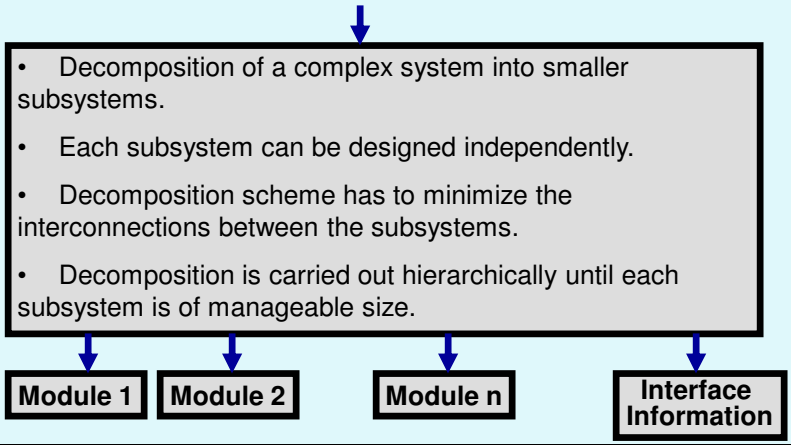
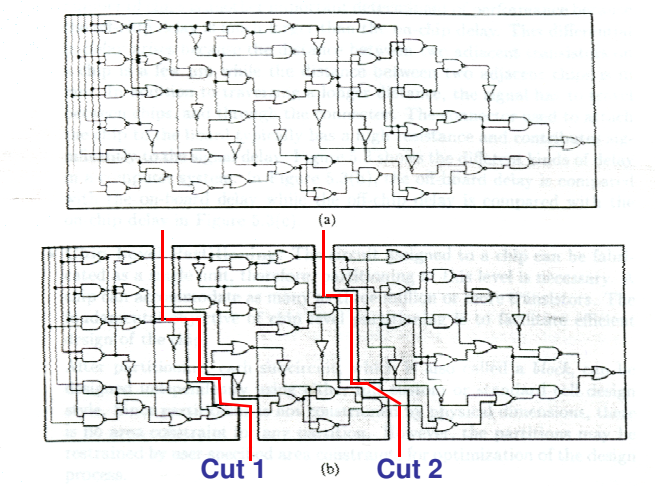
# BACKEND DESIGN

# Circuit Partitioning

## Partitioning

**System Design**






Cut 1 = 4

Cut 2 = 4

Size 1 = 15

Size 2 = 16

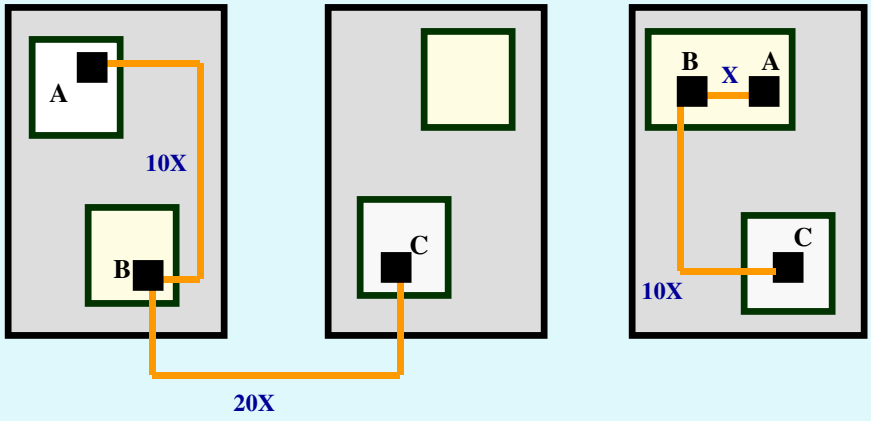
Size 3 = 17



# Partitioning at Different Levels

- Can be done at multiple levels:
  - System level
  - Board level
  - Chip level
- Delay implications are different:
  - Intrachip ➔ X
  - Intraboard ➔ 10X
  - Interboard ➔ 20X

# Different Delays in a Chip

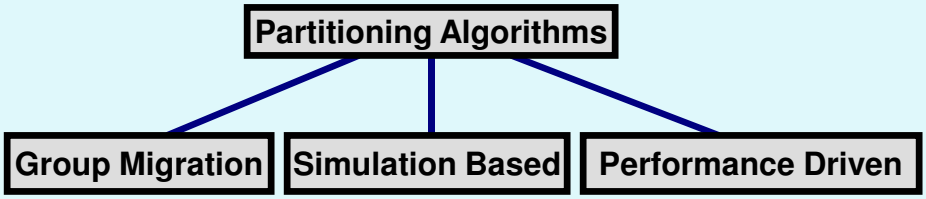





# Problem Formulation

- Partition a given netlist into smaller netlists such that:
  1. Interconnection between partitions is minimized.
  2. Delay due to partitioning is minimized.
  3. Number of terminals is less than a predetermined maximum value.
  4. The area of each partition remains within specified bounds.
  5. The number of partitions also remains within specified bounds.

# Classification of Partitioning Algorithms




Kernighan-Lin

Fiduccia-Mattheyses

Goldberg-Burstein

Simulated Annealing

Simulated Evolution



# Group Migration Algorithms

- Kernighan-Lin
  - An iterative improvement algorithm for balanced two-way partitioning.
- Goldberg-Burstein
  - Uses properties of graphs to improve the performance of K-L algorithm.
- Fiduccia-Mattheyses
  - Considers multi-pin nets.
  - Can generate partitions of unequal sizes.
  - Uses efficient data structure to represent nodes.

# Extension of K-L Algorithm

- **Unequal sized blocks**
  - To partition a graph with 2n vertices into two subgraphs of unequal sizes n1 and n2:
    - Divide the nodes into two subsets A and B, containing MIN(n1,n2) and MAX(n1,n2) vertices respectively.
    - Apply K-L algorithm, but restrict the maximum number of vertices that can be interchanged in one pass to MIN(n1,n2).



- **Unequal sized elements**
  - To generate a two-way partition of a graph whose vertices have unequal sizes:
    - Assume that the smallest element has unit size.
    - Replace each element of size s with s vertices which are fully connected (s-clique) with edges of infinite weight.
    - Apply K-L algorithm to the modified graph.

## Simulated Annealing and Evolution

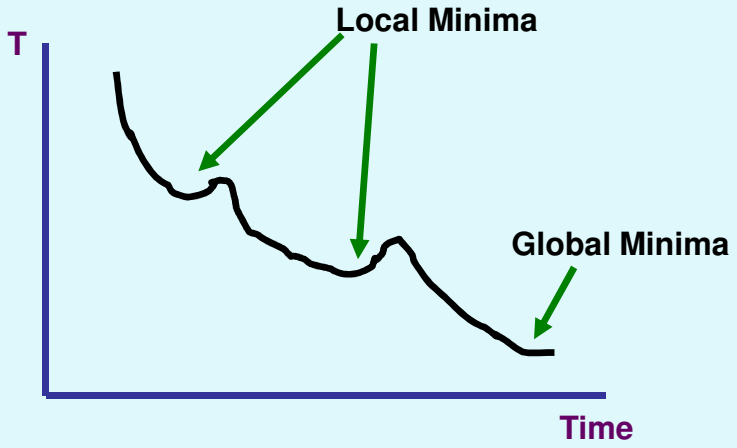
- These belong to the probabilistic and iterative class of algorithms.
- Simulated Annealing
  - Simulates the annealing process used for metals.
  - As in the actual annealing process, the value of temperature is decreased slowly till it approaches the freezing point.
- Simulated Evolution
  - Simulates the biological process of evolution.
  - Each solution (generation) is improved in each iteration by using operators which simulate the biological events in the evolution process.



## Simulated Annealing

- Concept analogous to the annealing process for metals and glass.
- A random initial partition is available as input.
- A new partition is generated by exchanging some elements.
- If the quality of partition improves, the move is always accepted.
- If not, the move is accepted with a probability which decreases with the increase in a parameter called *temperature* (T).

# The Annealing Curve






# Simulated Annealing Algorithm

```
Algorithm SA
begin
    t = t0;
    cur_part = ini_part;
    cur_score = SCORE (cur_part);
    repeat
        repeat
            comp1 = SELECT (part1);
            comp2 = SELECT (part2);
            trial_part = EXCHANGE (comp1,comp2,cur_part);
            trial_score = SCORE (trial_part);
            δs = trial_score – cur_score;
```

```
if (δs < 0)  then
                cur_score = trial_score;
                cur_part = MOVE (comp1, comp2);
            else
                r = RAND (0,1);
                if  (r < exp(– δs/t)) then
                    cur_score = trial_score;
                    cur_part = MOVE (comp1, comp2);
        until  (equilibrium at t is reached);
        t = αt;      /* 0 < α < 1  */
    until  (freezing point is reached);
end.
```



- **The SCORE function**

  Imbalance (A,B) = | size(A) – size(B) |

  Cutcost (A,B) = Sum of weights of cut edges

  Cost = $W_1$ * Imbalance(A,B) + $W_2$ * Cutcost(A,B)

- **The MOVE function**
  - Several alternatives:
    - Pairwise exchange ($W_1$ =0)
    - Subsets of elements exchanged
    - Select that node
      - which is internally connected to least number of vertices
      - whose contribution to external cost is highest

## Performance Driven Partitioning

- Typically, on-board delay is three orders of magnitude larger than on-chip delay.
  - On-chip delay is of the order of nanoseconds.
  - On-board delay can be in the order of milliseconds.
- If a critical path is cut many times by the partition, the delay in the path may be too large to meet the goals of high-performance systems.
- Goal of partitioning in high-performance systems:
  1. Reduce the cut-size.
  2. Minimize the delay in critical paths.
  3. Timing constraints have to be satisfied.



## Contd.

- The problem can be modeled as a graph.
  - Each vertex represents a component (gate).
  - Each edge represents a connection between two gates.
  - Each vertex has a weight specifying the component delay.
  - Each edge has a weight, which depends on the partitions to which the edges belong.
- This problem is very general and still a topic of intensive research.

# Summary

- **Broadly, two classes of algorithms:**
  1. **Group migration based**
     - **High speed**
     - **Poor performance**
  2. **Simulation based**
     - **Low speed**
     - **High performance**