CAD Algorithms

Partitioning

Mohammad Tehranipoor

ECE Department

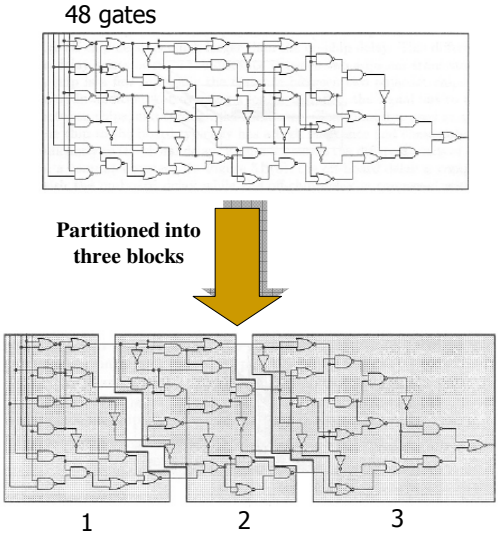
Circuit Partitioning

- **Partitioning**:
  - The process of decomposing a circuit/system into smaller subcircuits/subsystems, which are called **block**, is called ***partitioning***.
- The partitioning **speeds up** the design process.
- Blocks can be designed independently.
- Original functionality of system remains intact.
- An interface specification is generated during the decomposition.
- The decomposition must ensure minimization of interconnections.
- Time required for decomposition must be a small fraction of total design time.

Circuit Partitioning

- If the size of subsystem is still too large, then further partitioning may be required.
- Partitioning is a general technique and finds application in diverse areas such as *divide and conquer* in algorithm design.
- Each block has terminals located at the periphery.
- **Constraints:**
  - Area, pins, timing, number of partitions...
- Blocks can be designed independently and simultaneously speeding up the overall design process.
- **Partitioning is one of the fundamental problems in VLSI Physical Design.**

Example



- # of interconnections between any two blocks is 4 (cut size = 4).
- The cut sizes are not necessarily the same size.
- The blocks are not necessarily the same size.
- Circuit size = 48 gates
- Block 1 = 15 gates
- Block 2 = 16 gates
- Block 3 = 17 gates

Hierarchical Partitioning

- **System Level Partitioning**: A system is partitioned into a set of subsystems whereby each sub-system can be *designed* and *fabricated* independently on a PCB or MCM. The criterion for partitioning is the functionality and each PCB/MCM serves a specific task within a system.

If PCB is too large:

- **Board Level Partitioning**: The circuit assigned to a PCB is partitioned into sub-circuits such that each sub-circuit can be fabricated as a VLSI chip.

If chip is too large:

- **Chip Level Partitioning**: The circuit assigned to a chip is partitioned into smaller subcircuits.

System Level Partitioning

- The circuit assigned to PCB must meet criterion constraints:
  - Fixed area, i.e. *32 cm X 15cm*
  - Fixed number of terminals, i.e. 64
- **Objectives:**
  - Minimize the number of boards:
    - The reliability of the system is inversely proportional to the number of PCBs in the systems.
  - Optimize the system performance:
    - Partitioning must minimize any degradation of the performance caused by the delay due to the connections between components in different boards. System bus is slow!

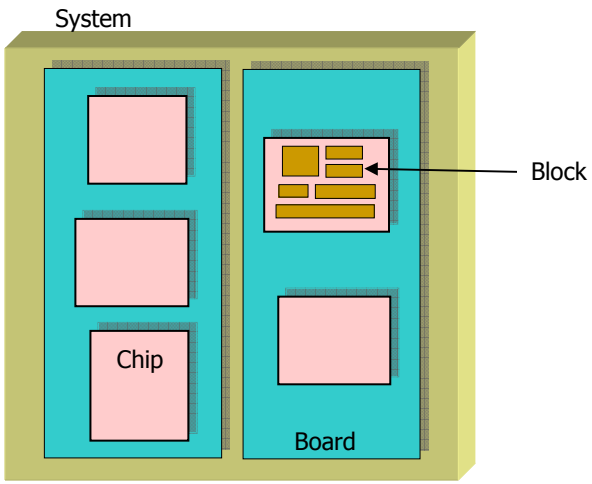
Board Level Partitioning

- Unlike system level partitioning, board level partitioning faces different set of constraints and fulfills different set of objectives.
- Unlike boards, chips can have different sizes and different number of terminals.
  - Size: i.e. from *2mm X 2mm* to *25mm X 25mm*
  - Terminal: i.e. from *64 to 300*
- **Objective:**
  - Minimize the number of chips in each board.
  - Minimize the area of each chip.
    - Both enhance board reliability.
  - Optimize the board performance.

Chip Level Partitioning

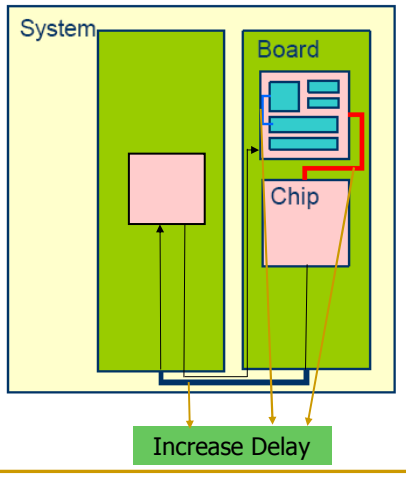
- Each block can be independently designed using either *full custom* or *standard cell* design style.
- There is no area constraint for any partition.
- The number of nets between blocks (partitions) cannot be greater than the terminal count of the partition.
  - The number of pins is based on the block size
- **Objective**:
  - The number of nets cut by partitioning should be minimized.
    - It simplifies the routing task.
    - It mostly results in minimum degradation of performnace.
- **Drawback:** Partitioning may degrade the performance.

Partitioning Levels

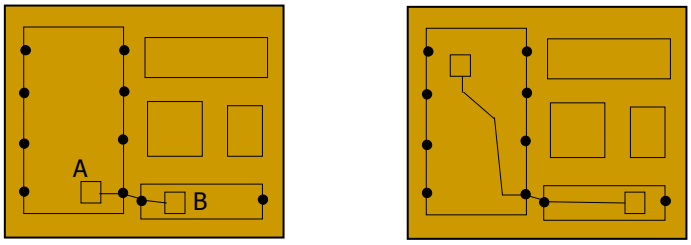


Different Delays in Computer System

- **The off-chip delay is much larger than on-chip delay.**
  - Partitioning results in performance degradation.
- Bad partitioning:
  - Degrades the performance of a chip/board/system *significantly*.



Bad Partitioning



Problem Formulation

- Partitioning problem is expressed in graph theoretic terms.
- $G(V,E)$ represents partitioning problem, let $V=\{v_1, v_2, \ldots, v_n\}$ be a set of vertices and $E=\{e_1, e_2, \ldots, e_m\}$ be a set of edges.
  - Each vertex represents a component.
  - Edges represent connections between components.
- **Partitioning Problem:**
  - V is partitioned into $V_1, V_2, \ldots, V_k$, where
    - $V_i \cap V_j = \emptyset, \quad i \neq j$
    - $U^k_{i=1} V_i = V$
- **Cut size:** Number of edges crossing the cut.
  - C_{ij} is the cut size between partition V_i and V_j.
- Area of each partition V_i: ***Area(V_i)*** $= \Sigma_{v \epsilon Vi}\, a(v)$
- Terminal count: ***Count(V_i)***

Problem Formulation

- Constraints and objective functions for the partitioning algorithms vary for each level of partitioning and each design style.
- **Partitioning Problem:**
  - The *objective* function must be *optimized* under certain *constraints*.
- **Partitioning Problem's Parameters**:
  - Interconnections between partitions
  - Delay due to partitioning
  - Number of terminals
  - Area of each partition
  - Number of partitions

Objective 1

- **Interconnections Between Partitions (Cut Size):**
  - Number of interconnections have to be minimized.
- Reducing the interconnections
  - Reduces the delay.
  - Reduces the interface between partitions.
  - Results in easier independent design and fabrication.
- Large number of interconnections:
  - Increases design area
  - Complicates placement and routing
- **Objective Function**: Minimize the cut (mincut problem)

$$Obj1: \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij} \ , (i \neq j)$$ is minimized

Objective 2

- **Delay Due to Partitioning:**
  - Critical path might go in between partitions a number of times.
  - Delay between partitions is significantly larger than delay within a partition.
  - This is an important factor during partitioning high performance circuits.
- **Objective Function**: Minimize the delay

$$\textit{Obj2}: \max_{p_i \in P}(H(p_i)) \quad \text{is minimized}$$

Constraint 1: Number of Terminals

- Number of nets must not exceed the terminal count of the subcircuit.
- System level partitioning:
  - This limit is decided by the maximum number of terminals available on PCB.
- Board level partitioning:
  - This limit is decided by pin count of the package used for the chips.
- Chip level partitioning:
  - The number of terminals of a subcircuit is determined by the perimeter of the area used by the subcircuit.
- **Constraint**: Number of terminals

$$\textit{Cons1}: \quad Count(V_i) \leq T_i \quad , \quad 1 \leq i \leq k$$

Constraint 2: Area of Each Partition

- System level partitioning:
  - Area of board is fixed. This is a constraint.
- Board level partitioning:
  - There is an upper bound on the area of a chip.
- Chip level partitioning:
  - The size of each partition is not so important (Especially for Full Custom design style).

- **Constraint**:

Cons2: $A_i^{min} \le Area(V_i) \le A_i^{max}$, $i=1,2,\ldots,k$

Constraint 3: Number of Partitions

- It is a constraint at *system level* and *board level* partitioning.
- This obviously prevents:
  - a system from having too many PCBs.
  - a PCB from having too many chips.
    - Both result in decreasing reliability.

- Chip Level Partitioning:
  - The number of partitions is determined by the capability of the placement algorithm.
  - Performance ???

- **Constraint**:

$$K_{min} \le k \le K_{max}$$

Design Style Specific Partitioning Problem

- **Full Custom Design Style:**
  - Partitions can be of different sizes, hence no area Constraints.
  - Partitioning has the most flexibility.
  - Partitioning process can be hierarchical.
  - Number of terminals is a constraint.
  - Minimize total number of nets.
- A partitioning algorithm for full custom design has objective function:
  - *Obj1 (Cut Size)* subject to *Cons1 (Count (V_i))* and *Cons2 (A_i)*.
  - *Obj2 (Delay)* since full custom design style is used for the design of high performance circuits.

Standard Cell Design Style

- **Standard Cell Design Style**:
  - The partitioning process is nonhierarchical.
  - Partition the circuit into a set of disjoint subcircuits.
  - The complexity of partitioning depends on the type of the standard cells available in standard cell library.
  - Partitioning problem has to satisfy *Cons1* and *Cons2*.
  - For high performance circuits, *Obj1* and *Obj2* are used as combined objective functions.

Gate Array Design Style

- **Gate Array Design Style**:
  - The circuit is bipartitioned recursively until each resulting partition corresponds to a gate on the gate array.
  - The objective for each bipartitioning is to minimize the number of nets (*Obj1*) that cross the partition boundries.

Classification of Partitioning Algorithms

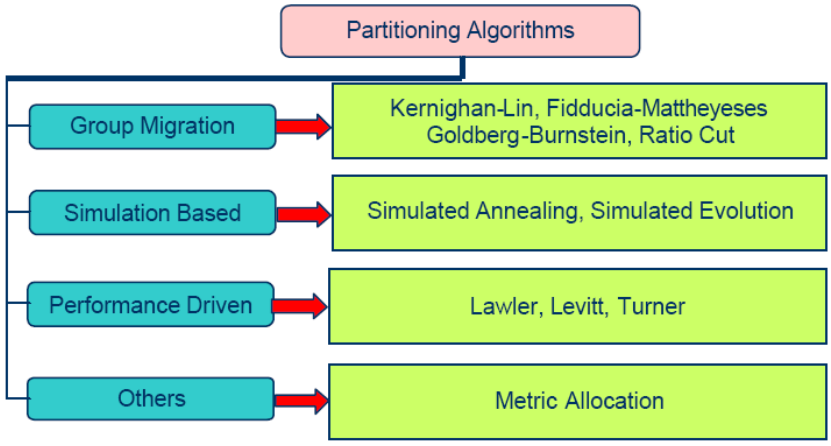
- ***Mincut*** problem is NP-complete.
  - Variety of heuristic algorithms have been developed.

- **Classes of Partitioning Algorithms:**
  - 1- Constructive Algorithms
    - Constructive algorithms are usually used to form some initial partitions which can be implemented by using other algorithms (Preprocessing).
  - 2- Iterative Algorithms
    - Accept a set of partitions and the netlist as input and generate an improved set of partitions with the modified netlist.

Classification of Partitioning Algorithms

- **Classes of Partitioning Algorithms:**
  - 1- Deterministic Algorithms
    - Produce repeatable solutions.
  - 2- Probabilistic Algorithms
    - Produce different solutions for the same problem each time they are used.
- Group Migration Algorithms
- Simulated Annealing and Evolution Based Algorithms
- Other Partitioning Algorithms

Classification of Partitioning Algorithms



Classification of Partitioning Algorithms

- Among all algorithms, *group migration* and *simulation annealing* have been the most successful heuristics for partitioning problems.
- Group Migration Algorithms:
  - Start with some partitions, usually generated randomly, and move components between partitions to improve the partitioning.
  - Quite efficient.
- Simulated Annealing/Evolution Algorithms:
  - Carry out the partitioning process by using a cost function
  - Computationally intensive as compared to group migration and other methods.

Group Migration Algorithms

- The group migration algorithms belong to class of *iterative* improvement algorithms.

1- These algorithms start with some initial partitions usually generated randomly.

2- Local changes are then applied to the partition to reduce the cutsize.

3- This process is repeated until no further improvement is achieved.

Kernighan-Lin (K-L) Algorithm

- K-L is a bisectioning algorithm
  - Input graph is partitioned into two subsets of equal sizes.

- **K-L Algorithm:**
  - Vertex pairs which give the largest decrease in cutsize are exchanged.
    - if decrease is not feasible then the pair that gives minimum increase are exchanged.
  - The procedure is carried out iteratively until no further improvement can be achieved.

K-L Heuristic [1970]

- Iterative improvement based method based on local changes.
- Given a graph $G(V, E)$. Divide vertex set V into two subsets, V_1 and V_2 such that,

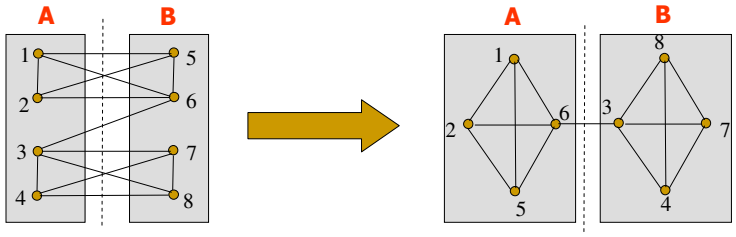
$$|V_1|=|V_2|\pm 1$$

and

$$V_1 \cap V_2 = \emptyset$$

Objective: ***Minimize*** *Cutsize*

Example



Initial Bisections Final Bisections

- **Initial partition: A={1,2,3,4} and B={5,6,7,8}**
- **Initial cutsize = 9**
- **Final partition: A={1,2,5,6} and B={3,4,7,8}**
- **Final cutsize = 1**

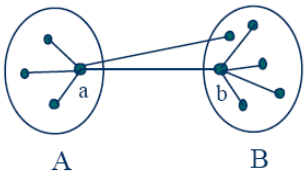
K-L - Definitions

External Cost

$$E_a = \sum_{v \in B} c_{<av>}$$

Internal Cost

$$I_a = \sum_{v \in A} c_{<av>}$$



Node gain

$$D_a = E_a - I_a$$

Decrease in cutsize = Gain of vertex *a*

Number of edges of vertex *a* that do not cross the bisection boundary

Number of edges that cross the boundary

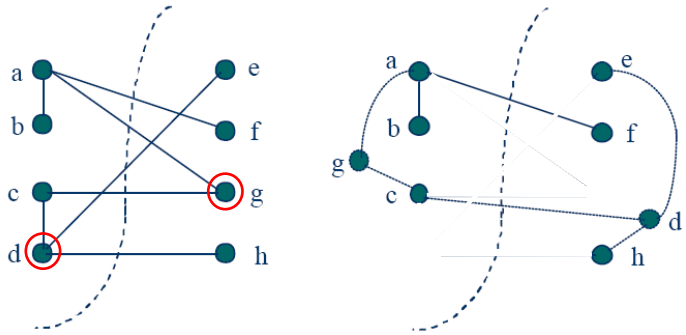
Swap Gain

$$g_{ab} = D_a + D_b - 2c_{<ab>}$$

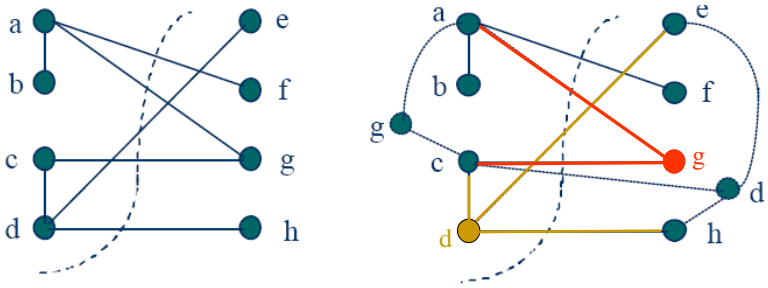
Connection (edges) between *a* and *b*

K-L Move

- $D_d=2-1=1$, $D_g=2-0=2$
- $g_{dg}=1+2-0=3$



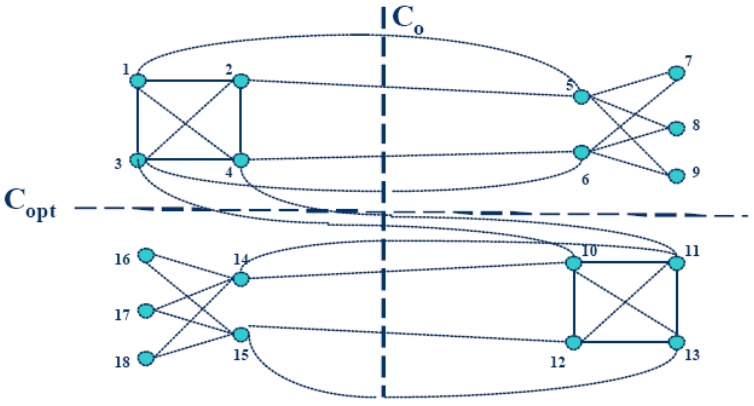
K-L Move



Step No.	Vertex Pair	Gain	Cutset
0		0	5
1	(d,g)	3	2
2	(c,f)	1	1
3	(b,h)	-2	3
4	(a,e)	-2	5

The first and second exchanges will be accepted.

Exhaustive Search



An exhaustive search procedure would require a search on,

$$\frac{1}{2}\binom{18}{2} = 24130$$ **bisections**

The Heuristic

Step #	Pair exchanged	Cutsize	Gain
0	-	10	0
1	4,10	12	-2
2	2,12	12	0
3	1,13	8	+4
4	3,11	2	+6
5	7,18	6	-4
6	8,17	10	-4
7	5,15	12	-2
8	9,16	12	0
9	6,14	10	+2

Log Table

K-L Algorithm

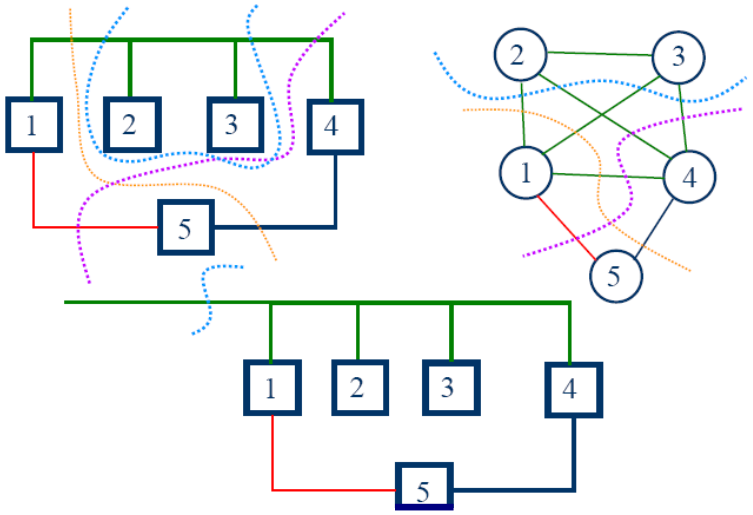
Algorithm KL
begin

```
INITIALIZE();      // Finds initial bisections
while (IMPROVE (table) = TRUE) do // IMPROVE: Tests if any improvement has been made.
  while (UNLOCK(A) = TRUE) do     // UNLOCK: Checks if any vertex is unlocked. Each
      vertex has a status of either locked or unlocked. Only those vertices whose status is
      unlocked are candidates for the next tentative exchanges.
    for (each a ϵ A) do
      if (a = unlocked) then
        for (each b ϵ B) do
          if (b = unlocked) then
            if (D_max < D(a) + D(b) – 2c) then
              D_max = D(a) + D(b) – 2c;
               a_max = a;
               b_max = b;
      TENT_EXCHANGE(a_max,b_max);   // Tentatively exchanges a pair of vertices
      LOCK(a_max,b_max);    // Locks the vertex pair
      LOG(table);  // Stores the table
      D_max = -∝;
end
ACTUAL_EXCHANGE(table);     // Determines the maximum partial sum
```


K-L Algorithm

- The time complexity of K-L algorithm is ***O(n³)***.
- K-L algorithm is robust.
- It can accommodate additional constraints, such as group of vertices requiring to be in a specific partition.
  - This is an important feature.
  - For example, all component of an adder must be kept together or all gates on a critical path must be kept closer to minimize wirelength and delay.
- The algorithm is not applicable for **hypergraphs**.
- The partition sizes have to be specified before partitioning.
- It cannot handle arbitrarily weighted graphs
  - K-L is bisection algorithms with two same size partitions
- The complexity is considered high even for moderate size problems.

Proper Model for Circuit Representation

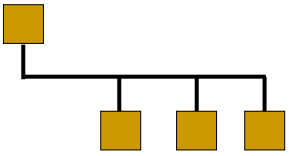


Hypergraph

- **Definition:** A *graph* whose *hyperedges* connect two or more *vertices*.
  - Informally a hypergraph is a graph whose edges, instead of each connecting just two vertices, connect more than two vertices.
- A *hypergraph* is a pair (V,E), where V is a set of vertices and E is a set of edges which is defined over a *family of set of vertices*.

$$e_i \subseteq V, 1 \le i \le m$$

H = (V,E)
V = $\{v_1, v_2, \ldots, v_n\}$
E = $\{e_1, e_2, \ldots, e_m\}$



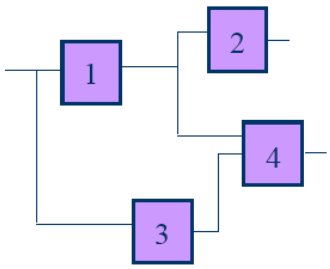
Fiduccia-Mattheyses (F-M) Alg.

- An improvement over Kernighan-Lin algorithm.
- Operates directly on hypergraph model.
- Significantly faster than K-L algorithm.
  - linear time complexity.
- Only a *single vertex* is moved across the cut in a single move.
  - Unlike K-L, it can handle unbalanced partitions and nonuniform vertex weights
- Faster selection procedure

Fiduccia-Mattheyses Method

- Circuit is viewed as a set of C cells and N nets.
  - # of cells in net i = $n(i)$.
  - size of cell i = $s(i)$.
  - # of pins on cell i = $p(i)$.
  - P is the total number of pins in the circuit.

$$P = \sum_{i=1}^{cells} p(i)$$

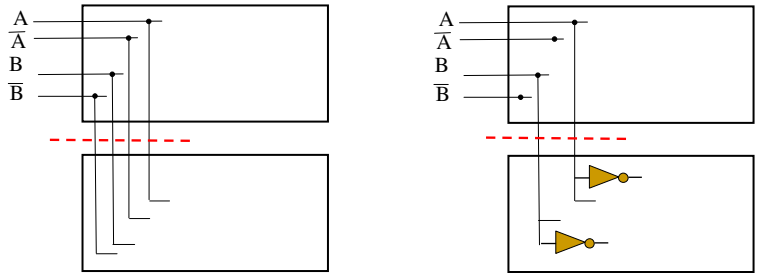


Coldberg and Burstein Algorithm

- Experimental results have shown that the quality of K-L algorithm depends heavily on the *ratio of the number of edges to the number of vertices (Ne/Nv)*.
  - K-L yields good bisection if the ratio is higher than 5.
  - If the ratio is less than 3, the algorithm performs poorly.
- The basic idea of Goldberg-Burstein algorithm is to find matching in a graph. (see pages 174-175 of text book). Each edge in the matching is contracted to increase the density of graph. Any bisection algorithm can be applied to the modified graph. (See Figure 5.9 (page 175))

Component Replication

- Partitioning definition:
  - V is partitioned into $V_1, V_2, \ldots, V_k$, where
    - $V_i \cap V_j = \emptyset, \quad i \neq j$
    - $U^k_{i=1} V_i = V$
- See pages 174-176 of text book.



See Figure 5.10 (page 176)

Simulated-Based Algorithms

Simulated Annealing Algorithm for Circuit Partitioning (Page 178):

?

Genetic Algorithm for Circuit Partitioning (Page 181):

?