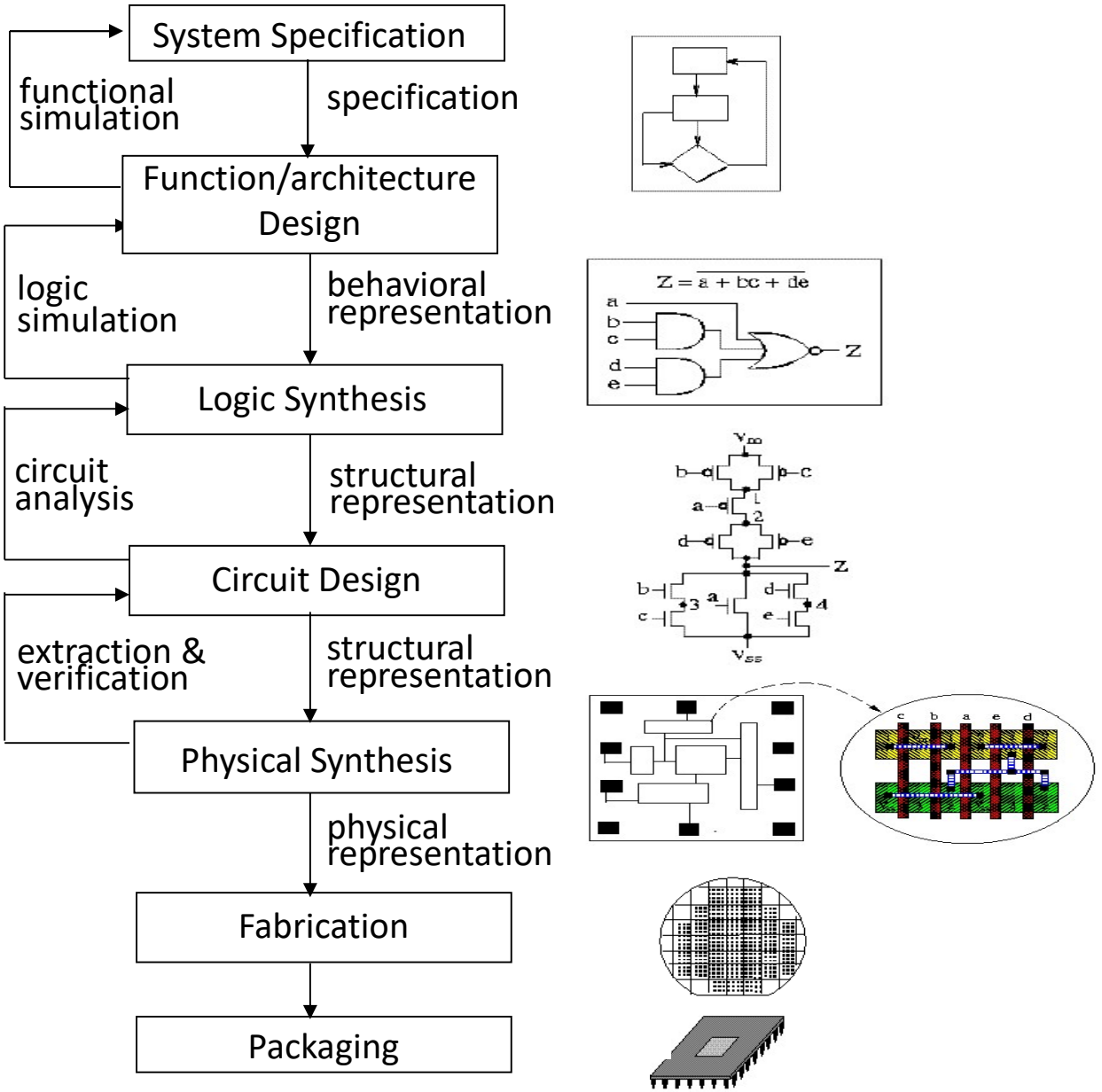
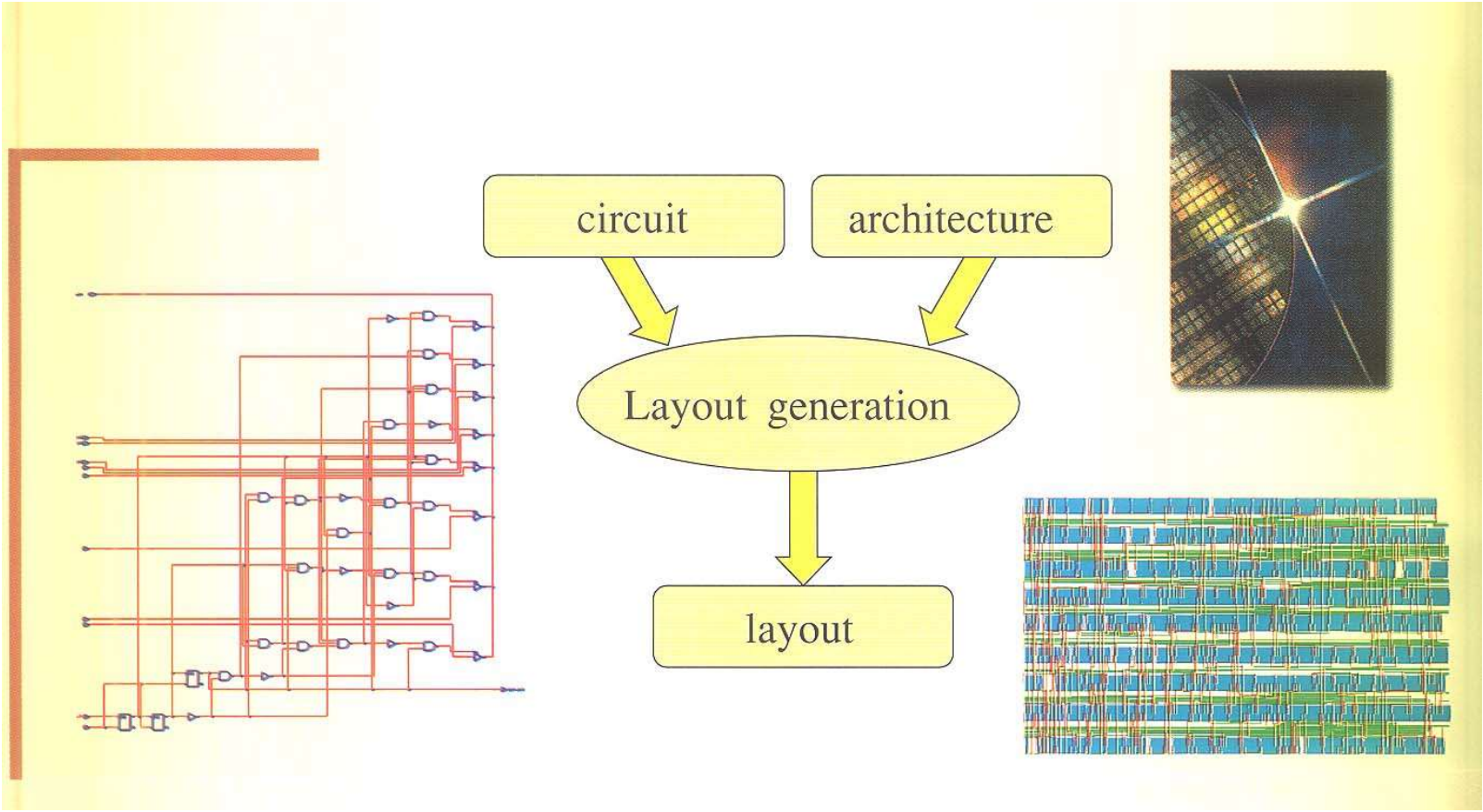
Partitioning

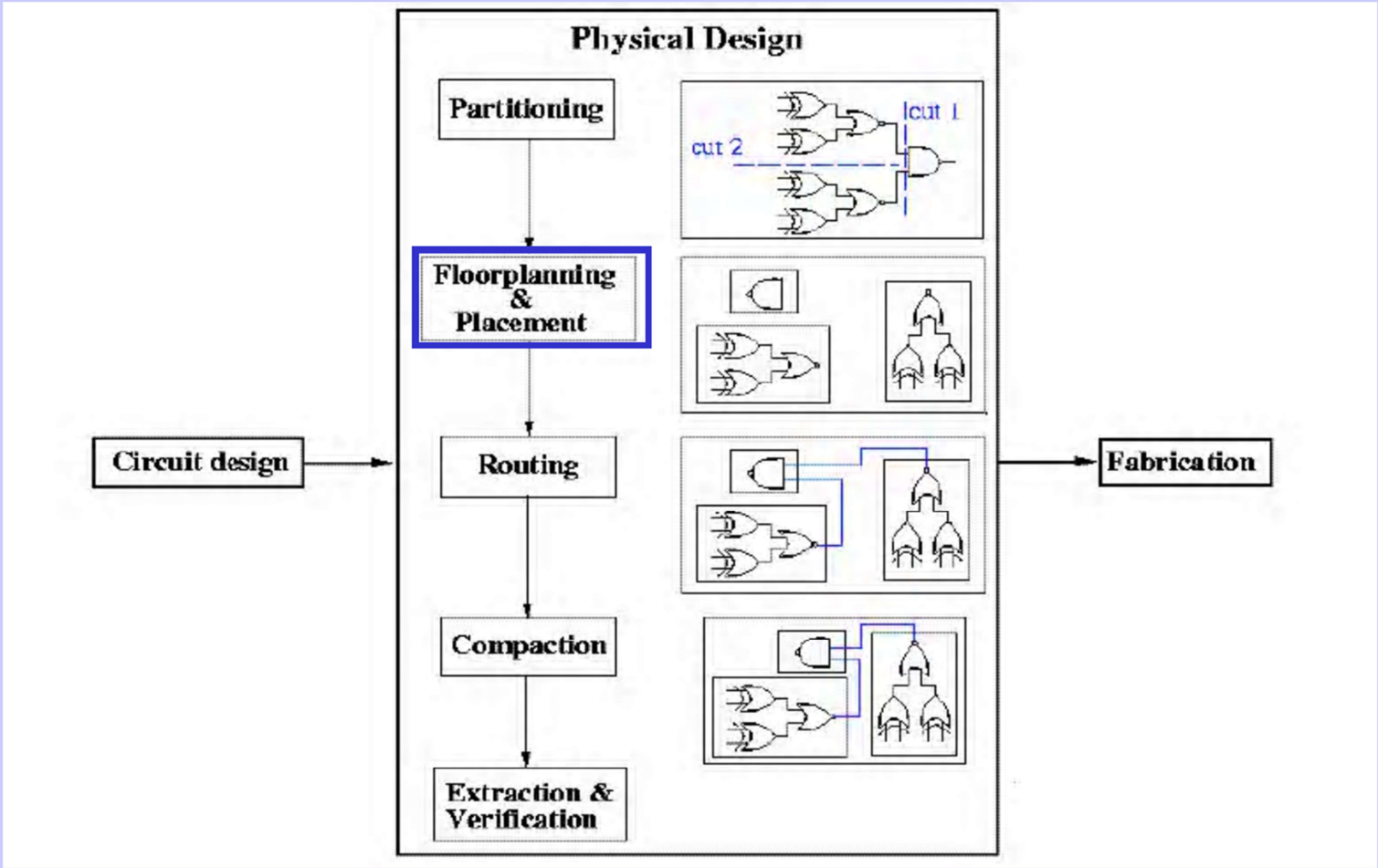
VLSI Design Flow



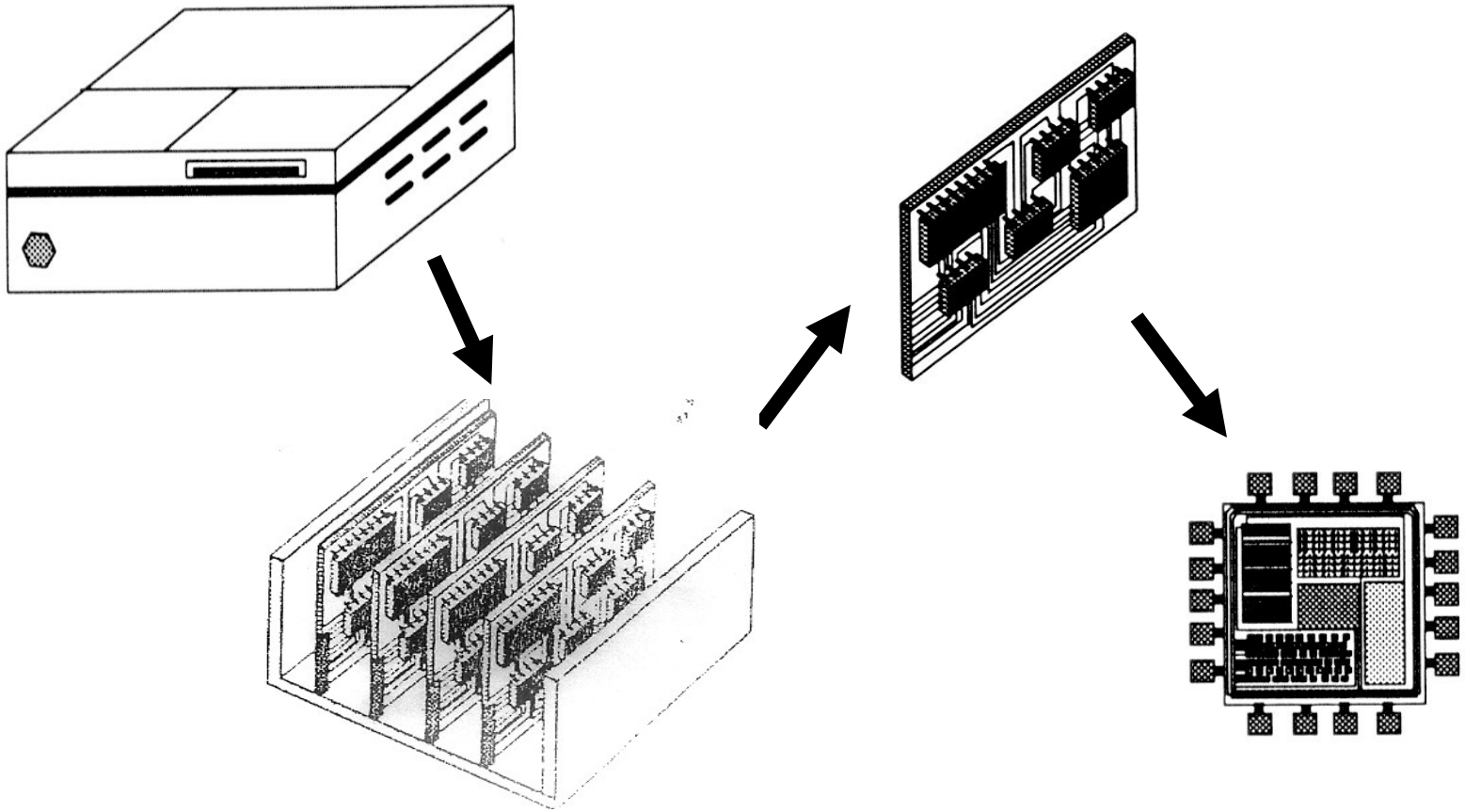
Physical Design



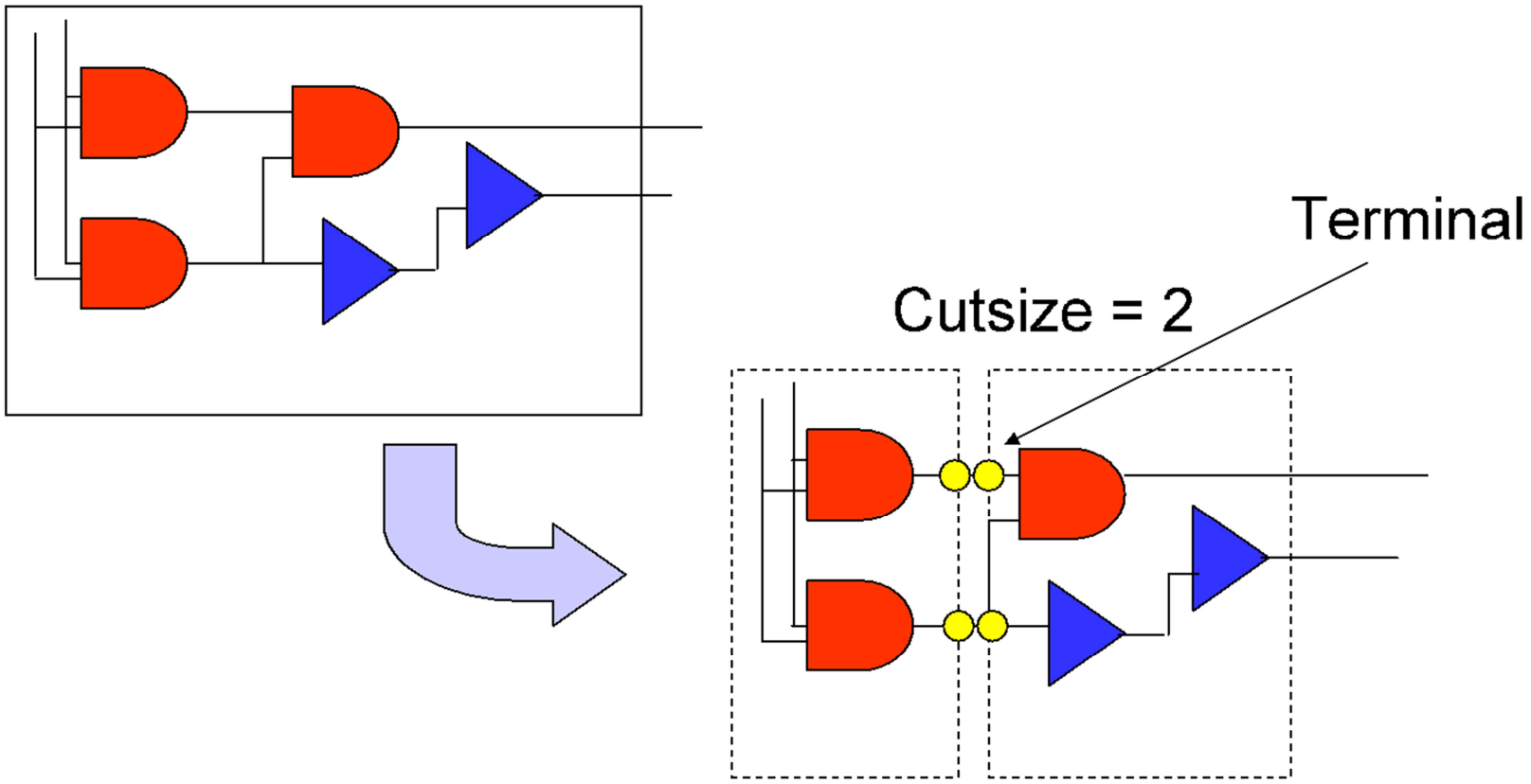
Physical Design Flow



Circuit Partitioning

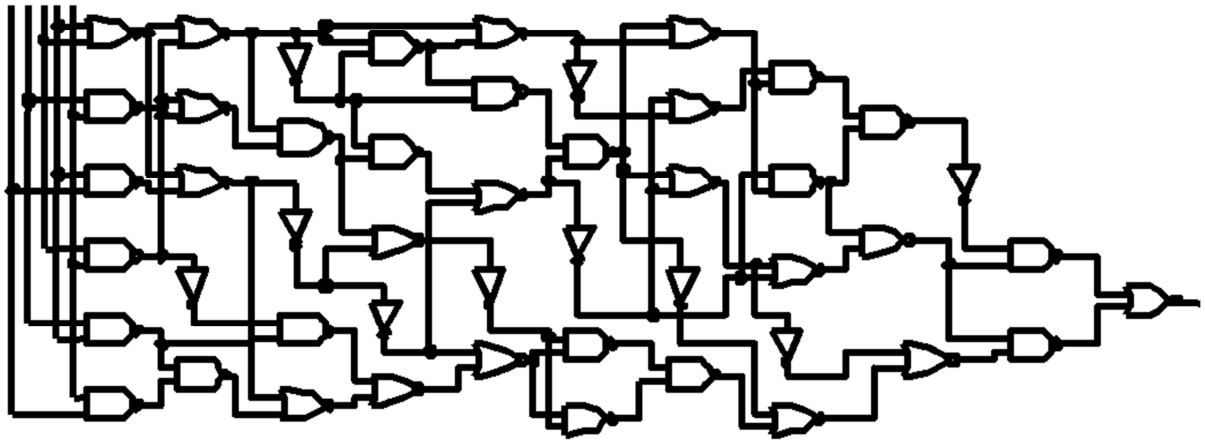


Partitioning of a Circuit: A Simple Example



Partitioning of a Circuit: Realistic Example

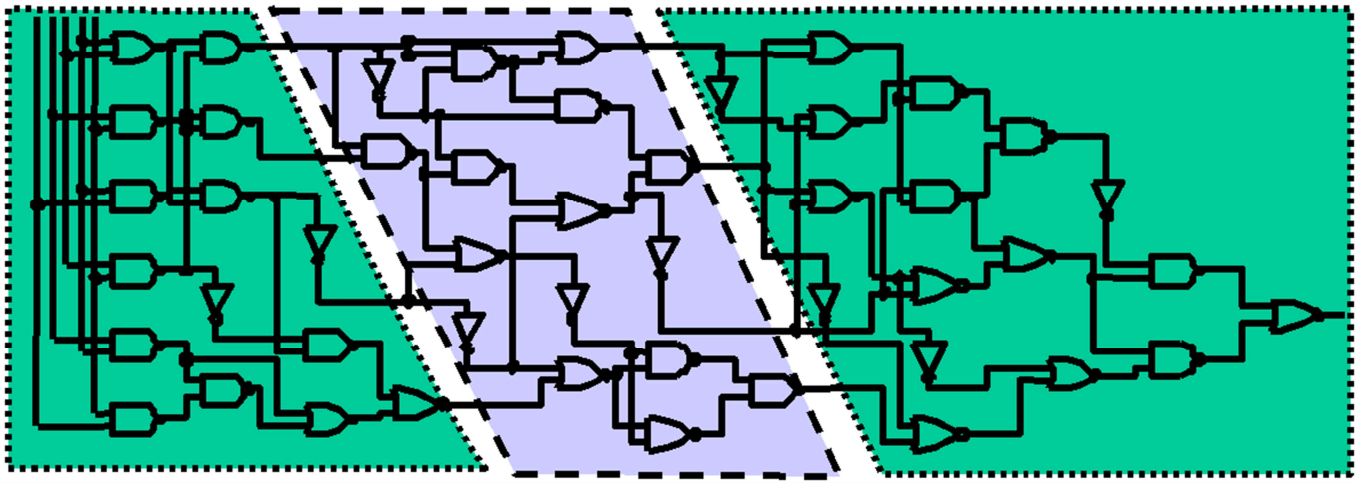
Problem Input size: 48



Cut 1=4
Size 1=15

Cut 2=4
Size 2=16

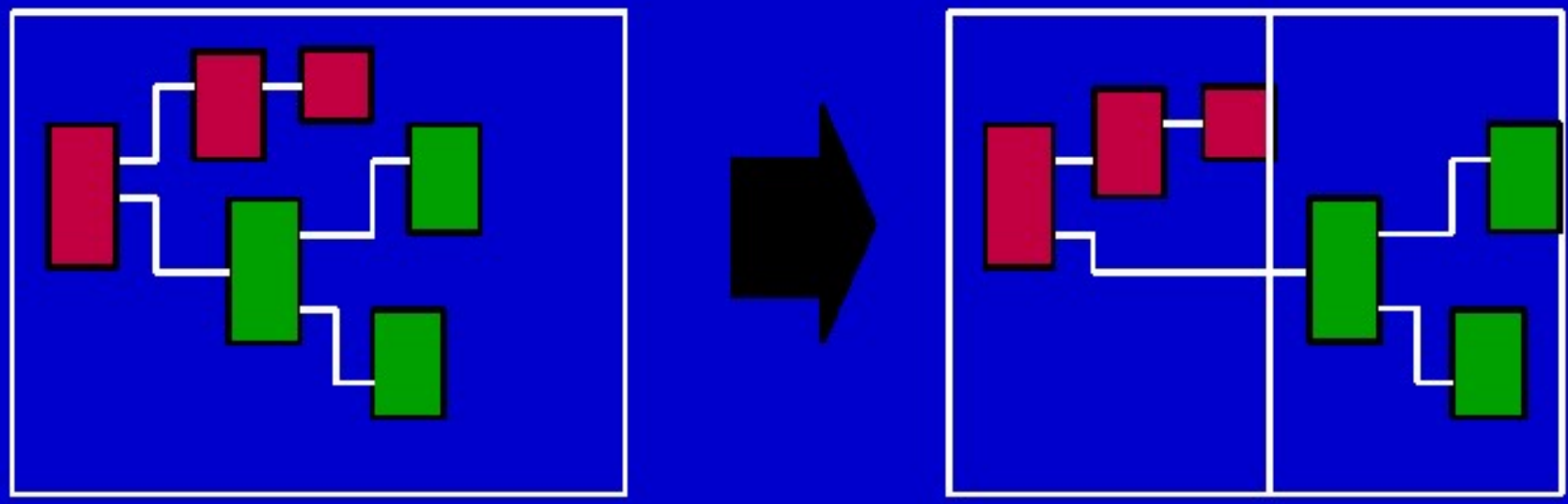
Size 3=17



Partitioning:

Objective:

Given a set of interconnected blocks, produce two sets that are of equal size, and such that the number of nets connecting the two sets is minimized.



Why Partitioning?

Partitioning-The process of decomposition of a large system into independent manageable subsystems.

Why partition a circuit ?

- Complex system with a large number of components
- Efficient design by breaking into smaller subsystems
- Each subsystem can be designed independently and concurrently
- Decompose the system so as to preserve the original functionality
- Minimize interface across the partition
- Time for decomposition should be small

The Goal: To improve the circuit performance and reduce the layout cost

Partitioning

- Decomposition of a complex system into smaller subsystems
  - –Done hierarchically
  - –Partitioning done until each subsystem has manageable size
  - –Each subsystem can be designed independently

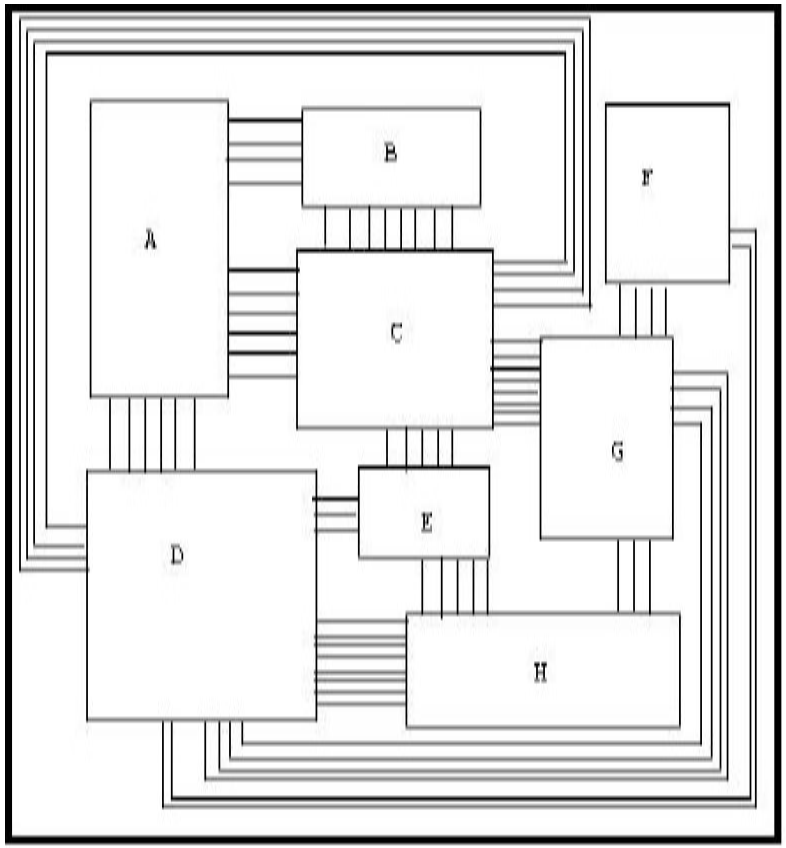
- Interconnections between partitions minimized
  - –Less hassle interfacing the subsystems
  - –Communication between subsystems usually costly

Hierarchical Partitioning

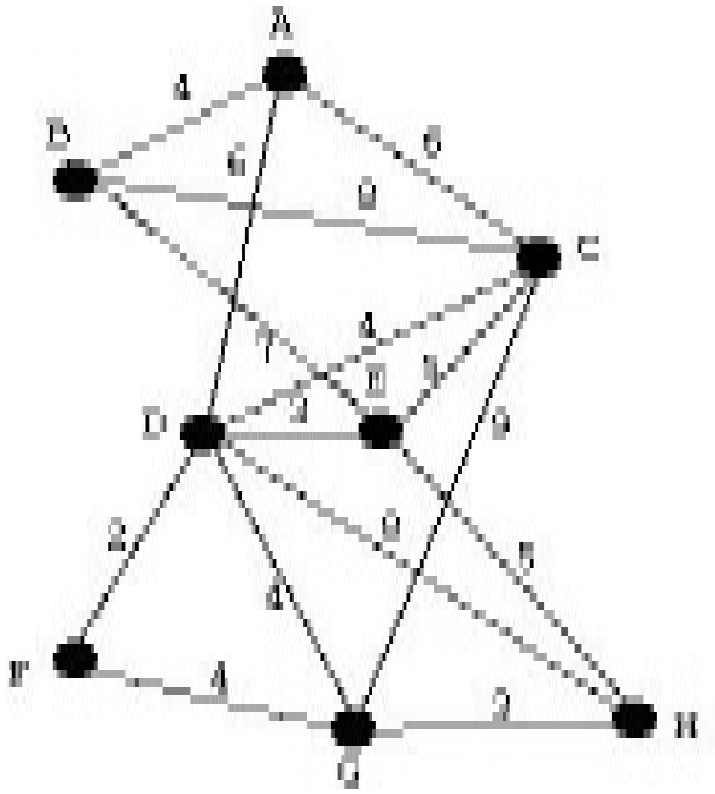
- Levels of partitioning:
  - System-level partitioning:
    Each sub-system can be designed as a single PCB
  - Board-level partitioning:
    Circuit assigned to a PCB is partitioned into sub-circuits each fabricated as a VLSI chip
  - Chip-level partitioning:
    Circuit assigned to the chip is divided into manageable sub-circuits
    NOTE: physically not necessary

VLSI Layout can be represented by a graph

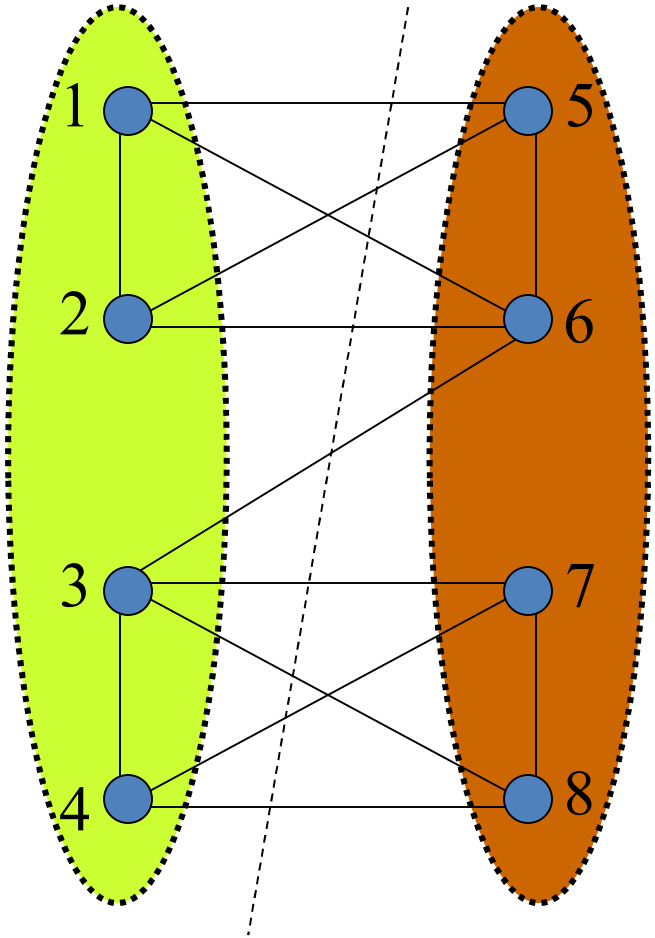
VLSI Layout



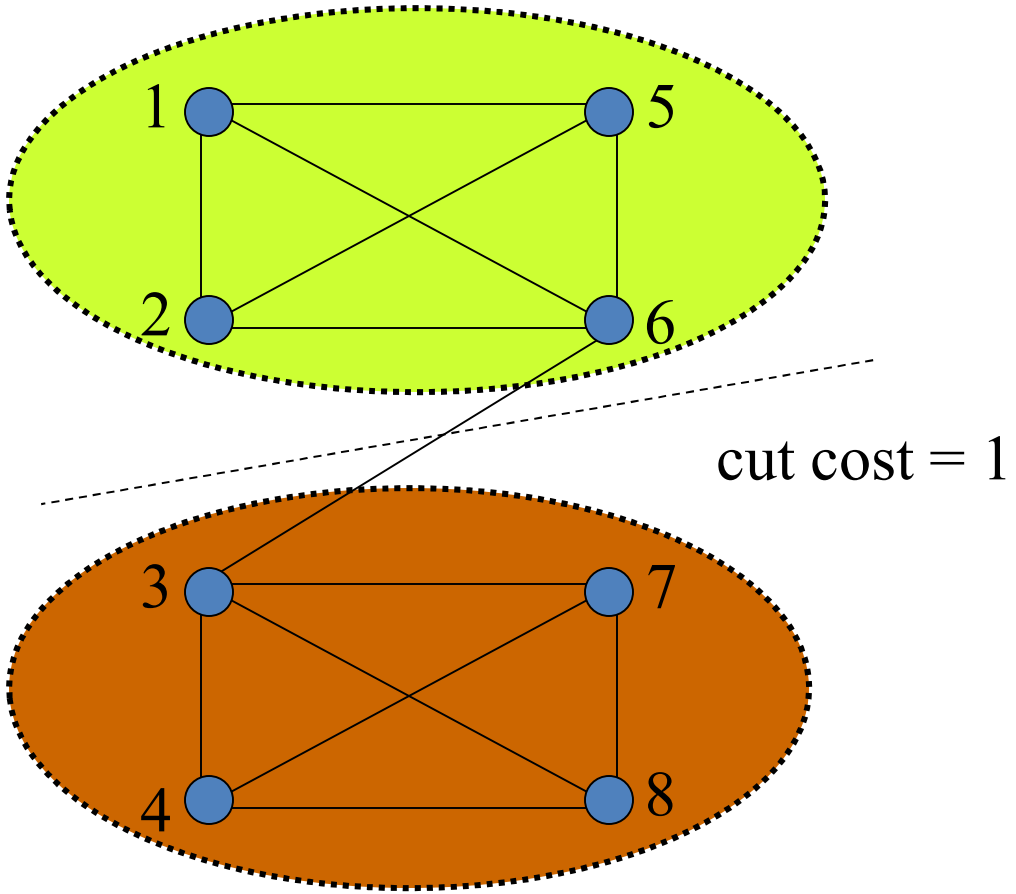
Graph Representation



Circuit Partitioning: A Graph-Theoretic Problem



cut cost = 9



cut cost = 1

Circuit Partitioning Complexity

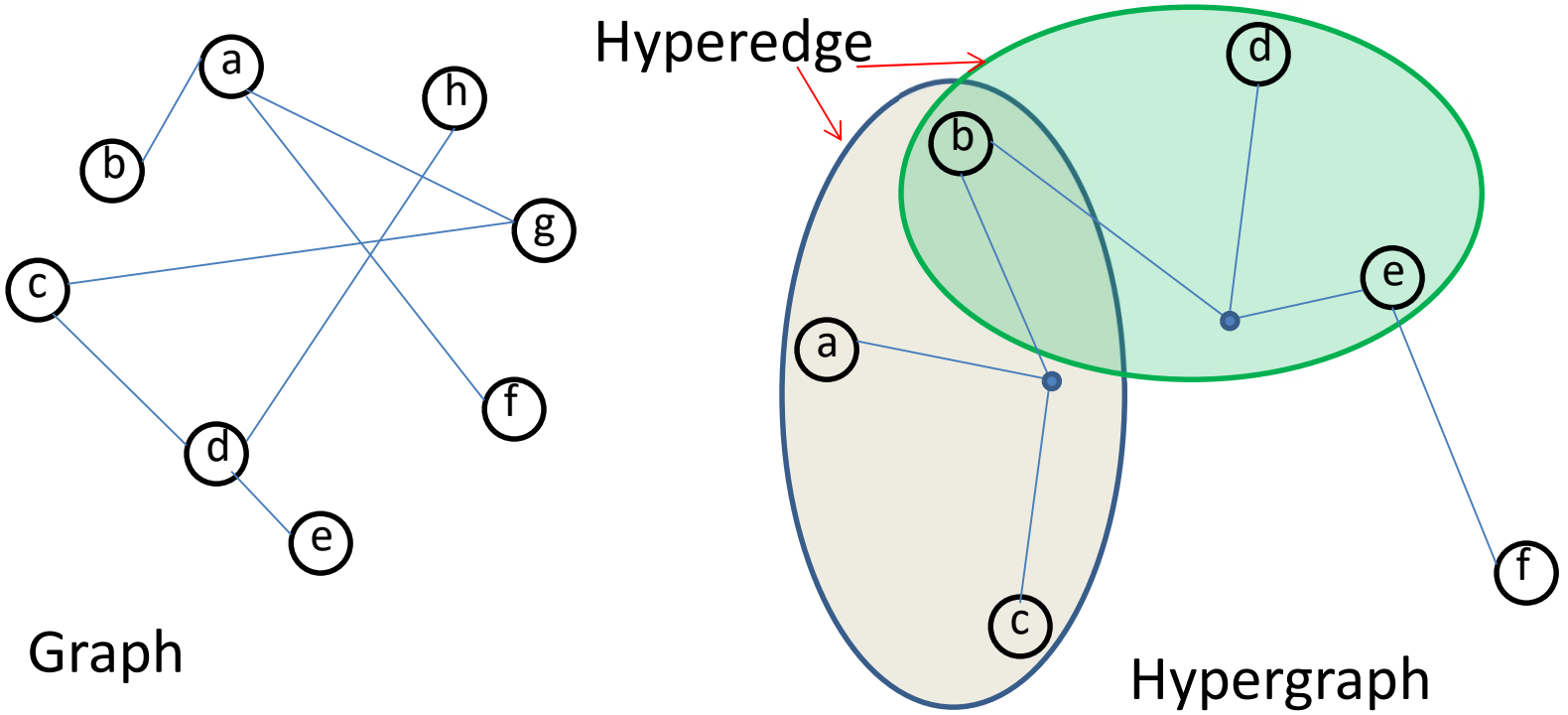
- For *2n circuit nodes*, # of possibilities = $\binom{2n}{n}$
- For a 100-gate circuit, there are 5 $\times 10^{28}$ possibilities;

  It will take around 1.59 x 10^{13} years if one checks

  100 M possibilities per second.

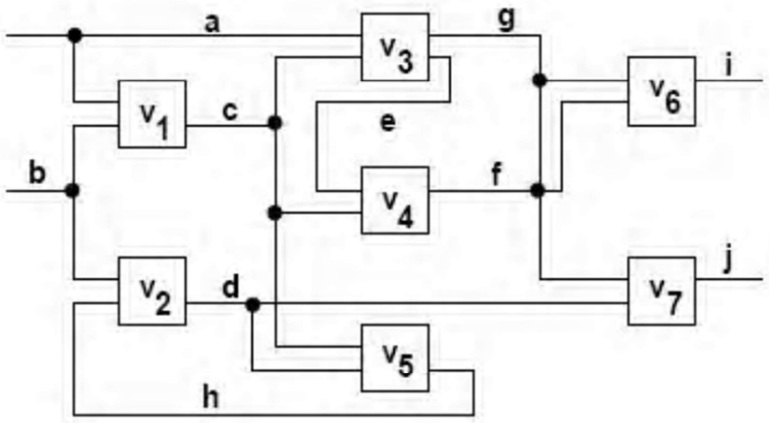
Graphs and Hypergraphs

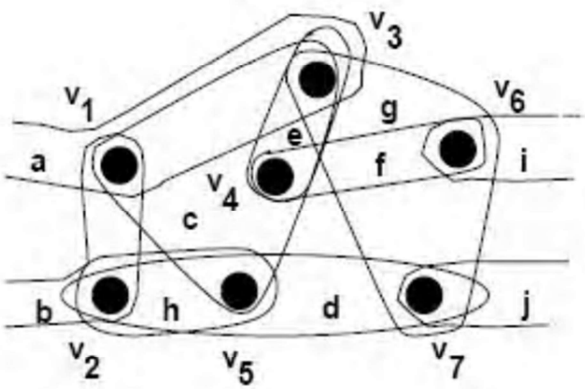
The input Circuit of a VLSI Physical Design problem is a hypergraph



Hyper-Graph Representations

Definition: A *hyper-edge* is an interconnection of a set of (> 2) distinct vertices.





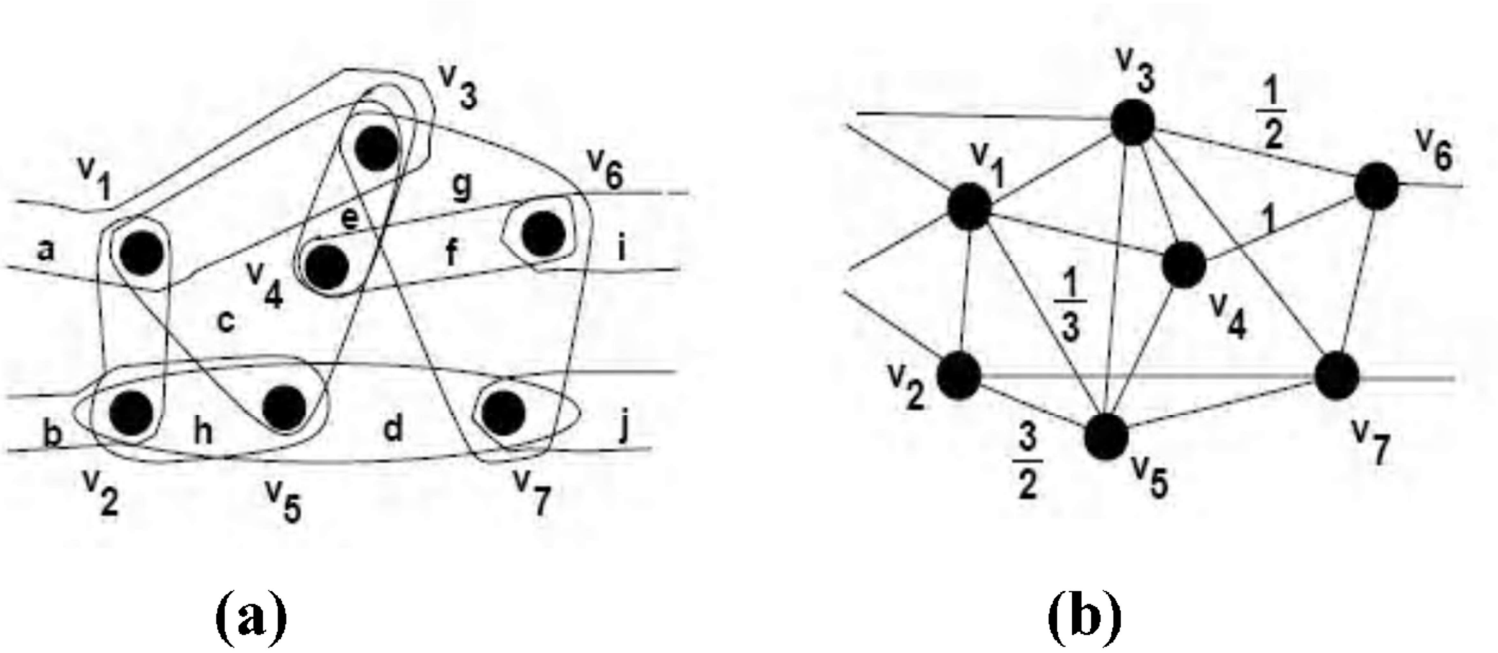
(a) **(b)**

A circuit with 7 modules and 10 signal nets:

(a) circuit diagram with all inputs on the left side of the modules and all outputs on the right side
(b) the hyper-graph representation

It is difficult to design efficient algorithms based on hypergraph

Hyper-Graph to Graph Representations



(a) the hyper-graph representation
(b) the weighted graph representation using the standard clique net model with uniform edge weight

Several algorithm exist to represent hyperedges with graph edges, but most effective one is still unknown

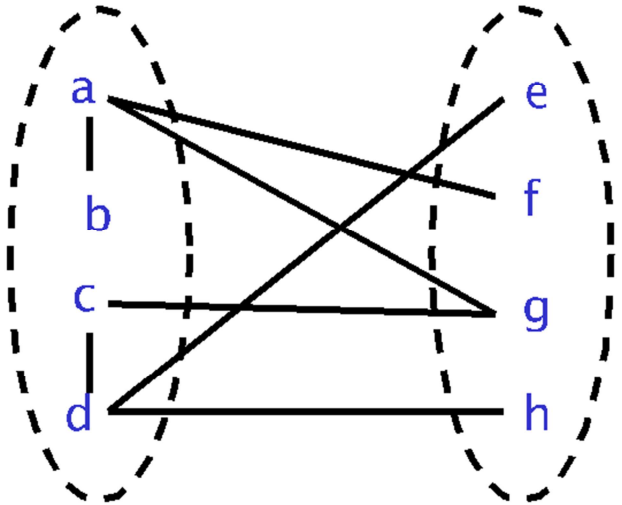
Partitioning: Formal Definition

- Input
  - Graph or hypergraph
  - Usually with vertex weights (sizes)
  - Usually weighted edges
- Constraints
  - Number of partitions (K-way partitioning)
  - Maximum capacity of each partition
    OR
    maximum allowable difference between partitions
- Objective
  - Assign nodes to partitions subject to constraints
    s.t. the cutsize is minimized
- Tractability
  - Is NP-complete

Kernighan-Lin (KL) Algorithm

- On non-weighted graphs
- An iterative improvement technique
- A two-way (bisection) partitioning algorithm
- The partitions must be balanced (of equal size)
- Iterate as long as the cutsize improves:
  - Find a pair of vertices that result in the largest decrease in cutsize if exchanged
  - Exchange the two vertices (potential move)
  - “Lock” the vertices
  - If no improvement possible, and still some vertices unlocked, then exchange vertices that result in smallest increase in cutsize

Kernighan-Lin (KL) Example



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

Kernighan-Lin (KL) Algorithm

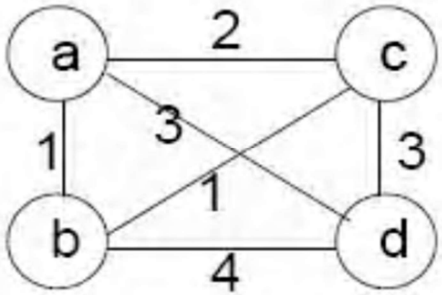
- Initialize
  - –Bipartition G into $V_1$and V_2, s.t., $|V_1| = |V_2| \pm 1$
  - –n = |V|
- Repeat
  - –for i=1 to n/2
    - Find a pair of unlocked vertices $v_{ai} \in V_1$ and $v_{bi} \in V_2$ whose exchange makes the *largest decrease or smallest increase in cut-cost (gain)*
    - *Mark v_{ai} and v_{bi} as locked*
    - *Store the gain g_i.*
  - *–Find k, s.t. $\Sigma_{i=1..k} g_i = Gain_k$ is maximized*
  - *–If $Gain_k > 0$ then*

    move $v_{a1},...,v_{ak}$ from V_1 to V_2 and $v_{b1},...,v_{bk}$ from V_2 to V_1.
- *Until Gaink≤0*

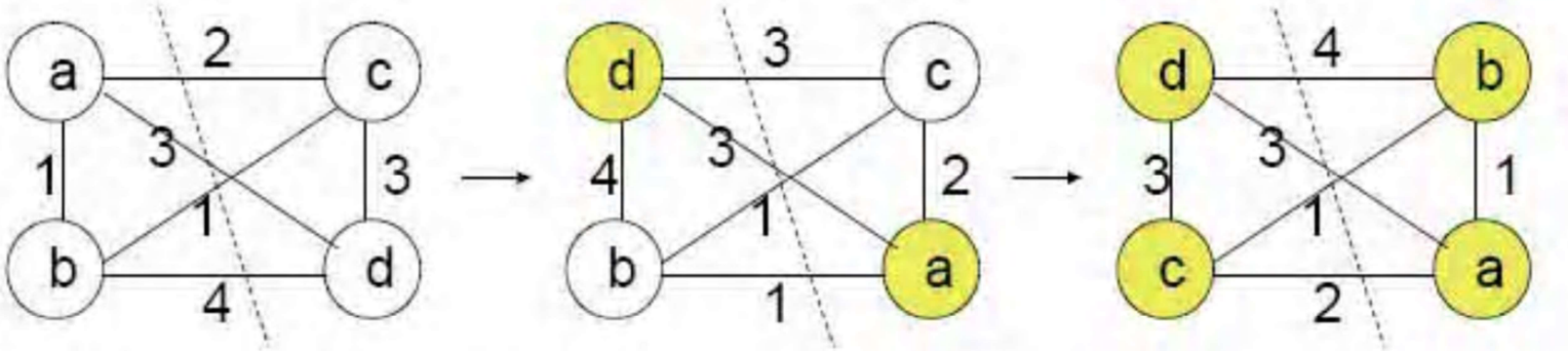
Kernighan-Lin (KL) Example

Example with Edge Weights

	a	b	c	d
a	0	1	2	3
b	1	0	1	4
c	2	1	0	3
d	3	4	3	0



Kernighan-Lin (KL) Example- Pass One

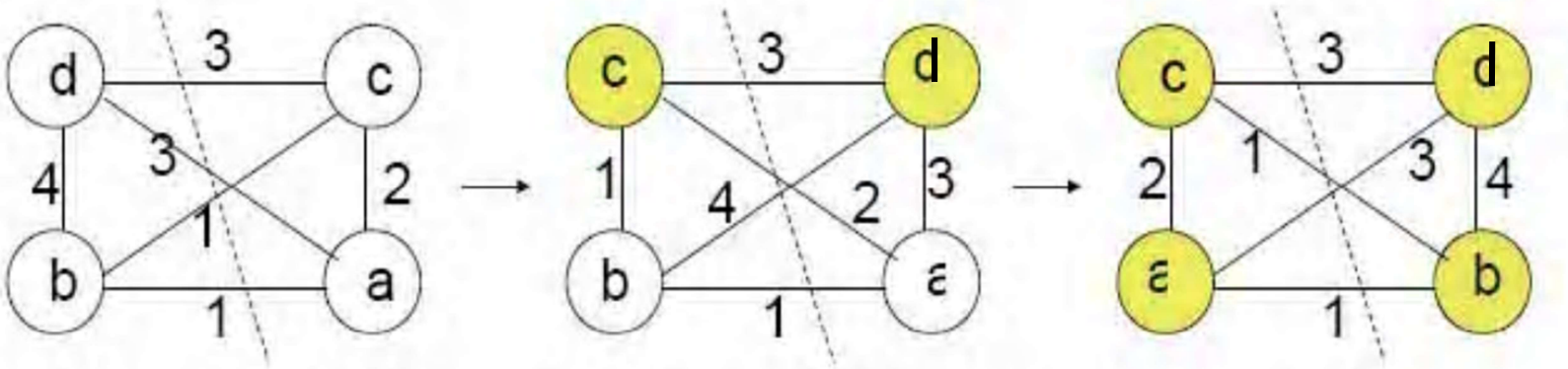


g(a,c) = -1+3-3+1 = 0
g(a,d) = -1+2-3+4 = 2
g(b,c) = -1+4-3+2 = 2
g(b,d) = -1+1-3+3 = 0
$g_1 = 2$

g(b,c) = -4+1-2+3 = -2
$g_2 = -2$

$\therefore G = g_1 = 2 \quad (k = 1)$

Kernighan-Lin (KL) Example- Pass Two



g(a,b) = -2+3-4+1 = -2
g(a,d) = -2+1-4+3 = -2
g(c,b) = -2+3-4+1 = -2
g(c,d) = -2+1-4+3 = -2
$g_1 = -2$

g(a,b) = -3+2-1+4 = 2
$g_2 = 2$

$G = g_1 + g_2 = 0 \quad (k = 2)$

STOP!

Kernighan-Lin (KL) Algorithm: Time Complexity

- Initialize –Bipartition G into $V_1$and V_2, s.t., $|V_1| = |V_2| \pm 1$
  - –n = |V|
- Repeat –for i=1 to n/2
  - Find a pair of unlocked vertices $v_{ai} \in V_1$ and $v_{bi} \in V_2$ whose exchange makes the *largest decrease or smallest increase in cut-cost (gain)*
  - *Mark v_{ai} and v_{bi} as locked*
  - *Store the gain g_i.*
  - *–Find k, s.t. $\Sigma_{i=1..k} g_i = Gain_k$ is maximized*
  - *–If $Gain_k > 0$ then*
  - *move $v_{a1},\ldots,v_{ak}$ from V_1 to V_2 and $v_{b1},\ldots,v_{bk}$ from V_2 to V_1.*
- *Until Gaink≤0*

–Inner (for) loop
- Iterates n/2 times
- Iteration 1: (n/2) x (n/2)
- Iteration i: $(n/2 – i + 1)^2$.

–Passes? Usually independent of n

–$O(n^3)$

Kernighan-Lin (KL) : Drawbacks?

- Local optimum
- Balanced partitions only
- No weight for the vertices
- High time complexity

Difficult to tackle the problems of Hyper-edges and weighted edges