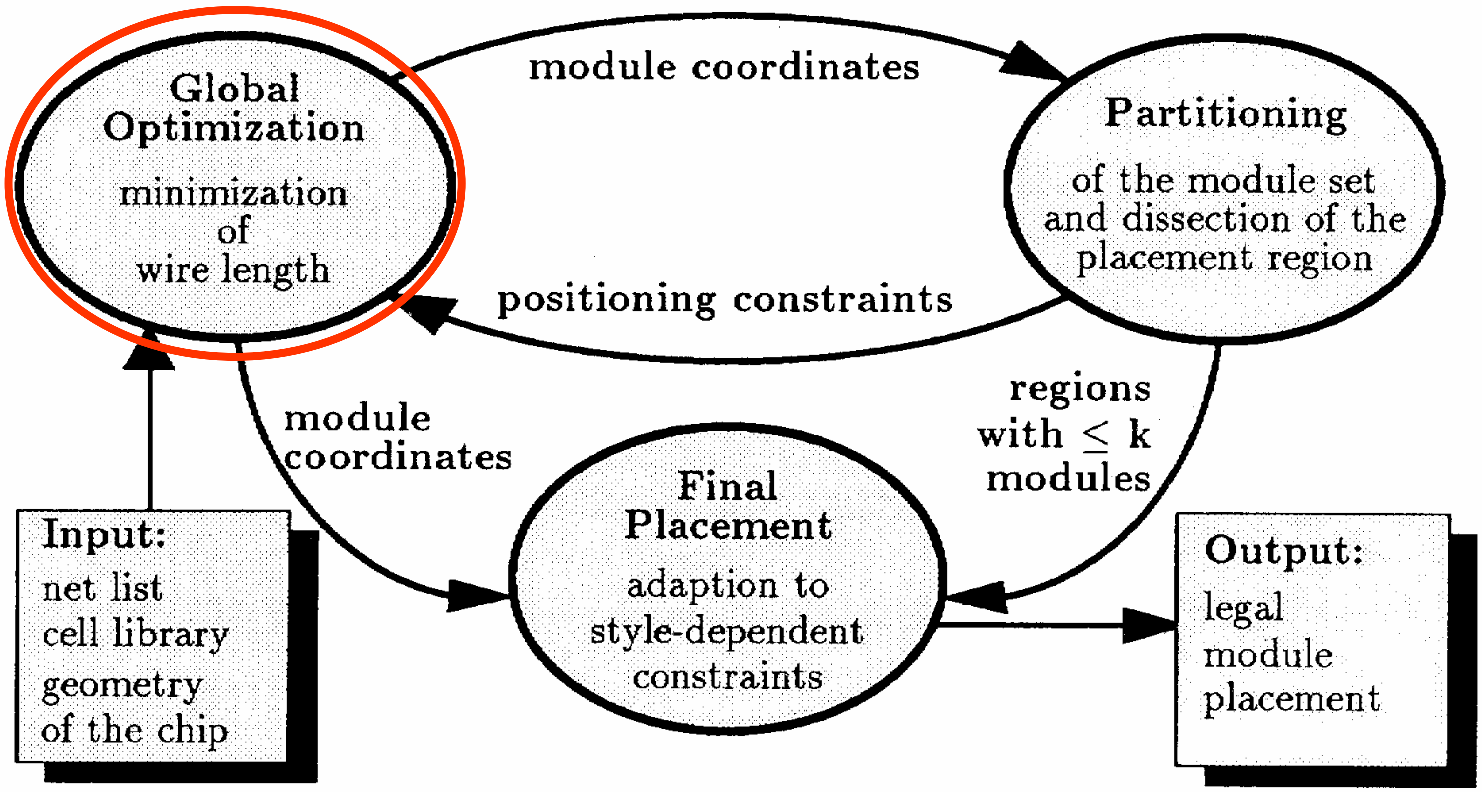
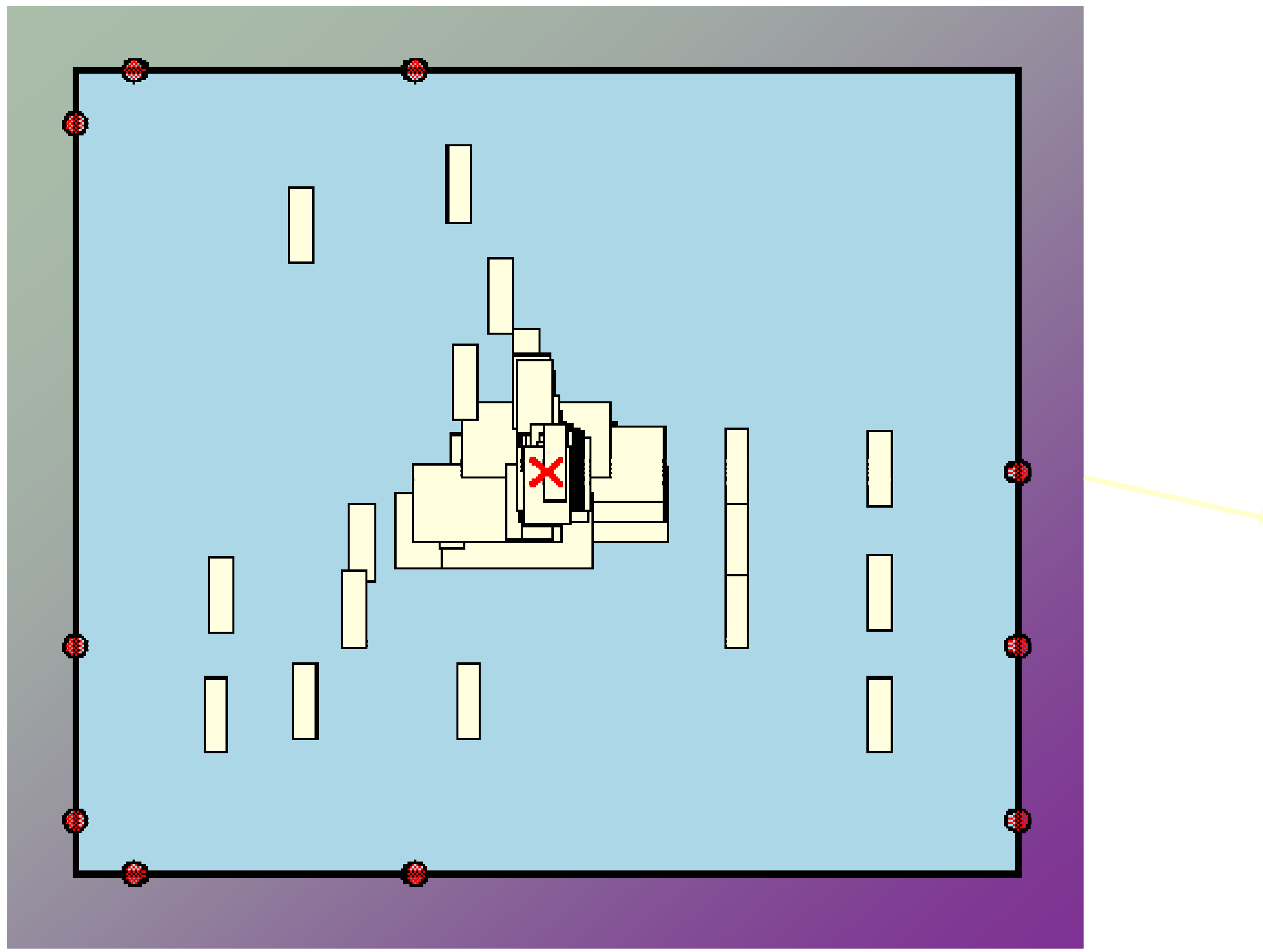


Fig. 1. Data flow in the placement procedure GORDIAN.

GORDIAN (quadratic + partitioning)

Initial
Placement



A. Kahng

Approach to Placement : GORDIAN 2

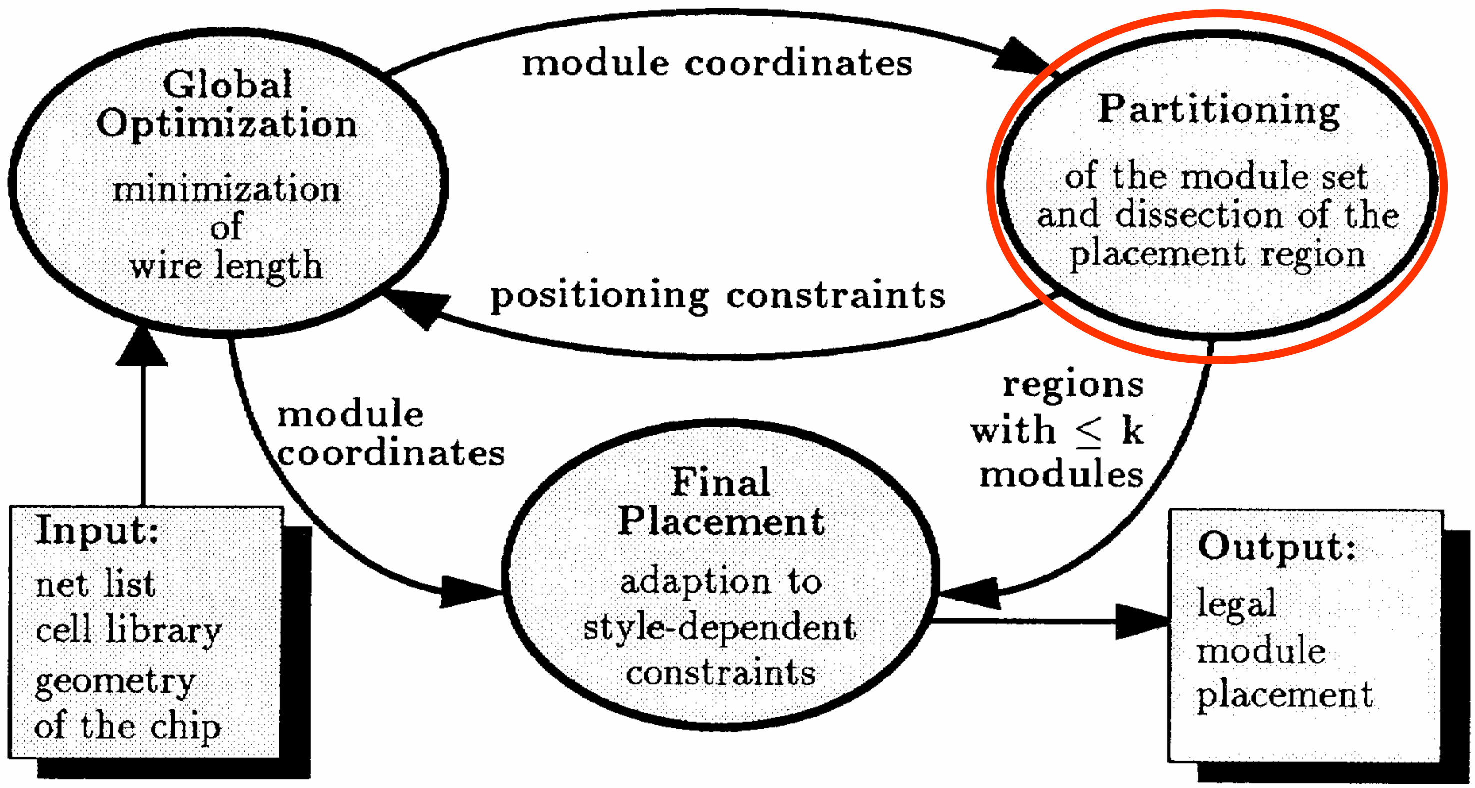
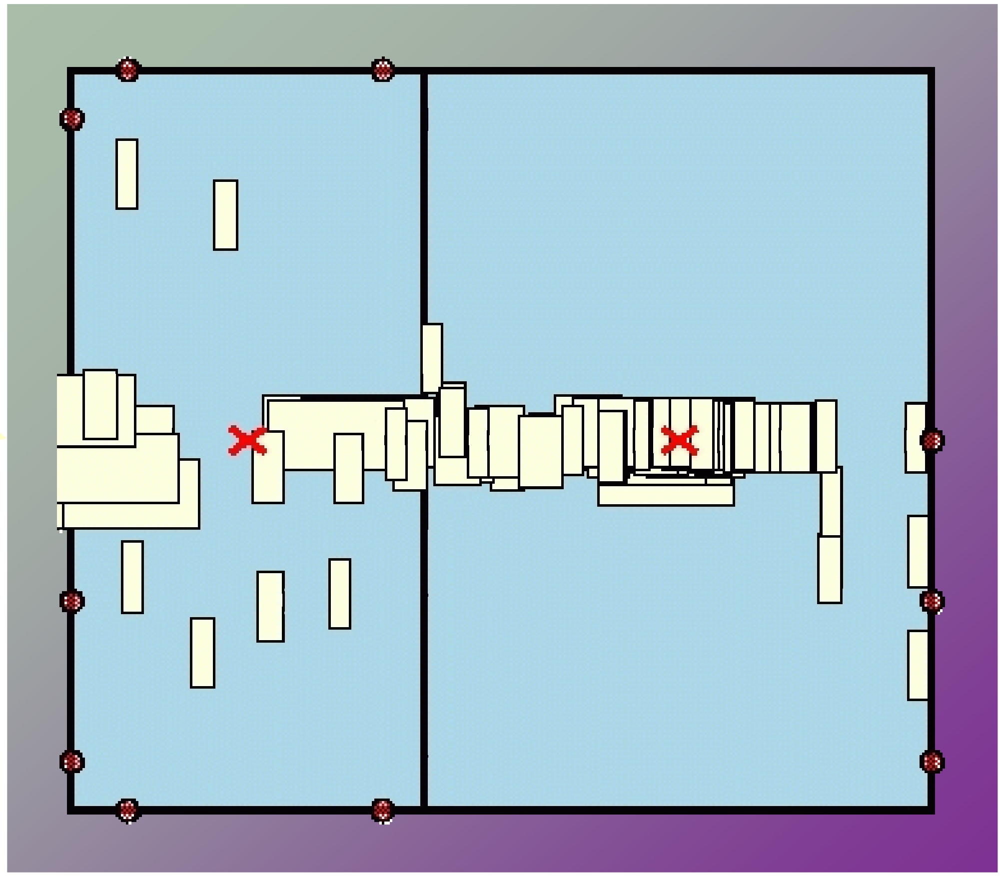


Fig. 1. Data flow in the placement procedure GORDIAN.

Partition in GORDIAN

Partition
and Replace

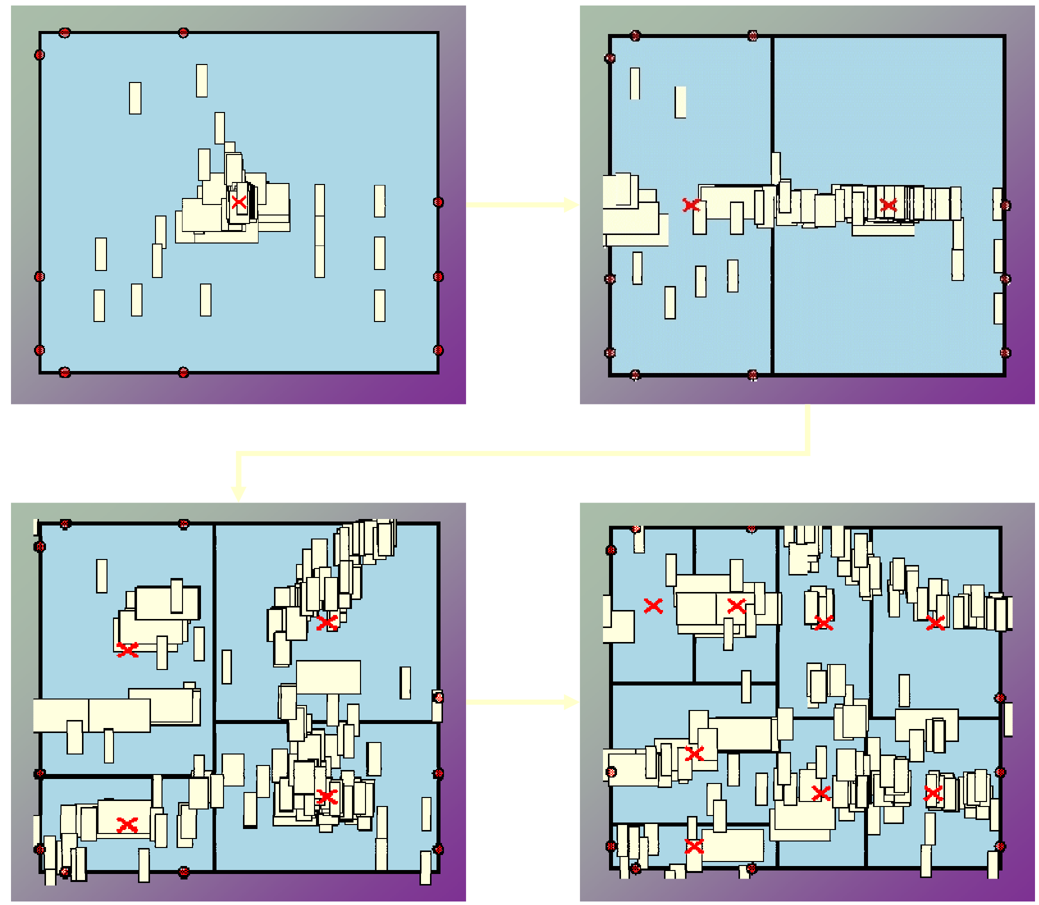


A. Kahng

GORDIAN (quadratic + partitioning)

Initial
Placement

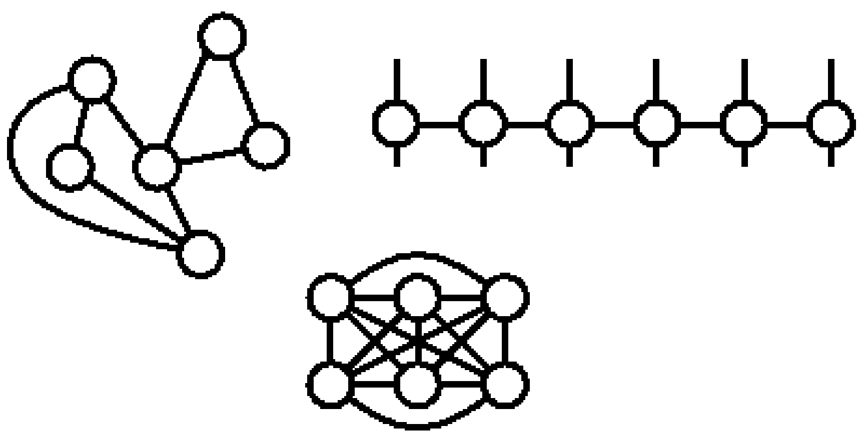
Partition
and Replace

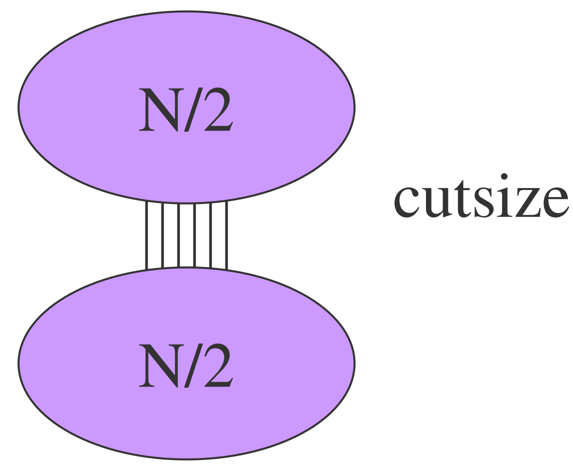


A. Kahng

Basic Idea of Partitioning

- Partition design into two (generally *N)* equal size halves
- Minimize wires (nets) with ends in both halves
- Number of wires crossing is bisection bandwidth
- lower bw = more locality





Netlist Partitioning: Motivation 1

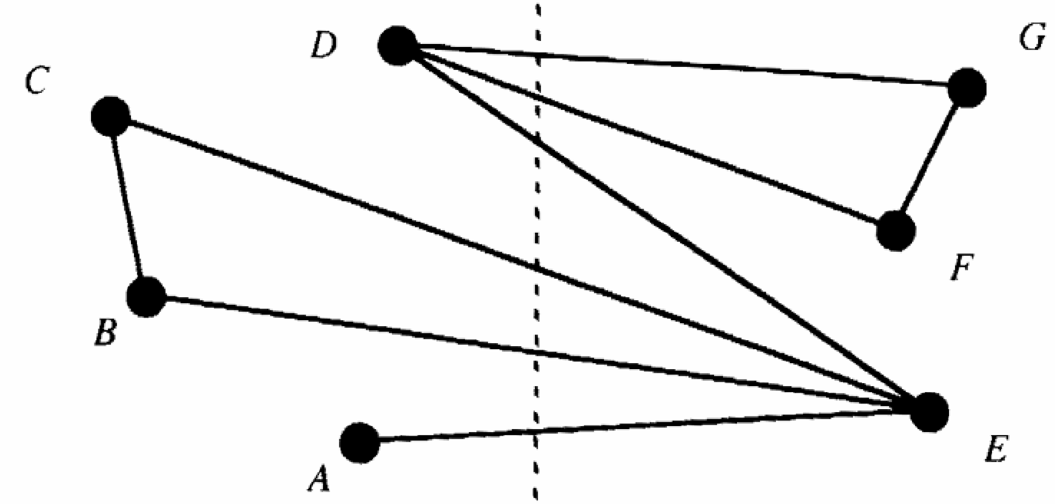
- Dividing a netlist into clusters to
  - Reduce problem size
  - Evolve toward a physical placement
- All top-down placement approaches utilize some underlying partitioning technique

- Influences the final quality of
  - Placement
  - Global routing
  - Detailed routing

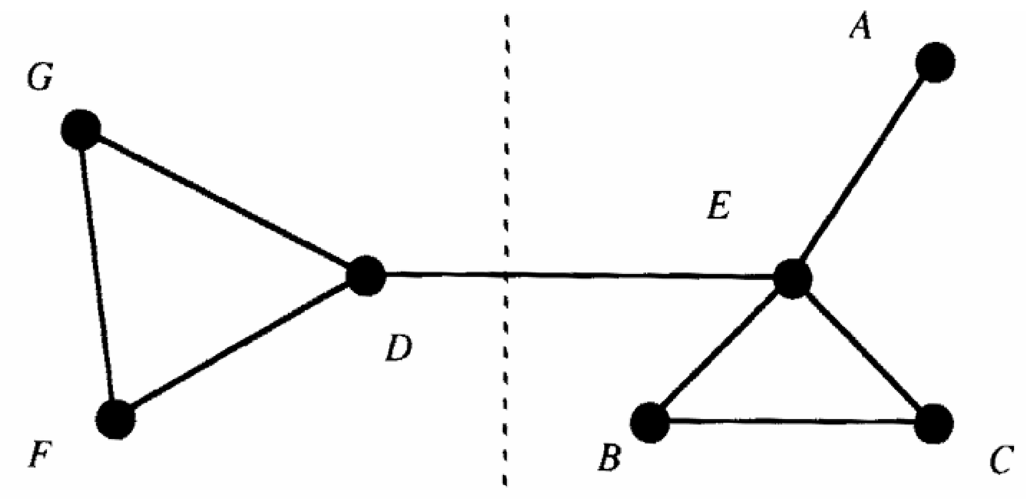
Netlist Partitioning: Motivation 2

- Becomes more critical with DSM
- System size increases
  - Need to minimize design coupling
- Interconnect dominates chip performance
  - Have to minimize number of block-to-block connections (e.g. global buses)
- Helps reduce chip area
  - Minimizes length of global wires

Partitioning for Minimum Cut-Set



(a) Original Partition (Random)

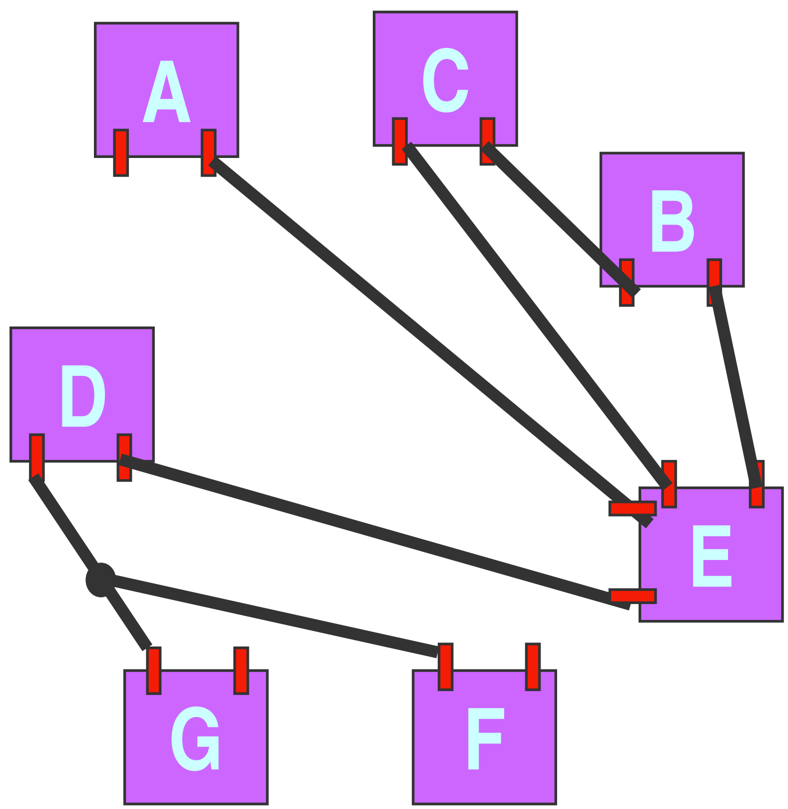


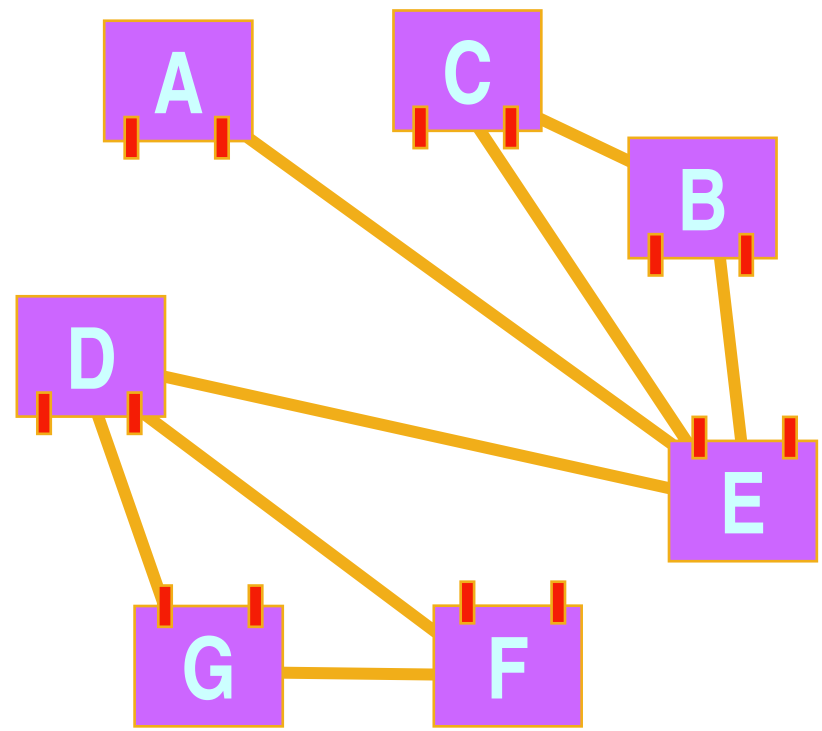
(b) Improved Partition

Graphs and Hypergraphs

- **A graph** $G = (V, E)$. **V - vertex set, E - edge set, a binary relationship on V.**
  $e_i = (v_{i1}, v_{i2})$. $|e_i| \equiv 2$.
- **In an undirected graph, the edge set consists of unordered pairs of vertices.**
- **In a hypergraph, H**(V, E), **a hyperedge** e **connects an arbitrary subset of vertices,**
  e.g. $|e_i| \geq 2$.
- **A circuit netlist is a hypergraph**

Netlist Partitioning

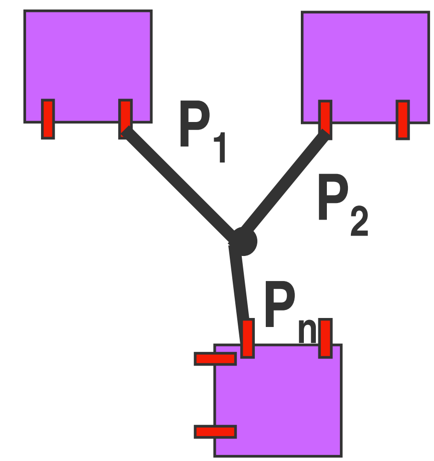




First problem transition from multi-terminal to two terminal edges

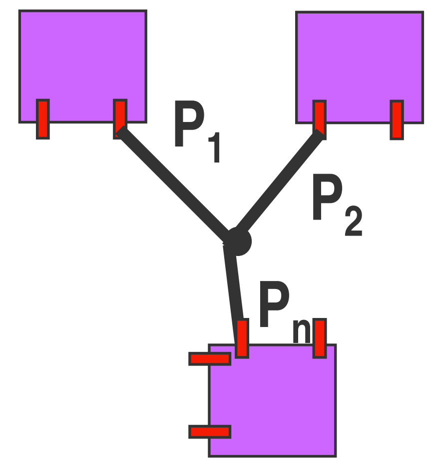
Edge Weights for Multiterminal Nets

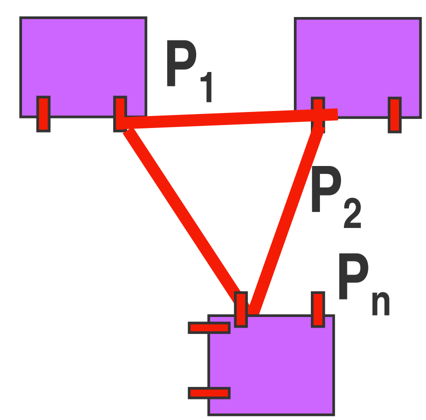
- Edges represent nets in the circuit netlist
- Each edge in the hypergraph will typically be given a weight which represents its criticality (cf. timing lecture)
- These weights will be used to “drive” partitioning, placement, and routing
- But if we want to use a graph structure, as opposed to a hypergraph, we must re-define the edges and their weights



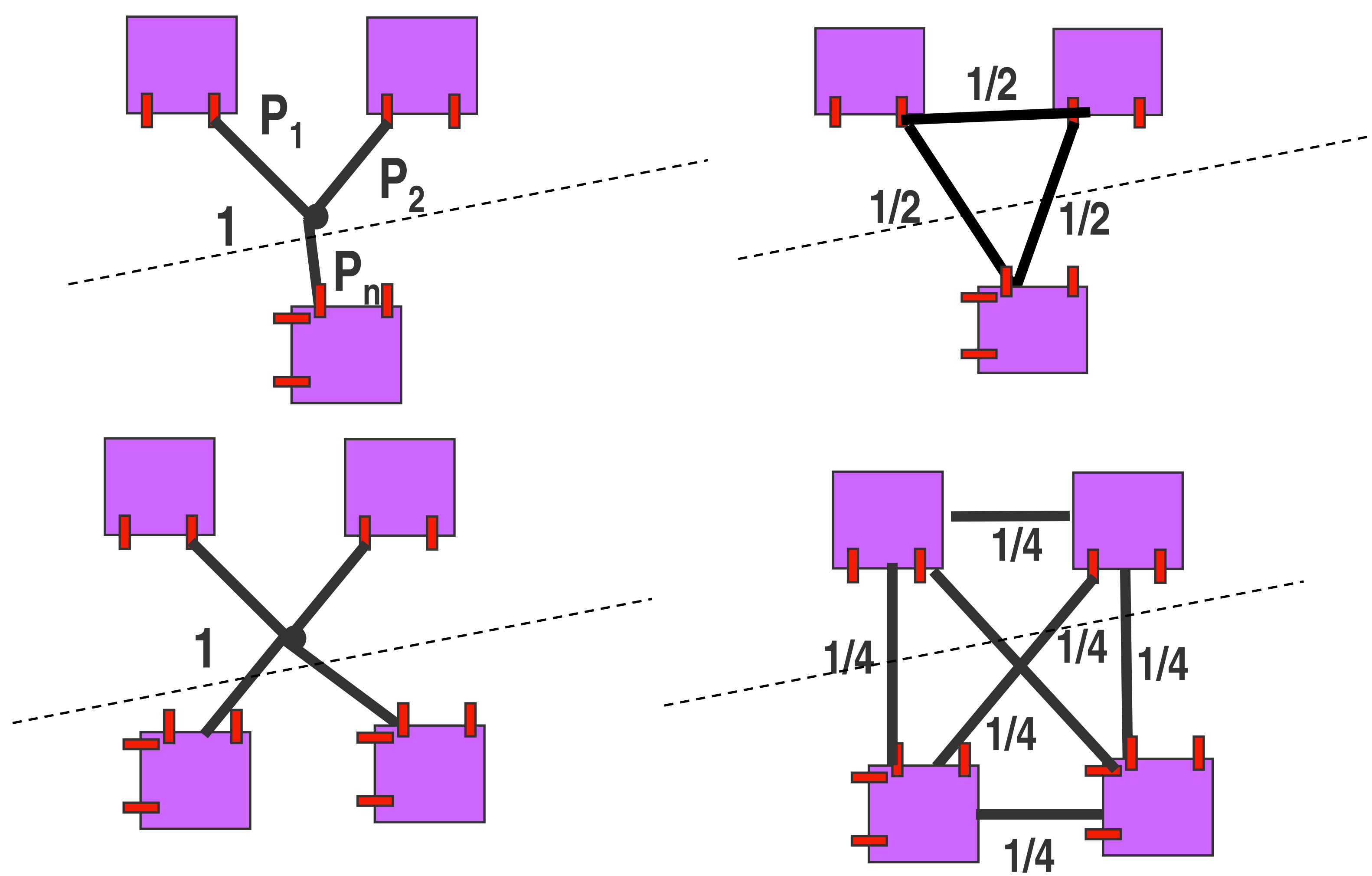
Edge Weights for Multiterminal Nets

- Replace each net S_i with its complete graph.
- What weight on each edge?
- One approach – assign weight of 1 to each net in the new graph
- Alternative: n-pin net, w=2/(n-1) has been used, also w=2/n
- "Standard" model: for n nets in the complete graph w=1/(n-1)
  - For any cut, cost >= 1
  - Large nets are less likely to be cut
  - Leads to highly sub-optimal partitions
  - Provides an *upper bound* on the cost of a cut in the actual netlist
- How about a *lower bound* on the cut cost?





Edge Weights for Multiterminal Nets



Another Weight Assignment for *Lower Bounding* the Net Cut

- Want to find a weight assignment that always underestimates net cuts
  - Gives a lower bound on the cost of the netlist cut
- Intuitively: choose weight assignment s.t max cost of a net cut in a graph is 1.
- Maximum cost happens when nodes are divided equally between 2 partitions
- The number of crossing edges in that situation (proof left to the reader ☺)

  $$(n^2\text{-}mod(n,2))/4$$

- Each edge is assigned the weight of

$$w = 4/(n^2\text{-}mod(n,2))$$

Example: for n=3, w=4/(9-1)=0.5

Partitioning

- Given a graph, *G*, with *n* nodes with sizes (weights) *w*:

  $$0 < w_i \le p, i = 1, \cdots, n$$

  with costs on its edges, partition the nodes of *G* into *k*, subsets, *k >0,* no larger than a given maximum size, *p*, so as to minimize the total cost of the edges cut.
- Define : $C = (c_{ij}), i, j = 1, \cdots, n$

  as a weighted connectivity matrix describing the edges of *G*.
- A *k-way partition* of *G* is a set of non-empty, pairwise-disjoint subsets of *G*, $v_1, \ldots, v_k$, such that $\bigcup_{i=1}^{k} v_i = G$
- A partition is said to be admissible if $|v_i| \le p, i = 1, \cdots, k$
- **Problem:** Find a minimal-cost permissible partition of *G*

How big is the search space?

- *n* nodes, *k* subsets of size *p* such that *kp=n*
- $\binom{n}{p}$ ways to choose the first subset
- $\binom{n-p}{p}$ ways to choose the second, etc.
- $\frac{1}{k!}\binom{n}{p}\binom{n-p}{p}\cdots\binom{2p}{p}\binom{p}{p}$ ways total

- *n*=40, *p*=10 $> 10^{20}$
- In general, solving problems where $T_n \propto n^{\beta}, \beta > 2$ are impractical for real circuits (>1,000,000 gates)

Heuristics for n-Way Partitioning

- Hard problem and no really good heuristics for $n>2$
  - **Direct Methods:** Start with seed node for each partition and assign nodes to each partition using some criterion (e.g. sum of weighted connections into partition)
  - **Group Migration Methods:** Start with (random) initial partition and migrate nodes among partitions via some heuristic
  - **Metric Allocation Methods**: uses metrics other than connection graph and then clusters nodes based on metric other than explicit connectivity.
  - **Stochastic Optimization Approaches:** Use a general-purpose stochastic approach like simulated annealing or genetic algorithms
- Usually apply two-way partitioning (Kernighan-Lin or Fiduccia-Matheyses) recursively, or in some cases simulated annealing

Partitioning: Random plus Improvement

- Random Partitions, Save Best to Date
  - Fast, but can be shown to be $O(n^2)$
  - Few optimal or near optimal solutions, hence low probability of finding one

e.g. 2-way partition of 0-1 weight graphs with 32 nodes, ~3-5 optimal partitions out of $\frac{1}{2}\binom{32}{16} \Rightarrow P(success) \text{ on any trial } < 10^{-7}$

Partitioning: Max-flow, Min-cut

- Max-flow, Min-cut: useful for unconstrained lower bound
  - Ford & Fulkerson, "Flows in Networks," Princeton Univ. Press, 1962
  - Edge weights of G correspond to maximum flow capacities between pairs of nodes
  - Cut is a separation of nodes into two disjoint subsets; cut capacity is the cost of a partition

Max-flow Min-cut Theorem: The maximum flow between any pair of nodes = the minimum cut capacity of all cuts which separate the two nodes

Computing max-flow through graph is probably too expensive

Two-Way Partitioning (Kernighan & Lin)

- Consider the set *S* of *2n* vertices, all of equal size for now, with an associated cost matrix $C = (c_{ij}), i, j = 1, \cdots, 2n$
- Assume *C* is symmetric and $c_{ii} = 0 \forall i$
- We want to partition *S* into two subsets *A* and *B*, each with *n* points, such that the external cost $T = \sum_{A \times B} C_{ab}$ is minimized
- Start with any arbitrary partition [*A*,*B*] of *S* and try to decrease the initial cost *T* by a series of interchanges of subsets of *A* and *B*
- When no further improvement is possible, the resulting partition [*A'*,*B'*] is a *local minimum* (and has some probability of being a *global minimum* with this scheme)
- (Be sure to take a moment to talk about local and global minima)

Kernighan & Lin: Value of a configuration

- For each vertex *a* in partition *A*: $a \in A$
  - external cost $E_a = \sum_{y \in B} c_{ay}$ (computed the same for ***E**b*)
  - internal cost $I_a = \sum_{x \in A} c_{ax}$ (computed the same for ***I**b*)
- For each vertex *z* in the set *S*, the difference (*D*) *between* external (*E*) *and* internal (*I*) costs is given by*:*

$$D_z = E_z - I_z \forall z \in S$$

Kernighan & Lin: Value of one swap

- For each $a \in A$:
  - external cost $E_a = \sum_{y \in B} c_{ay}$ (same for *Eb*)
  - internal cost $I_a = \sum_{x \in A} c_{ax}$ (same for *Ib*)

  $D_z = E_z - I_z \forall z \in S$
- If *a* $\in A$ and *b* $\in B$ are interchanged, then the gain:

  $g = D_a + D_b - 2c_{ab}$
- Proof: If *Z* is the total cost of connections between partitions *A* and *B*, excluding vertices *a* and *b*, then:

$$\left.\begin{aligned} T_{a,b} &= Z + E_a + E_b - c_{ab} \\ T_{b,a} &= Z + I_a + I_b + c_{ab} \end{aligned}\right\} gain = T_{a,b} - T_{b,a} = D_a + D_b - 2c_{ab}$$

Kernighan & Lin: Choosing swap

(1) Compute all *D* values in *S*

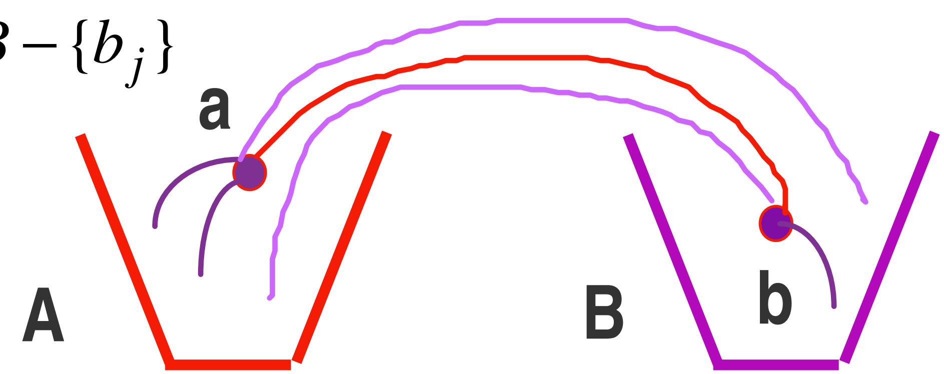
(2) Choose a_i, b_i such that $g_i = D_{a_i} + D_{b_j} - 2c_{a_i b_j}$ is maximized

(3) Set a_i and b_i aside and call them a_i' and b_i'

(4) Recalculate the D values for all the elements of $A - \{a_i\}, B - \{b_j\}$

$$D_x' = D_x + 2c_{xa_i} - 2c_{xb_j}, x \in A - \{a_i\}$$

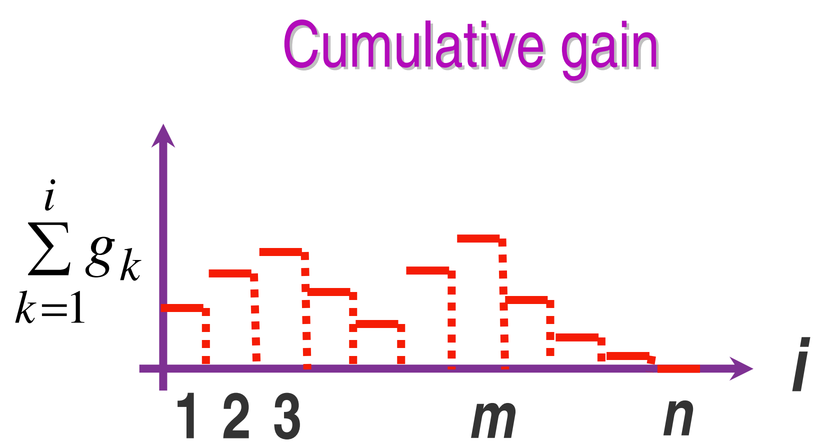
$$D_y' = D_y + 2c_{yb_j} - 2c_{ya_i}, y \in B - \{b_j\}$$



Kernighan & Lin: Partitioning Algorithm

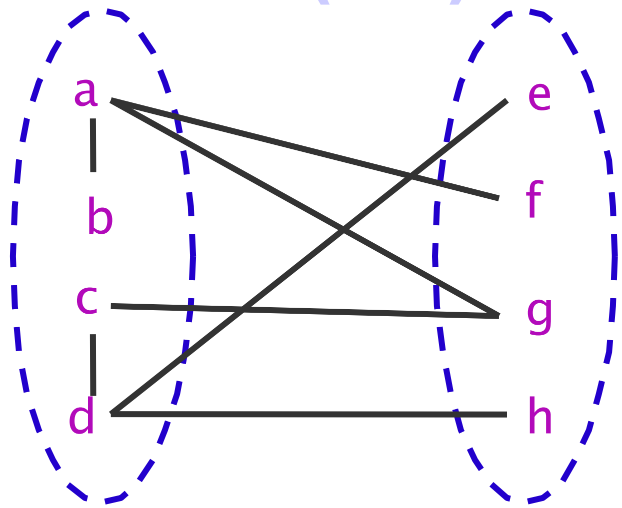
```
Algorithm KL(G, graph of 2N nodes)
Initialize  - create initial bi-partition into A, B each of N nodes
/* Compute global value of individual swaps of nodes */
Repeat until no further improvement{
    for I = 1 to N do           {
            find pair of unlocked nodes ai in A and bi in B whose exchange
            leads to largest decrease or smallest increase in cost
            cost_i = change in cost due to exchanging ai  and bi
            lock down ai  and bi  so they don't participate in future moves
    }
/* find which sequence of swaps gave the best result */
    find l  such that sum of cost(1<=l) is maximized
    move ai 0<=l from A to B
    move bi 0<=l from B to A
}
```

Two-Way Partitioning (Kernighan & Lin)



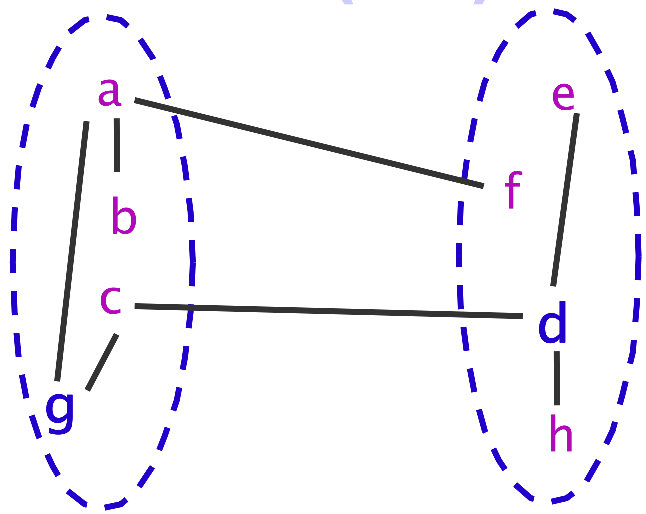
- Find point (exchange) *m* at which *cumulative* gain maximized
- Perform exchanges 1 through *m*
- What is the time and memory complexity of this algorithm?

Kernighan-Lin (KL) Example - 1



Step No.	Vertex Pair	Gain	Cut-cost
0	--	0	5

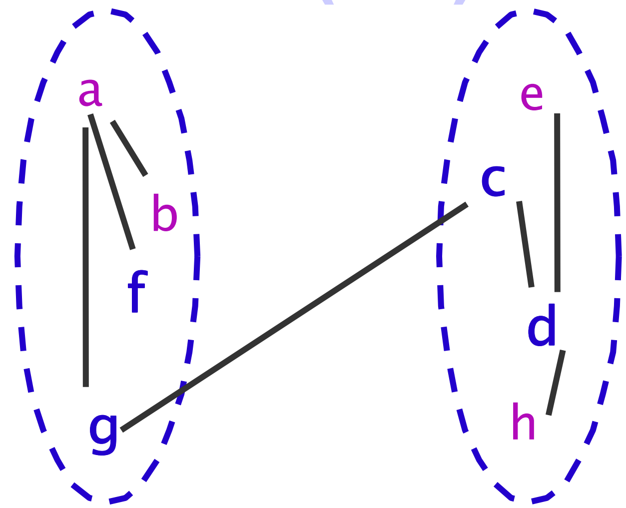
Kernighan-Lin (KL) Example - 2



Step No.	Vertex Pair	Gain	Cut-cost
0	--	0	5
1	{ d, g }	3	2

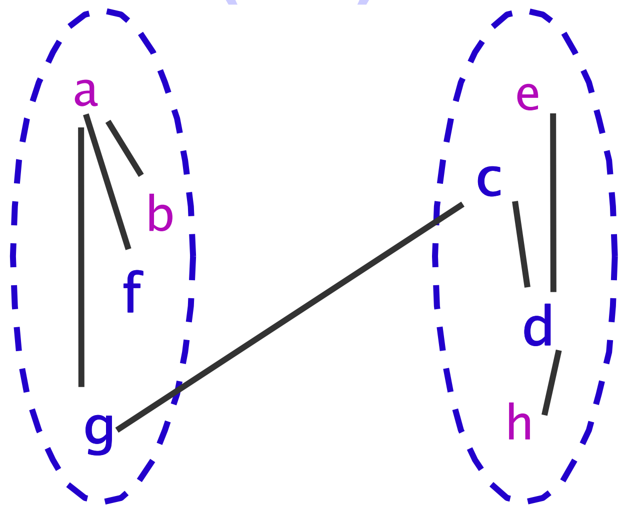
[©Sarrafzadeh]

Kernighan-Lin (KL) Example - 3



Step No.	Vertex Pair	Gain	Cut-cost
0	--	0	5
1	{ d, g }	3	2
2	{ c, f }	1	1

Kernighan-Lin (KL) Example - finish



Step No.	Vertex Pair	Gain	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

Time Complexity of K-L Partitioning

- A pass is a set of operations needed to find exchange sets
- Initial difference vector D computation is n^2
- Update of D after locking a pair (we lock down one more each pass)
  - $(n-1)+(n-2)+\ldots+2+1 \rightarrow n^2$
- Dominant time factor – selection of the next pair to exchange
  - Need to sort D values
  - Sorting is $n*\log(n)$
  - $(n)\log(n)+(n-1)\log(n-1)+(n-2)+\ldots+2\log2 \rightarrow n^2\log n$
- Total time is $n^2 \log n$

Just what does partitioning do?

- Reduces the problem size enabling a “divide and conquer” approach to problem solving
- Naturally evolves the netlist toward a full placement

Where does partitioning fit in?

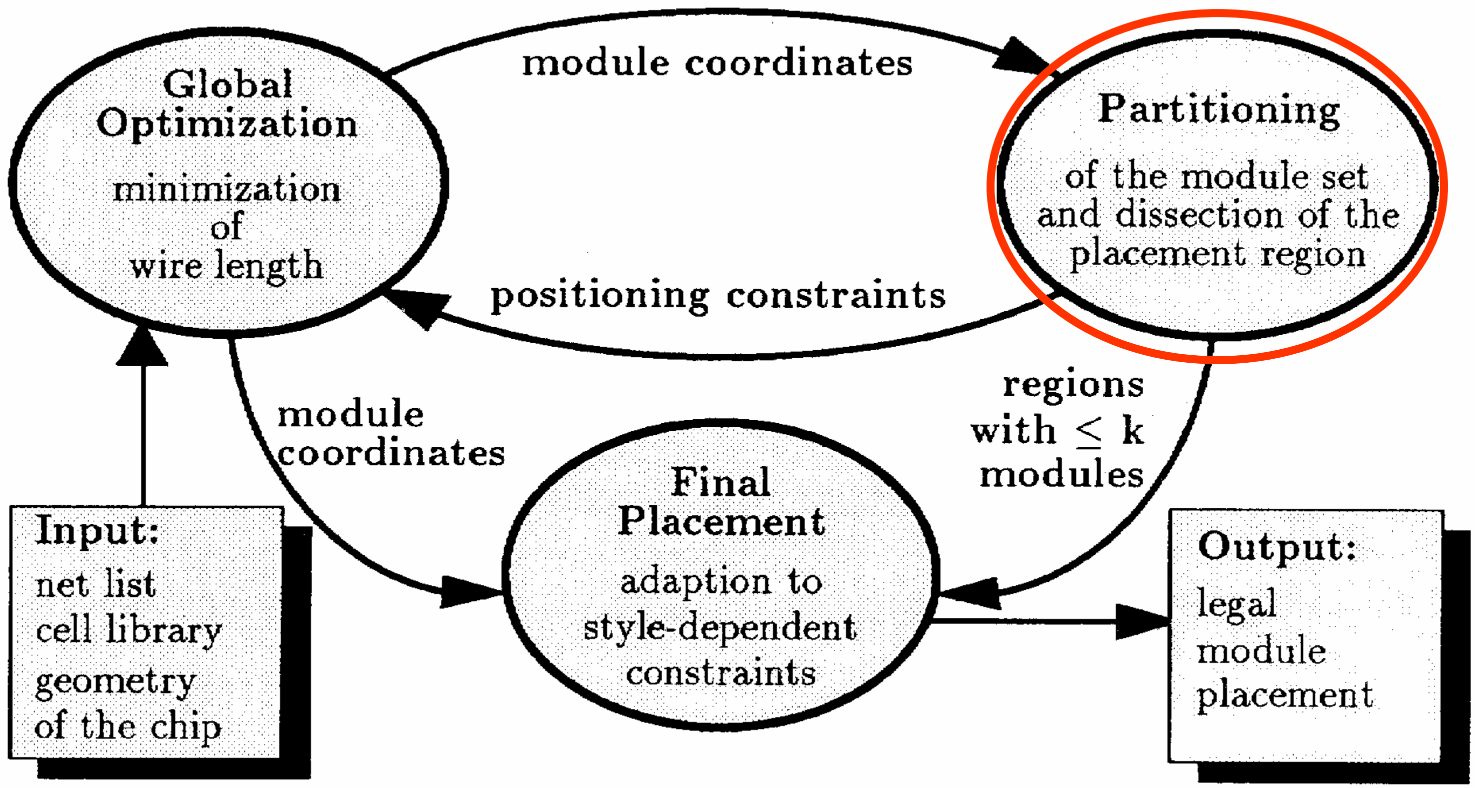


Fig. 1. Data flow in the placement procedure GORDIAN.

Partitioning

- **In GORDIAN, partitioning is used to constraint the movement of modules rather than reduce problem size**

- **By performing partitioning, we can iteratively impose a new set of constraints on the global optimization problem**
  - **Assign modules to a particular block**

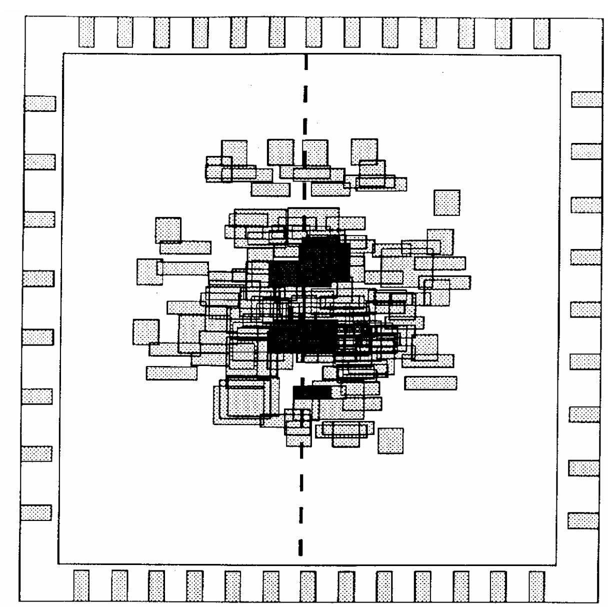
- **Partitioning is determined by**
  - **Results of global placement**
    - Spatial (x,y) distribution of modules
  - **Partitioning cost**
    - Want a min-cut partition

Partitioning due to Global Optimization

- Sort the modules by their x coordinate (for a vertical cut)

$$M_p \rightarrow M_{p'},\ M_{p''}$$

- Choose a cut line such that



$$x_{u'} \le x_{u''} \quad u' \in M_{p'},\ u'' \in M_{p''}$$

$$\alpha = \sum_{u' \in M_{p'}} F_u \Big/ \sum_{u \in M_p} F_u \approx 0.5$$

Partitioning Improvement - I

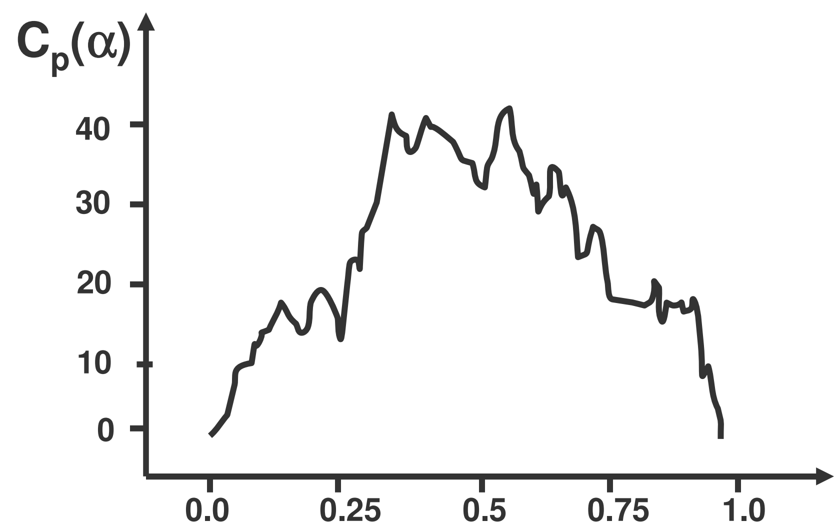
- The cost of initial partition may be too high
- Can change position of the cut to reduce the cost
- Plot the cost function, choose “best” position

$$M_p \rightarrow M_{p'}, M_{p''}$$

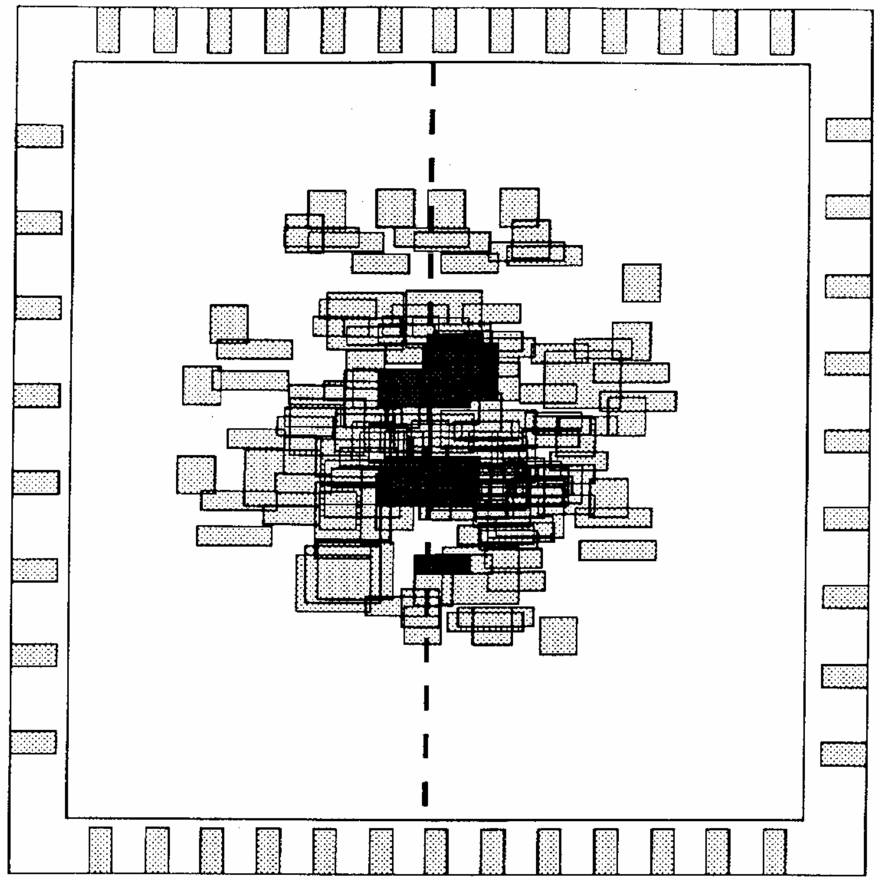
$$x_{u'} \leq x_{u''} \quad u' \in M_{p'}, u'' \in M_{p''}$$

$$\alpha = \sum_{u' \in M_{p'}} F_u \Big/ \sum_{u \in M_p} F_u \quad \approx 0.5$$

cut value: $C_p(\alpha) = \sum_{v \in Nc} w_v$



Layout after Min-cut



Now global placement problem will be solved again with two additional center_of_gravity constraints

Thoughts on Partitioning

Still an active area of research

- Results highly dependent on heuristic improvements and context

Partitioning is the workhorse of placement and floorplanning

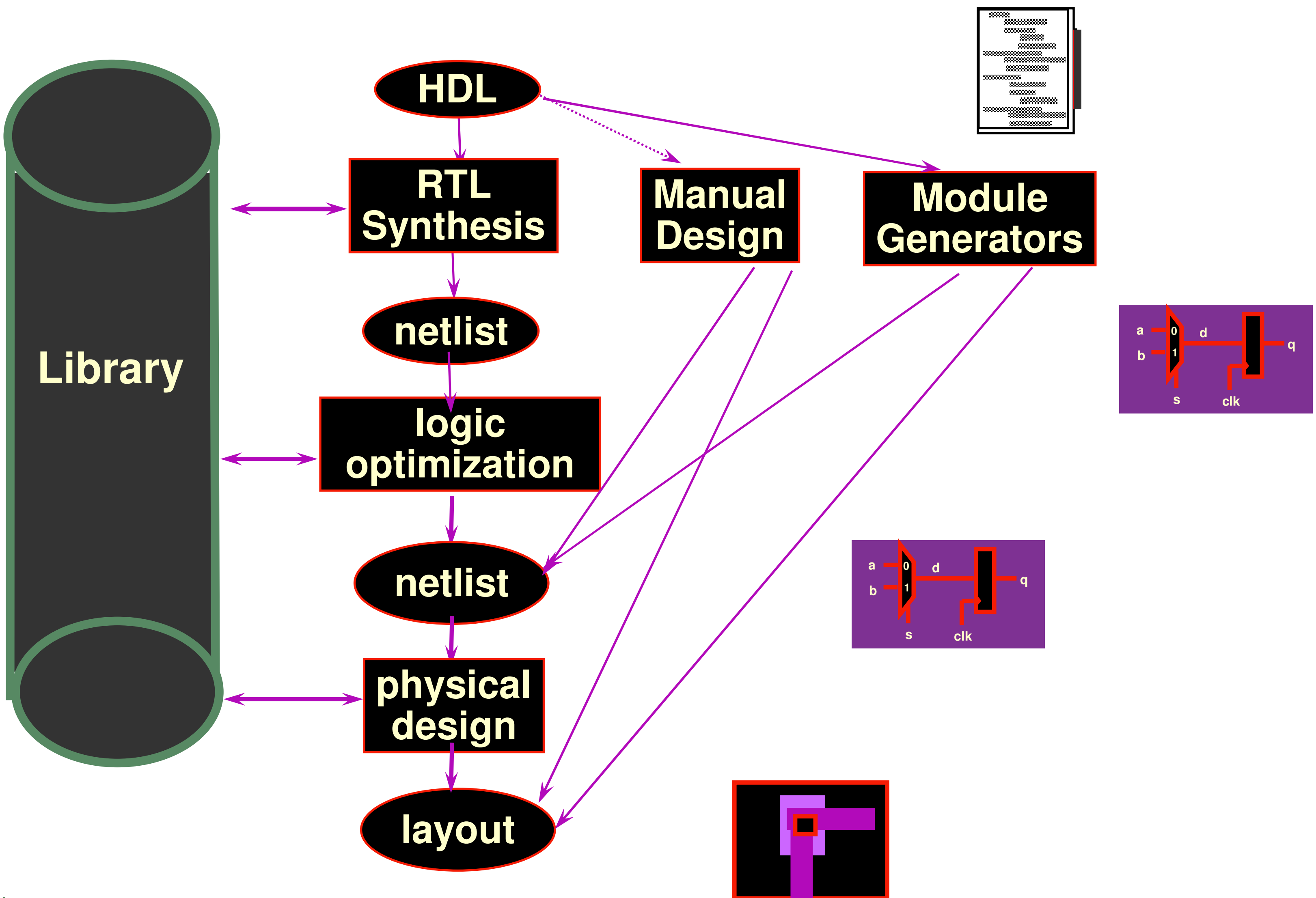
- As a result partitionings must be very fast
- A lot of wasted academic effort on slow (but slightly better) partitioning approaches

K&L, F&M have each held up very well

Reviewing our General Procedure

- Take a real world problem – partitioning of netlists
- Cast in a mathematical abstraction – this often requires simplification
- Identify cost function to be optimized
- Identify size of search space
- Is global optimality computationally feasible?
  - Yes – go to it!
  - No –
    - Identify heuristics that approximate global optimum
    - Simplify problem further and see if you can achieve a local optimum in a computationally efficient manner
- Plug back in the original problem and see how it works

Back in the RTL Design Flow



For Next Class

- Read the Fiduccia & Mattheyses paper
- Read the Gordian paper

Extra Slides

- Simulated annealing
- Fiduccia & Mattheyses

Simulated Annealing

- Uses analogy with metallurgical annealing
- Start with a random initial partitioning
- Generate a new partitioning by exchanging two randomly chosen components from part1 and part2
- Compute the change in score: δs
- If $\delta s < 0$, a lower energy state is found, the move is accepted
- If $\delta s \geq 0$, the move is accepted with probability $\exp(-\delta s / t)$, where t is "temperature"
- Temperature, t, is slowly reduced
  - Helps avoid local minima

Two-Way Partitioning (Fiduccia & Mattheyses)

- Move one cell at a time from one side of the partition to the other in an attempt to minimize the cutset of the final partition
  - **base cell** -- cell to be moved
  - **gain *g(i)*** -- no. of nets by which the cutset would decrease if cell i were moved from partition *A* to partition *B* (may be negative)
- To prevent thrashing, once a cell is moved it is locked for an entire pass
- Claim is O(*n*) time

Two-Way Partitioning (Fiduccia & Mattheyses)

- Steps:

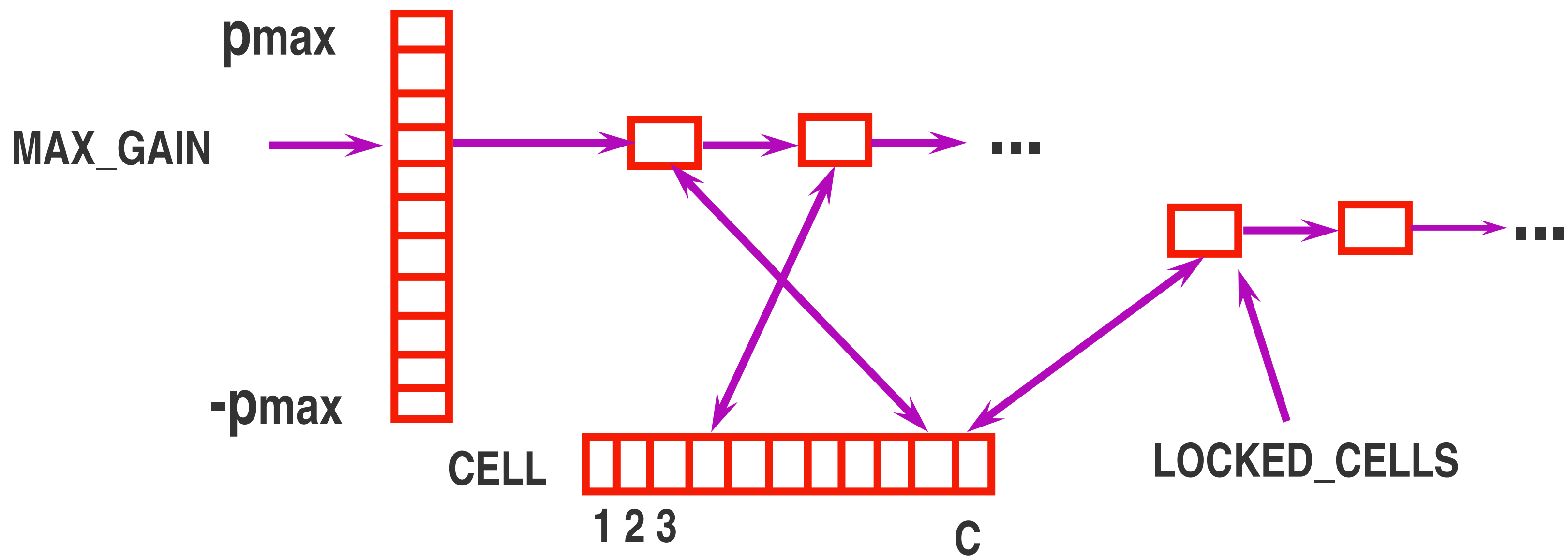
  (1) Choose a cell

  (2) Move it

  (3) Update the g(i)’s of the neighbors

Two-Way Partitioning (Fiduccia & Mattheyses)

- If *p(i)* = no. of pins on cell *i*: $-p(i) < g_i < p(i)$
- Bin-sort cells on *gi*



- Time required to maintain each bucket array O(P)/pass

Two-Way Partitioning (Fiduccia & Mattheyses)

- **Move the Cell**

(1) Find the first cell of highest gain that is not locked and such that moving it would not cause an imbalance

  - Break tie by choosing the one that gives the best balance

(2) Choose this as the base cell. Remove it from the bucket list and place it on the LOCKED list. Update it to the other partition.

- **Updating Cell Gains**

  Critical net

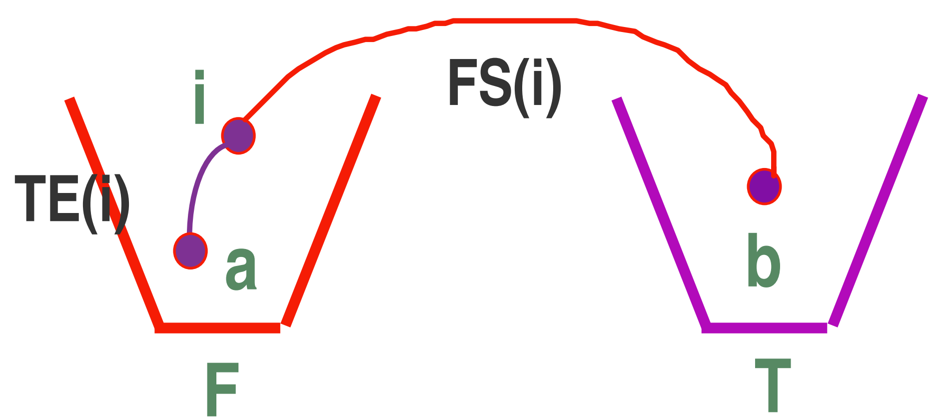
  - Given a partition (*A*/*B*), we define the distribution of *n* as an ordered pair of integers (*A(n)*,*B(n)*), which represents the number of cells net *n* has in blocks *A* and *B* respectively (can be computed in O(P) time for all nets)

Two-Way Partitioning (Fiduccia & Mattheyses)

- Net is critical if there exists a cell on it such that if it were moved it would change the net's cut state (whether it is cut or not).
- Net is critical if $A(n)=0,1$ or $B(n)=0,1$
- Gain of cell depends only on its critical nets:
  - If a net is not critical, its cutstate cannot be affected by the move
  - A net which is not critical either before or after a move cannot influence the gains of its cells
- This is the basis of the linear-time claim

Two-Way Partitioning (Fiduccia & Mattheyses)

- Let *F* be the *from* partition of cell *i* and *T* the *to* partition
- g(i) = FS(i) - TE(i), where:
  - FS(i) = no. of nets which have cell *i* as their only *F* cell
  - TE(i) = no. of nets which contain *i* and have an empty *T* side



Two-Way Partitioning (Fiduccia & Mattheyses)

- **Compute the initial gains of all unlocked cells:**

```
foreach(free cell i) {
    g(i) = 0;
    F = the "from" partition of cell i;
    T = the "to" partition of cell i;
    foreach(net n on cell i) {
        if(F(n) = 1) g(i)++;
        if(T(n) = 0) g(i)--;
    }
}
```

- Requires O(P) work to intialize
  - net is critical before the move iff F(n)=1 or T(n)=0 or T(n) =1
    - F(n) =0 does not occur because base cell on F side before
  - net is critical after the move iff T(n)=1 or F(n)=0 or F(n)=1
    - T(n) =0 does not occur because base cell on T side after

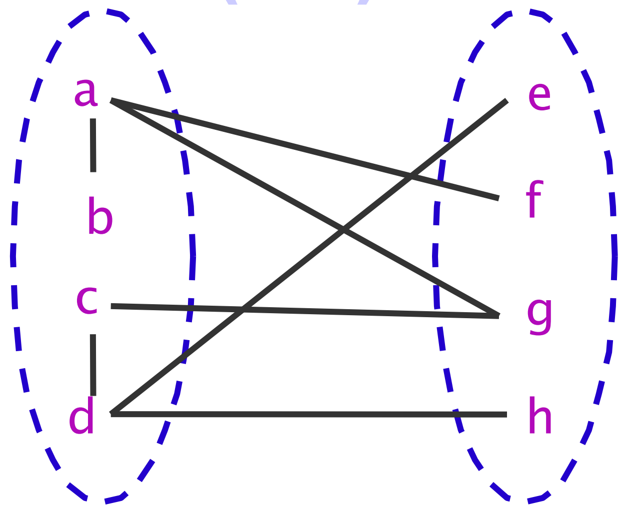
Two-Way Partitioning (Fiduccia & Mattheyses)

- Main loop:

```
lock base cell;
foreach(net n on base cell) {
    if(T(n) == 0) increment gains of all free cells on net n;
    else if(T(n) == 1) decrement gains of the T cell on net n
      if it is free;
    F(n)--;
    T(n)++;
    /* check critical nets after the move */
    if(F(n)== 0) decrement gains of all free cells on net n;
    else if(F(n) == 1) increment gain of the only F cell on
      net n if it is free;
}
```

- Time complexity O(nlog(n))?

Kernighan-Lin (KL) Example - finish



Step No.	Vertex Pair	Gain	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