**EE244: Design Technology for Integrated Circuits and Systems**

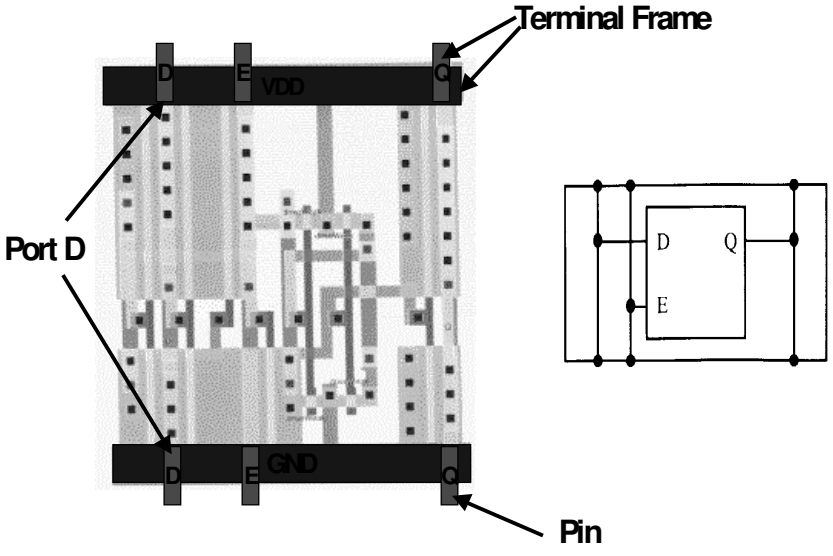
# Outline

**Lecture 2.2**

- **Overall Physical Design Flow**
- **Introduction to Partitioning**
  - **Kernighan-Lin**
  - **Fiduccia & Mattheyses**



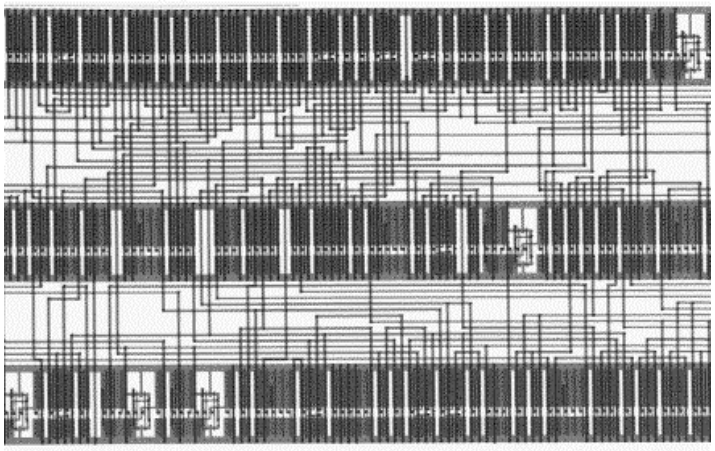
# Abstracting Layouts





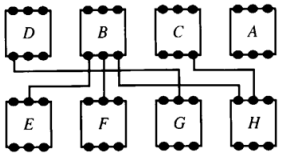

Copyright © 1997, A. Richard Newton

# Standard Cell Layout

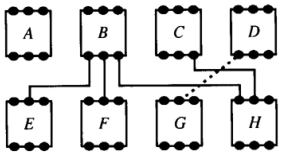




# Goals of Placement



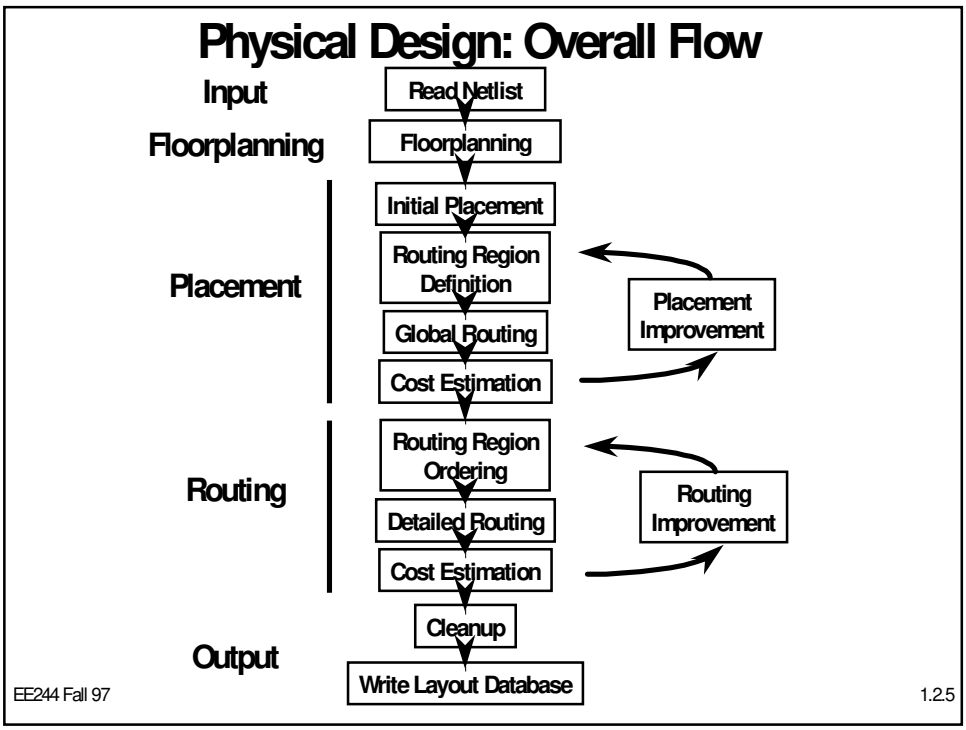

(a) Two tracks required and all connections routed.

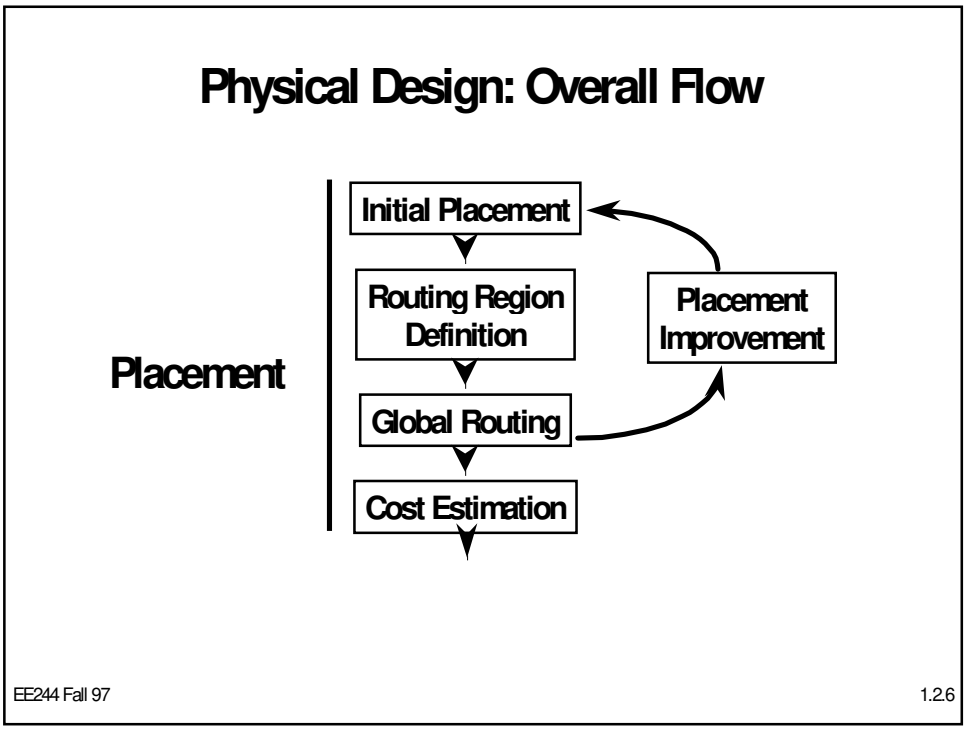



(b) Shorter wire length but three tracks required.



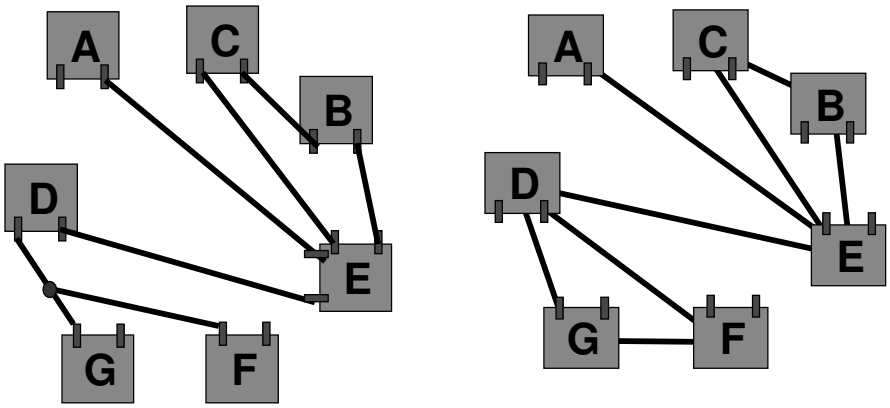
Copyright © 1997, A. Richard Newton



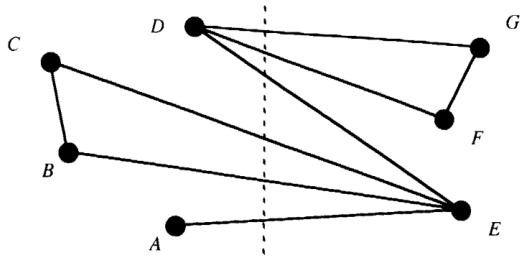

Copyright © 1997, A. Richard Newton

# Netlist Partitioning

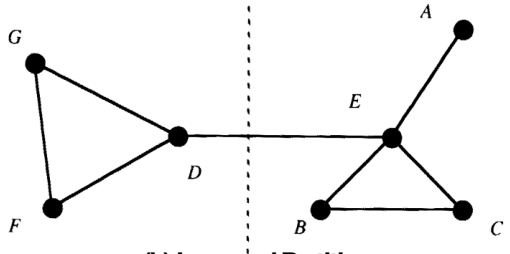





# Partitioning for Minimum Cut-Set




(a) Original Partition (Random)




(b) Improved Partition



Copyright © 1997, A. Richard Newton

# 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 maximim 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*

EE244 Fall 97 1.2.9

# Exact Solutions

- *n* nodes, *k* subsets of size *p*: *kp=n*
- $\binom{n}{p}$ ways to choose the first subset
- $\binom{n-p}{}$ ways to choose the second, etc.
- $\binom{n}{}\binom{n-p}{}\cdots\binom{2p}{}\binom{p}{}$ ways total
- *n*=40, *p*=10 20
- In general, problems where $\beta$

are impractical for real circuits (>1,000,000 gates)

EE244 Fall 97 1.2.10

Copyright © 1997, A. Richard Newton

# *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 clusters nodes based on metric.
  - 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 simulated annealing



# Two-Way Partitioning (Kernighan & Lin)

- Consider the set $S$ of $2n$ vertices, all of equal size for now, with an associated cost matrix
- Assume $C$ is symmetric and
- We want to partition $S$ into two subsets $A$ and $B$, each with $n$ points, such that the external cost 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 a fairly high probability of being a *global minimum* with this scheme



Copyright © 1997, A. Richard Newton

# Two-Way Partitioning (Kernighan & Lin)

- **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 *A* and *B*, excluding *a* and *b*, then:**

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



# Two-Way Partitioning (Kernighan & Lin)

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

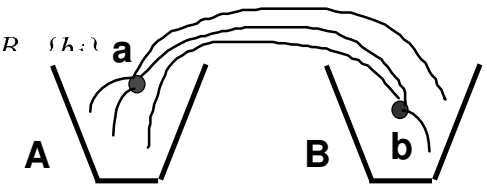
**(2) Choose $a_i$, $b_i$ such that is maximized**

**(3) Set $a_i$ and $b_i$ asside and call them $a_i'$ and $b_i'$**

**(4) Recalculate the D values for all the elements of**

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

$$D_y' = D_y + 2c_{ybj} - 2c_{yai}, y \in B - \{b_i\}$$



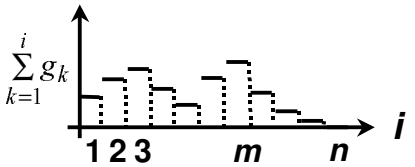



Copyright © 1997, A. Richard Newton

## Two-Way Partitioning (Kernighan & Lin)

- Repeat (2)-(4) on a new pair until all nodes exhausted

$(a_1', b_1'), (a_2', b_2'), \cdots, (a_n', b_n')$




- If sum to $m > 0$, some gain, so repeat until sum to $m = 0$
- What is the time and memory complexity of this algorithm?



## 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



Copyright © 1997, A. Richard Newton

## Two-Way Partitioning (Fiduccia & Mattheyses)

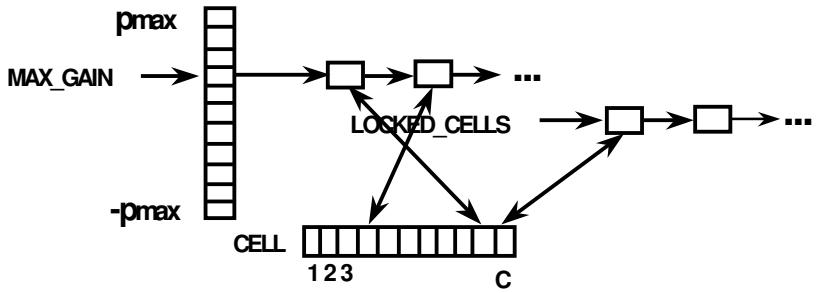
- Steps:
  (1) Choose a cell
  (2) Move it
  (3) Update the g(i)'s of the neighbors
- O(n) neighbors, each cell recomputed each time neighbor moved, and must recompute for each pin (assume #pins=K.n, Amdahl 470 K~2.7)

$$n_i^2 + n_i^2 + \cdots n_i^2 = O(\#\,pins^2) \,/\, pass!$$



## 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



Copyright © 1997, A. Richard Newton

## 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)

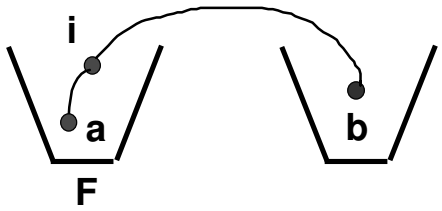
- Net is critical if there exists a cell on it such that if it were moved it would change the net's cut state (whether it is cut or not).
- Net is critical if *A(n)=1* or *B(n)=1* or =0
- Gain of cell depends only on its critical nets:
  - ▲ If a net is not critical, its cutstate cannot be affected by 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



Copyright © 1997, A. Richard Newton

## 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) = 1) 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
  - net is critical after the move if T(n)=1 or F(n)=0 or F(n)=1



Copyright © 1997, A. Richard Newton

# 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(n)?**



Copyright © 1997, A. Richard Newton