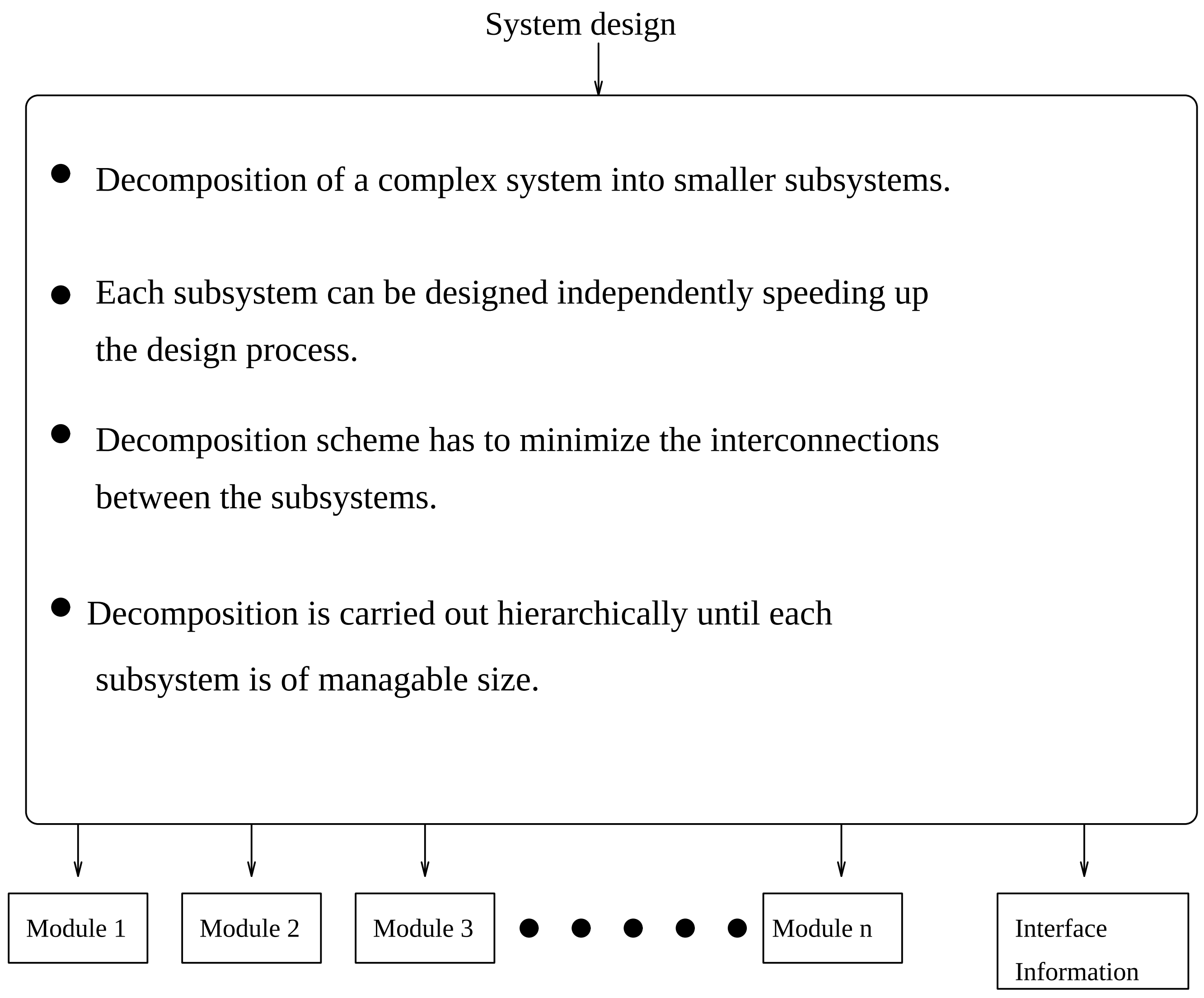
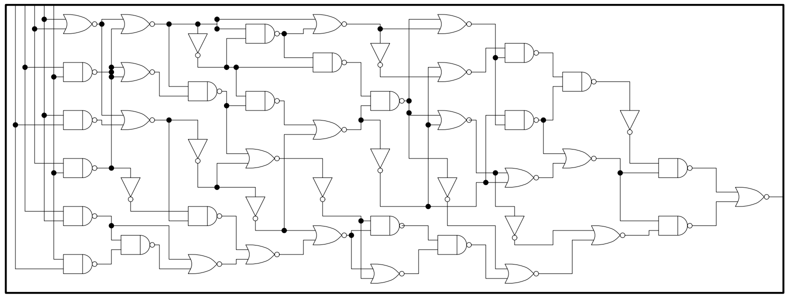
Partitioning

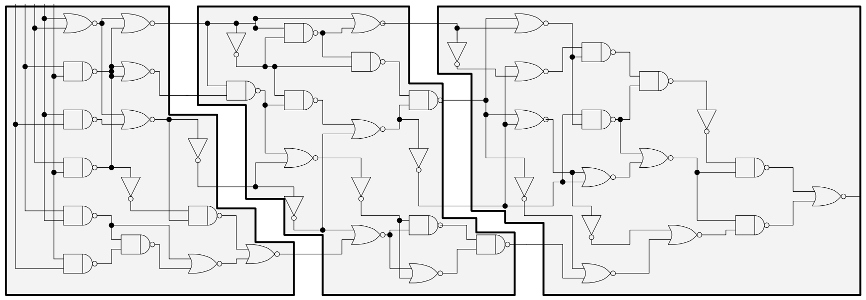


- Decomposition of a complex system into smaller subsystems.
- Each subsystem can be designed independently speeding up the design process.
- Decomposition scheme has to minimize the interconnections between the subsystems.
- Decomposition is carried out hierarchically until each subsystem is of managable size.

Partitioning of A Circuit

Input size = 48

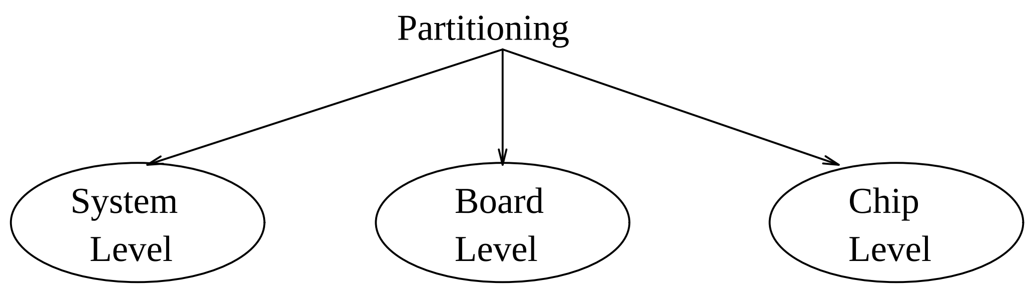

(a)

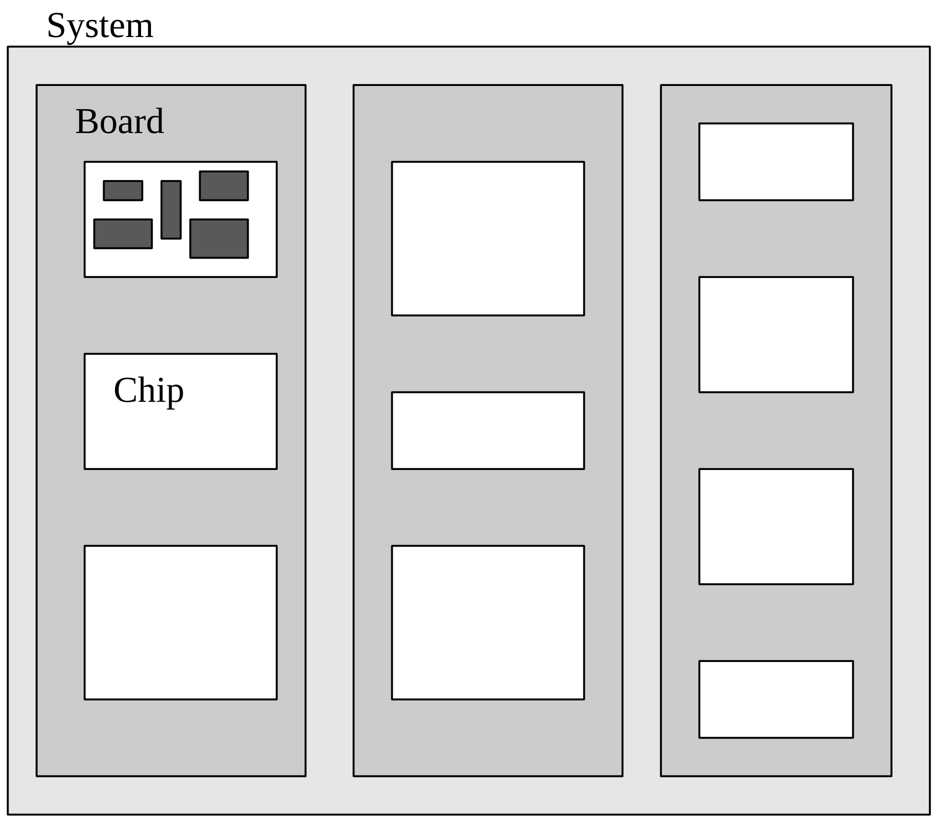

(b)

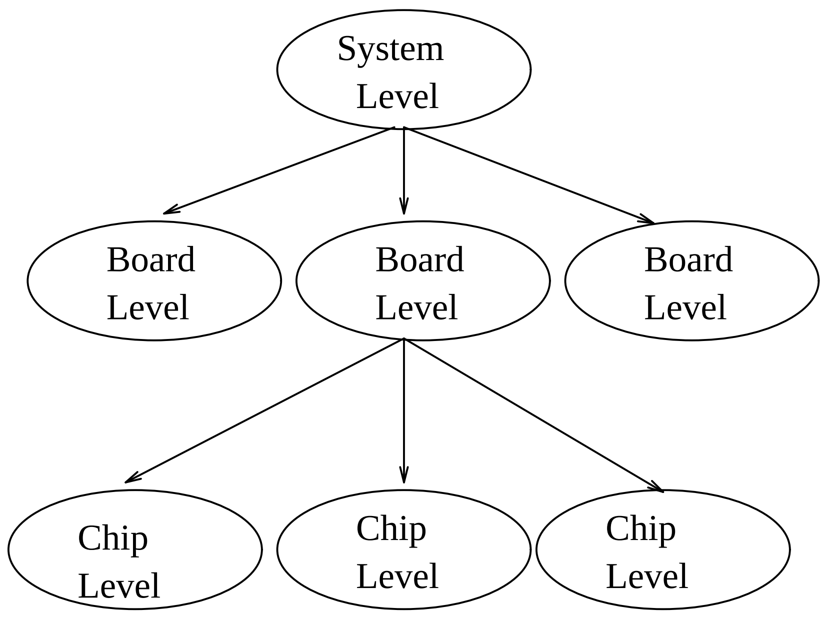
Cut 1 = 4   Cut 2 = 4

Size 1 = 15   Size 2 = 16   Size 3 = 17

Partitioning at different levels







Hierarchical Partitioning

The levels of partitioning are:

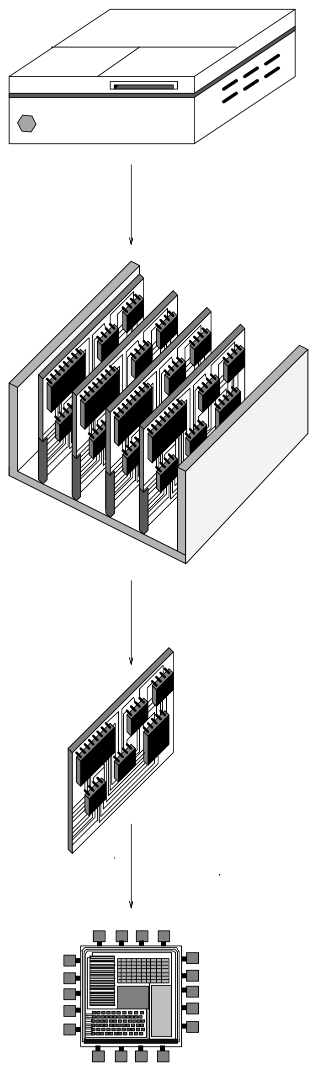
1. System Level Partitioning: A system is partitioned into a set of subsystems such that each subsystem can be designed and fabricated independently on a single PCB.

2. Board Level Partitioning: The circuit assigned to a PCB is partitioned into subcircuits such that each subcircuit can be fabricated as a VLSI chip.

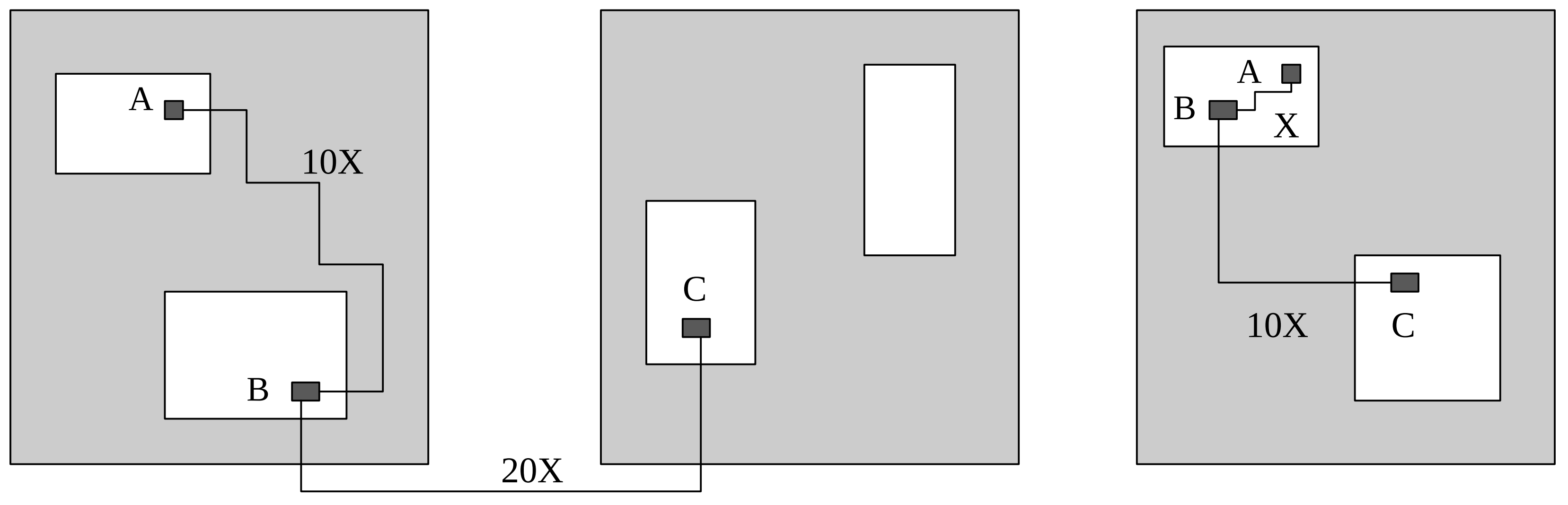
3. Chip Level Partitioning: The circuit assigned to a chip is partitioned into smaller subcircuits.

Chip level partitioning is physically not necessary.

System Hierarchy



Different Delays in A Computer System



(a)

(b)

- No critical net should cut boundaries if delay is not justified.

Problem Formulation

1. Interconnections between partitions:

$$Obj_1 : \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij}, (i \neq j) \quad \text{is minimized}$$

2. Delay due to partitioning:

$$Obj_2 : \max_{p_i \in P}(H(p_i)) \quad \text{is minimized}$$

3. Number of terminals:

$$Cons_1 : Count(V_i) \leq T_i, \quad 1 \leq i \leq k$$

where,

c_{ij} is the cutsize between partitions V_i and V_j.

$H(p_i)$ is the number of times a hyperpath p_i is cut.

$Count(V_i)$ is the terminal count for partition V_i.

Problem Formulation

1. Area of each partition:

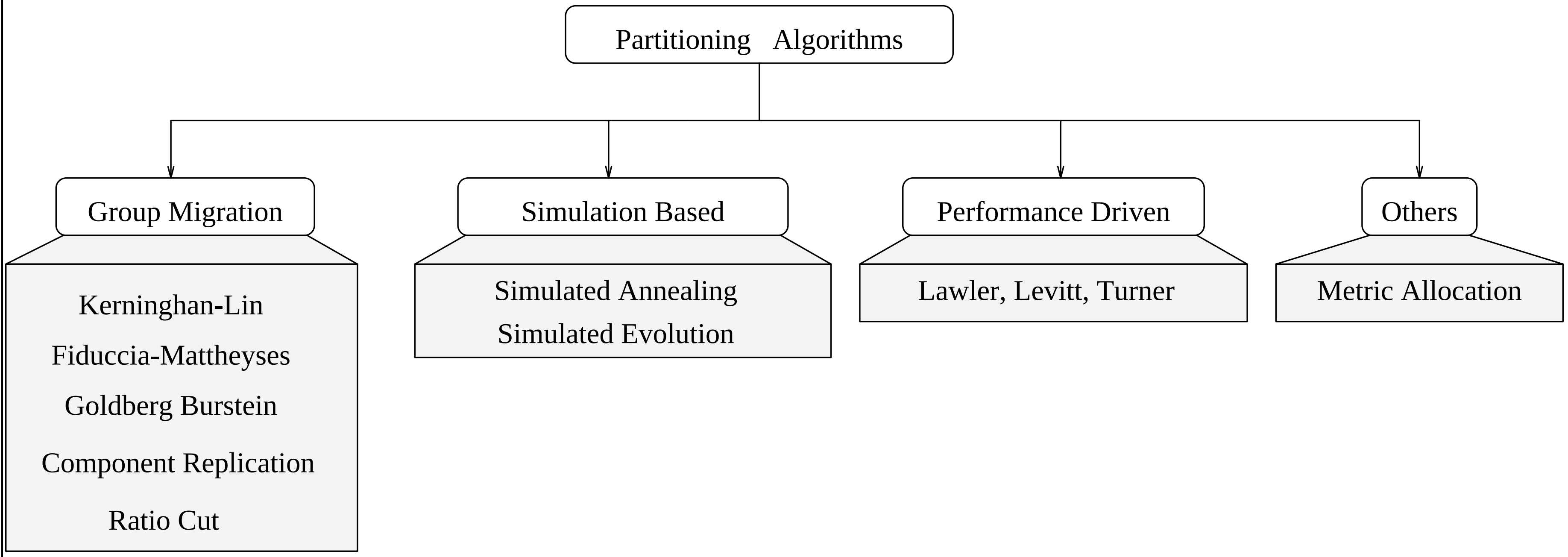
$$Cons_2 : A_i^{\min} \leq Area(V_i) \leq A_i^{\max}, \quad i = 1, 2, \ldots, k$$

2. Number of partitions:

$$Cons_3 : K_{\min} \leq k \leq K_{\max}$$

The partitioning problem at any level or design style deals with one or more of the above parameters.

Classification of Partitioning Algorithms



Kernighan-Lin Algorithm

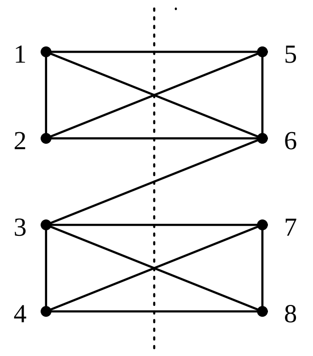
- It is a bisectioning algorithm
- The input graph is partitioned into two subsets of equal sizes.
- Till the cutsize keeps improving,
  - Vertex pairs which give the largest decrease in cutsize are exchanged
  - These vertices are then locked
  - If no improvement is possible and some vertices are still unlocked, the vertices which give the smallest increase are exchanged

W. Kernighan and S. Lin, Bell System Technical Journal, 1970.

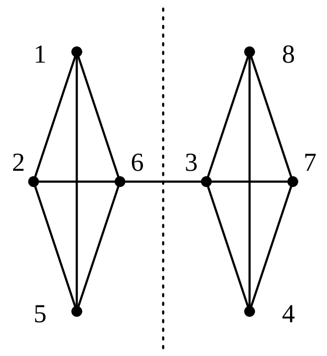
Kernighan-Lin Algorithm

```
Algorithm KL
begin
    INITIALIZE();
    while( IMPROVE(table) = TRUE ) do
    (* if an improvement has been made during last iteration,
    the process is carried out again. *)
        while ( UNLOCK($A$) = TRUE ) do
        (* if there exists any unlocked vertex in $A$,
        more tentative exchanges are carried out. *)
            for ( each $a \in A$ ) do
                if ($a$ = unlocked) then
                    for( each $b \in B$ ) do
                        if ($b$ = unlocked) then
                            if ($D_{\max} < D(a) + D(b)$) then
                                $D_{\max} = D(a) + D(b)$;
                                $a_{\max} = a$;
                                $b_{\max} = b$;
            TENT-EXCHGE($a_{\max}, b_{\max}$);
            LOCK($a_{\max}, b_{\max}$);
            LOG(table);
            $D_{\max} = -\infty$;
        ACTUAL-EXCHGE(table);
end.
```

A Graph Bisected by K-L Algorithm



(a) Initial Bisections



(b) Final Bisections

i	Vertex Pair	$g(i)$	$\Sigma_{j=1}^{i} g(i)$	Cutsize
0	-	-	-	9
1	(3,5)	3	3	6
2	(4,6)	5	8	1
3	(1,7)	-6	2	7
4	(2,8)	-2	0	9

Drawbacks of K-L Algorithm

- K-L algorithm considers balanced partitions only.
- As vertices have unit weights, it is not possible to allocate a vertex to a partition.
- The K-L algorithm considers edges instead of hyperedges.
- High, $O(n^3)$ complexity.

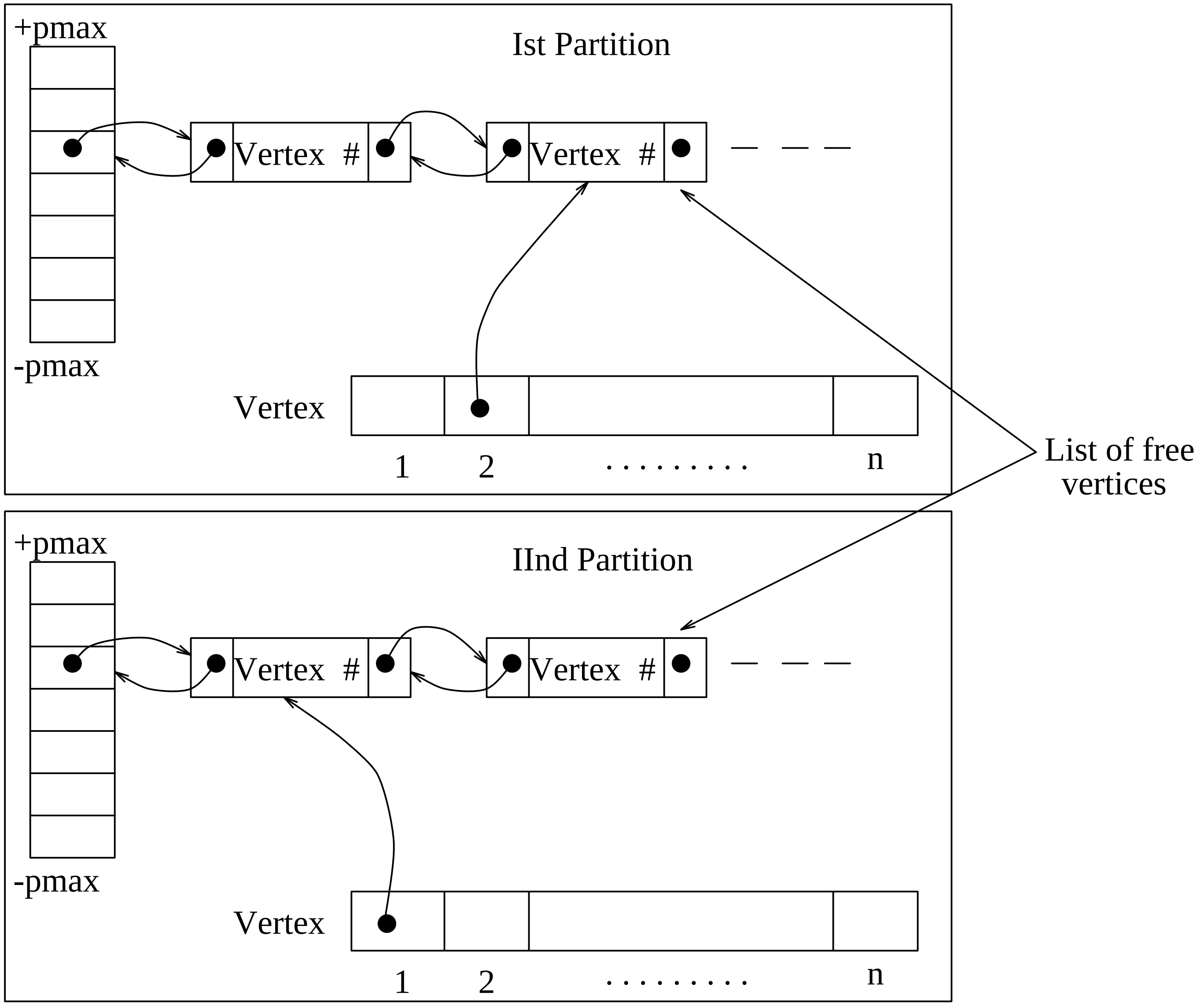
Fiduccia-Mattheyses Algorithm

This algorithm is a modified version of Kernighan-Lin Algorithm.

- A single vertex is moved across the cut in a single move which permits handling of unbalanced partitions.
- The concept of cutsize is extended to hypergraphs.
- Vertices to be moved are selected in a way to improve time complexity.
- A special data structure is used to do this.
- Overall time complexity of the algorithm is $O(n^2)$.

C. M. Fiduccia and R. M. Mattheyses, 19th DAC, 1982.

Data Structure Used in Fiduccia-Mattheyses Algorithm

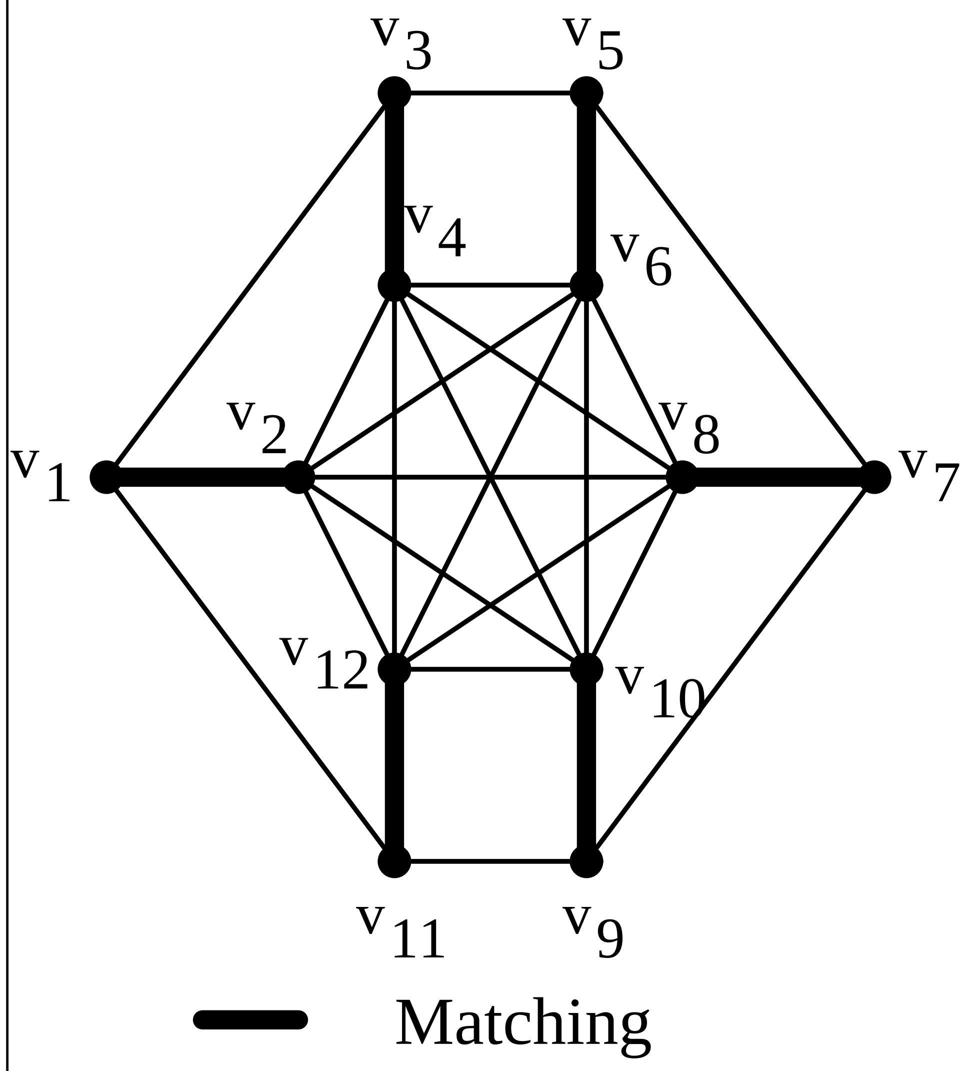


Goldberg and Burstein Algorithm

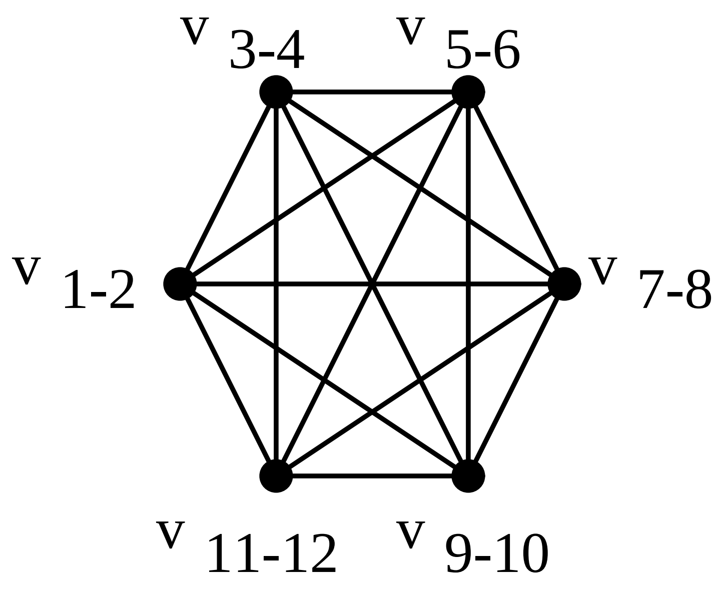
- Performance of K-L algorithm depends on the ratio R, of edges to vertices
- K-L algorithm yield good bisections if $R > 5$
- In a typical VSLI design problem, $1.8 \leq R \leq 2.5$
- Goldberg and Burstein improve K-L algorithm to increase R
    - Find a matching M in graph G
    - Each edge in the matching is contracted
      to increase the density of graph.
    - Any bisection algorithm is applied
      to the modified graph
    - Edges are uncontracted within each partition.

M. K. Goldberg and M. Burstein, ICCD, 1983.

Example of Goldberg and Burstein Algorithm



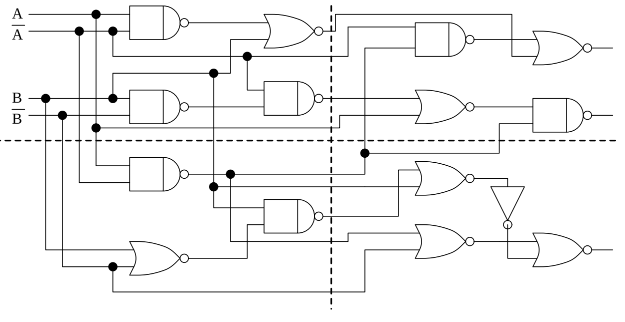
(a) Matching of Graph



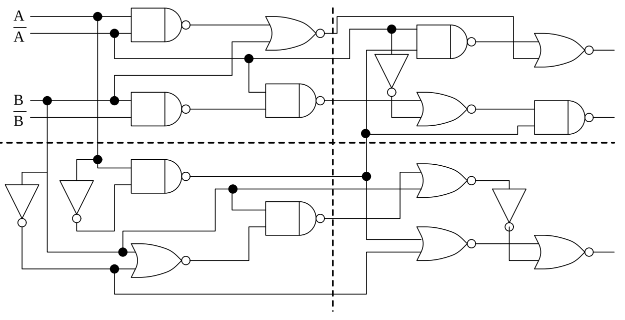
(b) After Contracting

Component Replication to Reduce Cutsize

- Vertices are replicated to improve cutsize
- Good results if limited number of components are replicated



(a)



(b)

Fig(a) shows a 4-way partition before component replication with total cutsize=16
Fig(b) shows the 4-way partition after component replication with total cutsize=10

C. Kring and A. R. Newton, ICCAD, 1991.

Comparison of Different Circuits

Circuit Type	Maximum Circuit Expansion	Net Cutsize Reduction	Component Replication
Combinational	10%	41%	3.2%
	30%	43%	3.9%
Industrial	10%	15%	0.7%
	30%	9%	0.3%

Ratio Cut

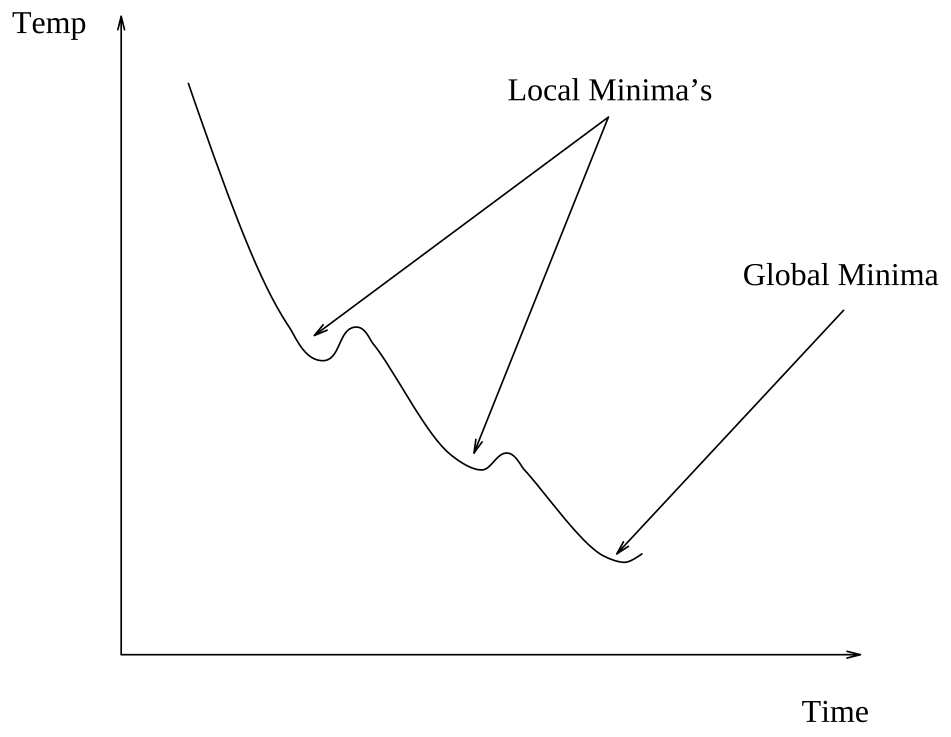
- Locates natural clusters in the circuit
- Forces partitions to be of equal size
- Each edge has a capacity defined for it
- The algorithm tries to minimize the ratio of capacity of the edge going in between the partitions to the product of number of nodes in the partitions.

Y. Wei and C. Cheng, ICCAD, 1989.

Simulated Annealing Algorithm

- Concept analogous to the annealing process used for metals and glass.
- A random initial partition is available as input.
- A new partitioning is generated by exchanging some elements.
- If the partitions improve the move is always accepted.
- If not then the move is accepted with a probability which decreases with the increase in a parameter called temperature T

The Annealing curve

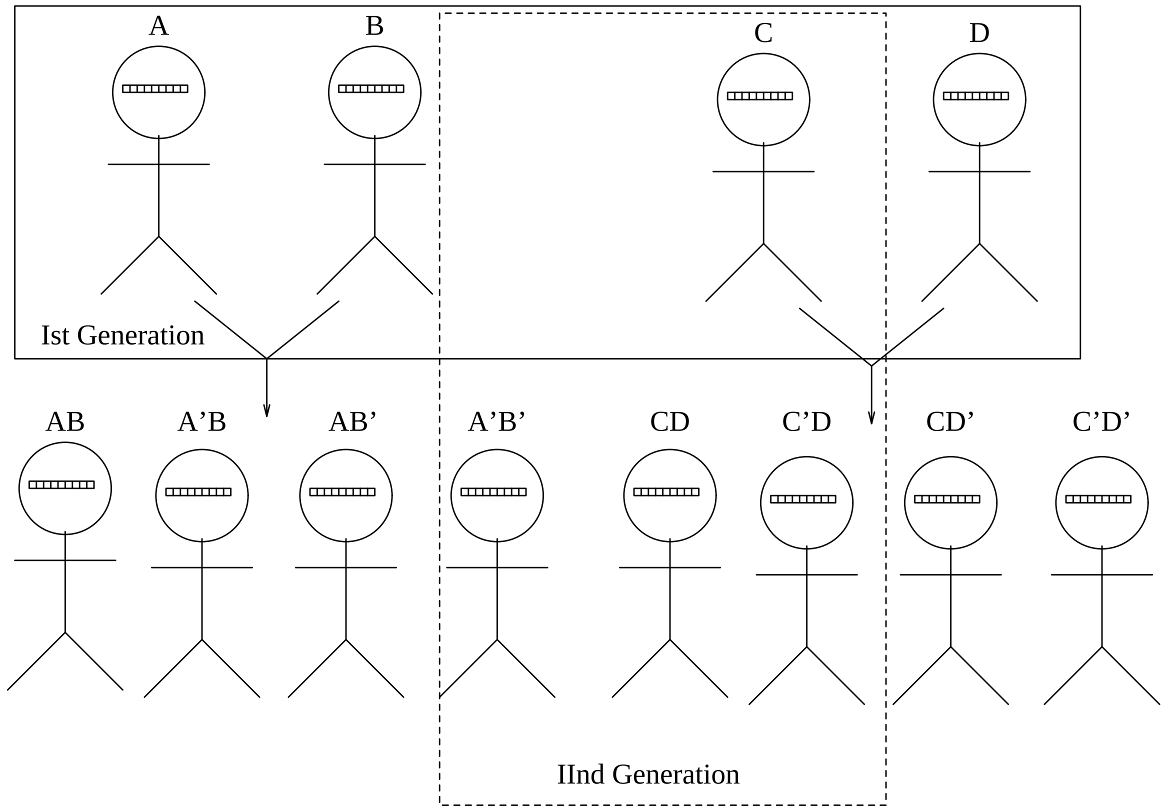


Simulated Annealing Algorithm

```
Algorithm SA
begin
    $t = t_0$;
    $cur\_part = ini\_part$;
    $cur\_score = \text{SCORE}(cur\_part)$;
    repeat
        repeat
            $comp1 = \text{SELECT}(part1)$;
            $comp2 = \text{SELECT}(part2)$;
            $trial\_part = \text{EXCHANGE}(comp1, comp2, cur\_part)$;
            $trial\_score = \text{SCORE}(trial\_part)$;
            $\delta s = trial\_score - cur\_score$;
            if ($\delta s < 0$) then
                $cur\_score = trial\_score$;
                $cur\_part = \text{MOVE}(comp1, comp2)$;
            else
                $r = \text{RANDOM}(0, 1)$;
                if ($r < e^{-\frac{\delta s}{t}}$) then
                    $cur\_score = trial\_score$;
                    $cur\_part = \text{MOVE}(comp1, comp2\ )$;
        until (equilibrium at $t$ is reached)
        $t = \alpha t$     (* $0 < \alpha < 1$ *)
    until (freezing point is reached)
end.
```

Simulated Evolution Algorithm

- Each feasible solution to the problem is considered as a generation.
- The bad individuals in the population are eliminated to generate a new generation.



Y. Saab and V. Rao, DAC 1990.

Simulated Evolution Algorithm

```
**Algorithm** SE
**begin**
    $S = S_0$;    (* initial state *)
    $S_{BEST} = S$;    (* save initial state *)
    $p = p_0$;    (* initialize control parameter *)
    $\gamma = 0$;    (* initialize counter *)
    **repeat**
        $C_{pre} = \text{COST}(S)$;
        $S = \text{PERTURB}(S, p)$;
        $C_{cur} = \text{COST}(S)$;
        $\text{UPDATE}(p, C_{pre}, C_{cur})$;
        **if** $(\text{COST}(S) < \text{COST}(S_{BEST}))$ **then**
            $S_{BEST} = S$;    (* save best state *)
            $\gamma = \gamma - R$;    (* decrement counter *)
        **else**
            $\gamma = \gamma + 1$;    (* increment counter *)
    **until** $(\gamma > R)$;    (* stopping criterion *)
    **return** $(S_{BEST})$;    (* report best state *)
**end.**
```

PERTURB and UPDATE Procedures

```
**Procedure** PERTURB($S$)
**begin**
    **for**( each $m \in M$ ) **do**
        $S' =$ SUB-MOVE($S, m$);
        $Gain(m) =$ COST($S$) - COST($S'$);
        **if** $Gain(m) >$ RANDOM($p, 0$)
            $S = S'$;
    $S =$ MAKE-STATE($S$);
    **return** $S$;
**end.**

**Procedure** **UPDATE**($p, C_{pre}, C_{cur}$)
**begin**
    **if** $C_{pre} = C_{cur}$ **then**
        $p = p - 1$;
    **else**
        $p = p_0$;
**end.**
```

METRIC-PARTITION Algorithm

- Input is a set V of the nodes and a set $S = S_1, S_2, ..., S_N$ of the nets.
- A metric value over $V \times V$ is computed
- Nodes in V are then partitioned into subsets
- Partitions meet area constraints and

  terminal constraints

```
Algorithm METRIC-PARTITION
begin
  for($i = 1$ to $N$) do
     CONSTRUCT-ST( $S_i$ );
        ADD-EDGES( $S_i, L$);
  SORT-ASCENDING($L, metric$);
  $no\_groups$ = INITIALIZE-GROUPS( $V$ );
  while( $L \neq \phi$ ) do
```

METRIC-PARTITION Algorithm

```
  $e_{ij}$ = SELECT-EDGE($L$);
  **if**( $(G_i \neq G_j)$ **and**
      (AREA($G_i$) + AREA($G_j$) $\leq A$) **and**
      (COUNT($G_i$) + COUNT($G_j$) $\leq T$) )
          MERGE-GROUPS( $G_i$, $G_j$ );
          $no\_groups = no\_groups - 1$;
          **if**($no\_groups \leq K$)
              **return**($G$);
  **else**
      **continue**;
**while**( $no\_groups > K$) **do**
  $G_i$ = SELECT _ SMALL();
  **for**($j = 1$ to $no\_groups$) **do**
      **if**( $i \neq j$ ) **and**
          (AREA($G_i$) + AREA($G_j$) $\leq A$) **and**
          (COUNT($G_i$) + COUNT($G_j$) $\leq T$) )
              STORE-MIN( $G_j$ );
```

METRIC-PARTITION Algorithm

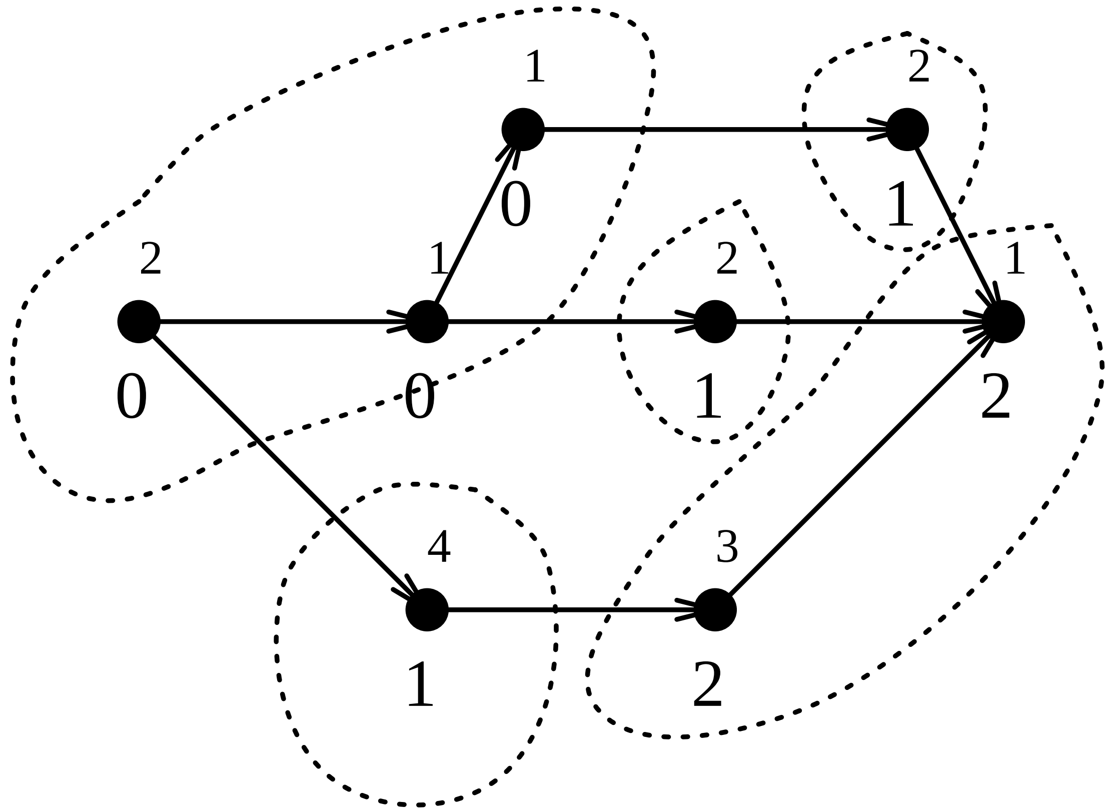
```
        $merge\_success$ = TRUE;
    MERGE-GROUPS( $G_i$, RESTORE-MIN($G_j$) );
    if( $no\_groups \leq K$ )
        return( $G$ );
    if( $merge\_success$ = FALSE )
        if( SIZE( $G_i$) = 1 )
            return( $\phi$ );
    else
        $G_j$ = SELECT _ LARGE();
        DECOMPOSE( $G_j$, $G_k$, $G_l$ );
end.
```

Lawler, Levitt and Turner (LLT) Algorithm

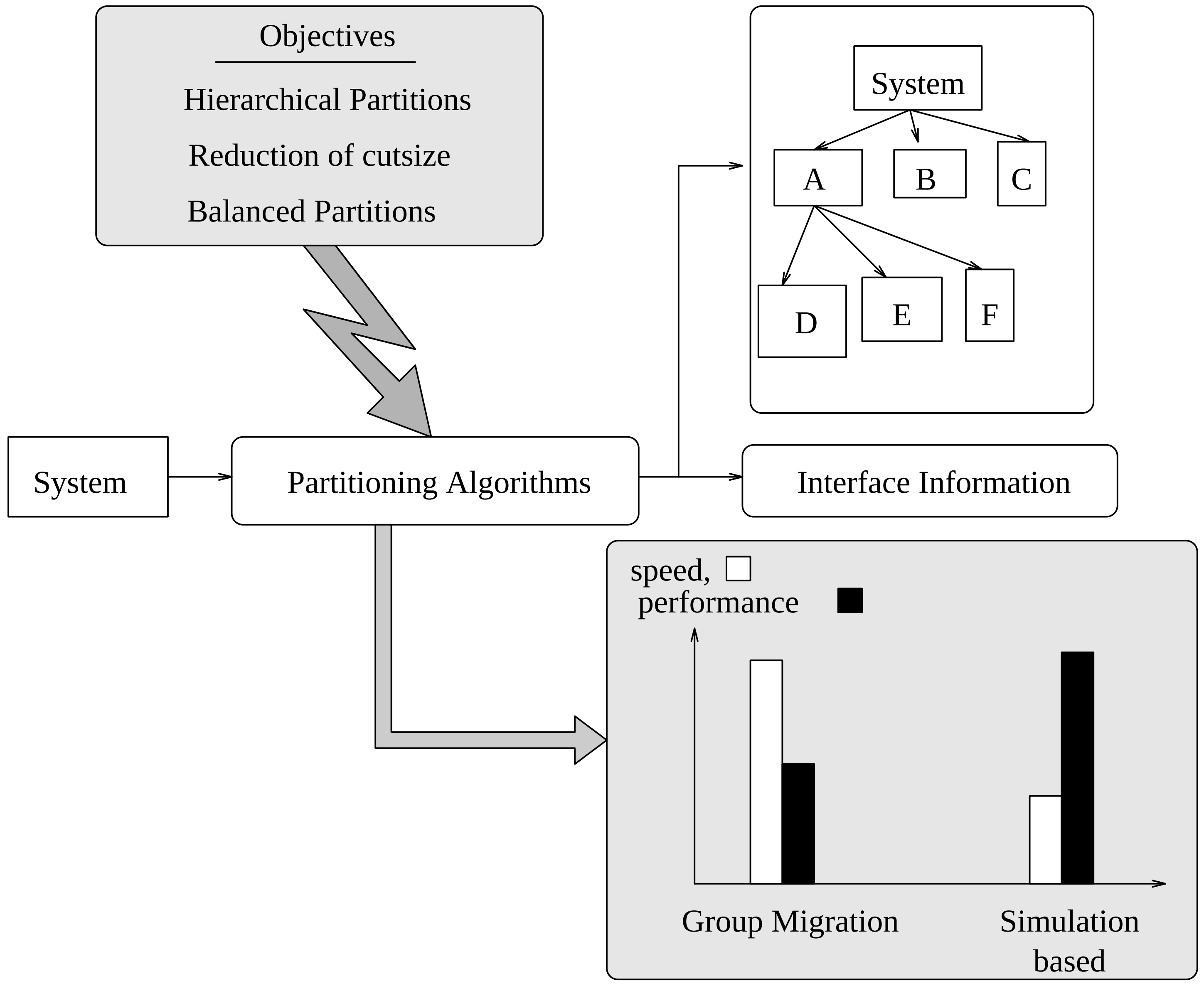
- Performance driven approach.
- Unit delay model is used.
- Circuit is represented by a directed graph.
- Each vertex has a weight indicating area occupied.
- Vertices are labelled as follows:
  - All input nodes are labelled ‘0’.
  - A node is given the same label as its predecessor if it can be accomodated.
- All nodes with same label are grouped in a cluster.
- Minimizes the maximum delay of signal under area constraints.
- Assumption: Delay within a cluster is negligible.

E. L. Lawler, K. N. Levitt and J. Turner, IEEE Transactions on Computers, 1969.

Example of LLT Alogrithm



Summary


Objectives
Hierarchical Partitions
Reduction of cutsize
Balanced Partitions
System
Partitioning Algorithms
System
A
B
C
D
E
F
Interface Information
speed, □
performance ■
Group Migration
Simulation based