EE 5301 – VLSI Design Automation I

Part III: Partitioning

Kia Bazargan

University of Minnesota

References and Copyright

- Textbooks referred (none required)
  - [Mic94] G. De Micheli
    "Synthesis and Optimization of Digital Circuits"
    McGraw-Hill, 1994.
  - [CLR90] T. H. Cormen, C. E. Leiserson, R. L. Rivest
    "Introduction to Algorithms"
    MIT Press, 1990.
  - [Sar96] M. Sarrafzadeh, C. K. Wong
    "An Introduction to VLSI Physical Design"
    McGraw-Hill, 1996.
  - [She99] N. Sherwani
    "Algorithms For VLSI Physical Design Automation"
    Kluwer Academic Publishers, 3rd edition, 1999.

References and Copyright (cont.)

- Slides used: (*Modified by Kia when necessary*)
  - [©Sarrafzadeh] © Majid Sarrafzadeh, 2001;
    Department of Computer Science, UCLA
  - [©Sherwani] © Naveed A. Sherwani, 1992
    (companion slides to [She99])
  - [©Keutzer] © Kurt Keutzer, Dept. of EECS,
    UC-Berekeley
    http://www-cad.eecs.berkeley.edu/~niraj/ee244/index.htm
  - [©Gupta] © Rajesh Gupta
    UC-Irvine
    http://www.ics.uci.edu/~rgupta/ics280.html
  - [©Kang] © Steve Kang
    UIUC
    http://www.ece.uiuc.edu/ece482/

Partitioning

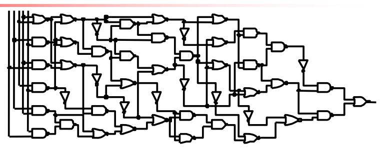
- Decomposition of a complex system into smaller subsystems
  - Done hierarchically
  - Partitioning done until each subsystem has manageable size
  - Each subsystem can be designed independently
- Interconnections between partitions minimized
  - Less hassle interfacing the subsystems
  - Communication between subsystems usually costly

[©Sherwani]

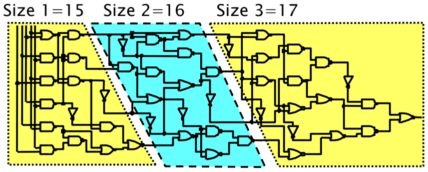
Fall 2005 EE 5301 - VLSI Design Automation I III-4

Example: Partitioning of a Circuit

Input size: 48



Cut 1=4 Cut 2=4
Size 1=15 Size 2=16 Size 3=17



[©Sherwani]

Fall 2005 EE 5301 - VLSI Design Automation I III-5

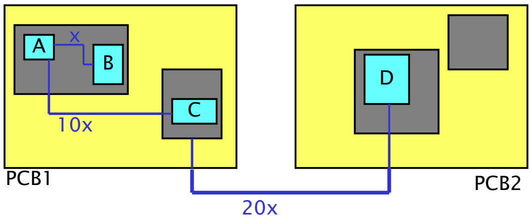
Hierarchical Partitioning

- 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

[©Sherwani]

Fall 2005 EE 5301 - VLSI Design Automation I III-6

Delay at Different Levels of Partitions



[©Sherwani]

Fall 2005 EE 5301 - VLSI Design Automation I III-7

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 ☹

Fall 2005 EE 5301 - VLSI Design Automation I III-8

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

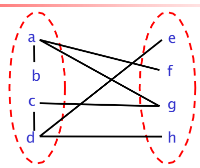
W. Kernighan and S. Lin, Bell System Technical Journal, 1970.

Fall 2005 EE 5301 - VLSI Design Automation I III-9

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
    - Mark v_{ai} and v_{bi} as locked
    - Store the gain g_i.
  - Find k, s.t. $\sum_{i=1..k} g_i = Gain_k$ 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 $Gain_k \leq 0$

Kernighan-Lin (KL) Example



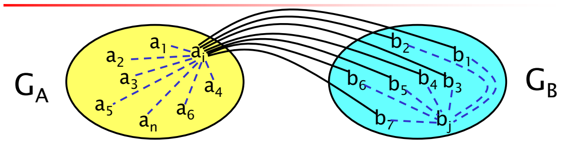
Step No.	Vertex Pair	Gain	Cut-cost
0	--	0	5
1	{ d, g }	3	2
2	{ c, f }	1	1
3	{ b, h }	-2	3
4	{ a, e }	-2	5

[©Sarrafzadeh]

Kernighan-Lin (KL) : Analysis

- Time complexity?
  - 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
  - $O(n^3)$
- Drawbacks?
  - Local optimum
  - Balanced partitions only
  - No weight for the vertices
  - High time complexity
  - Hyper-edges? Weighted edges?

Gain Calculation



$$I_{a_i} = \sum_{x \in A} C_{a_i x}, \qquad E_{a_i} = \sum_{y \in B} C_{a_i y}$$

Likewise,
$$D_{a_i} = E_{a_i} - I_{a_i}$$
$$D_{b_j} = E_{b_j} - I_{b_j} = \sum_{x \in A} C_{b_j x} - \sum_{y \in B} C_{b_j y}$$

[©Kang]

Gain Calculation (cont.)

- Lemma: Consider any $a_i \in A$, $b_j \in B$.
  If a_i, b_j are interchanged, the gain is
  $$g = D_{a_i} + D_{b_j} - 2C_{a_i b_j}$$
- Proof:
  Total cost before interchange (T) between A and B
  $$T = E_{a_i} + E_{b_j} - C_{a_i b_j} + \text{(cost for all others)}$$
  Total cost after interchange (T') between A and B
  $$T = I_{a_i} + I_{b_j} + C_{a_i b_j} + \text{(cost for all others)}$$
  Therefore
  $$g = T - T' = \overbrace{E_{a_i} - I_{a_i}}^{D_{a_i}} + \overbrace{E_{b_j} - I_{b_j}}^{D_{b_j}} - 2C_{a_i b_j}$$

[©Kang]

Gain Calculation (cont.)

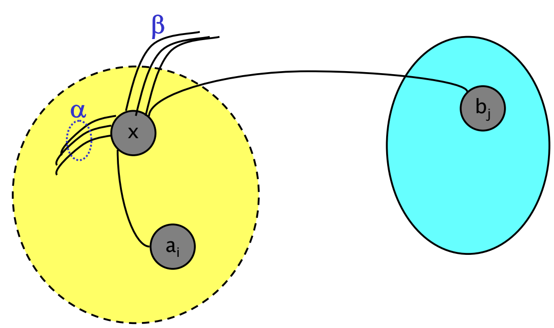
- Lemma:
  - Let D_x', D_y' be the new D values for elements of $A - \{a_i\}$ and $B - \{b_j\}$. Then after interchanging a_i & b_j,

$$D_x' = D_x + 2C_{xa_i} - 2C_{xb_j}, \quad x \in A - \{a_i\}$$
$$D_y' = D_y + 2C_{yb_j} - 2C_{ya_i}, \quad y \in B - \{b_j\}$$

- Proof:
  - The edge x-a_i changed from internal in D_x to external in D_x'
  - The edge y-bj changed from internal in D_x to external in D_x'
  - The x-b_j edge changed from external to internal
  - The y-a_i edge changed from external to internal
- More clarification in the next two slides

[©Kang]

Clarification of the Lemma



Clarification of the Lemma (cont.)

- Decompose Ix and Ex to separate edges from ai and bj:

$$I_x = C_{xa_i} + \alpha \qquad E_x = C_{xb_j} + \beta$$

- Write the equations before the move

$$D_x = E_x - I_x = (C_{xb_j} + \beta) - (C_{xa_i} + \alpha)$$

$$= -\alpha + \beta - C_{xa_i} + C_{xb_j}$$

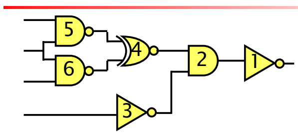
- ... And after the move

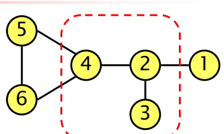
$$I_x' = C_{xb_j} + \alpha \qquad E_x' = C_{xa_i} + \beta$$

$$D_x' = -\alpha + \beta + C_{xa_i} - C_{xb_j}$$

$$= D_x + 2C_{xa_i} - 2C_{xb_j}$$

Example: KL





- Step 1 - Initialization

  A = {2, 3, 4}, B = {1, 5, 6}

  A' = A = {2, 3, 4}, B' = B = {1, 5, 6}

- Step 2 - Compute D values

  $D_1 = E_1 - I_1 = 1-0 = +1$
  $D_2 = E_2 - I_2 = 1-2 = -1$
  $D_3 = E_3 - I_3 = 0-1 = -1$
  $D_4 = E_4 - I_4 = 2-1 = +1$
  $D_5 = E_5 - I_5 = 1-1 = +0$
  $D_6 = E_6 - I_6 = 1-1 = +0$

[©Kang]

Example: KL (cont.)

- Step 3 - compute gains

  $g_{21} = D_2 + D_1 - 2C_{21} = (-1) + (+1) - 2(1) = -2$
  $g_{25} = D_2 + D_5 - 2C_{25} = (-1) + (+0) - 2(0) = -1$
  $g_{26} = D_2 + D_6 - 2C_{26} = (-1) + (+0) - 2(0) = -1$

  $g_{31} = D_3 + D_1 - 2C_{31} = (-1) + (+1) - 2(0) = 0$
  $g_{35} = D_3 + D_5 - 2C_{35} = (-1) + (0) - 2(0) = -1$
  $g_{36} = D_3 + D_6 - 2C_{36} = (-1) + (0) - 2(0) = -1$

  $g_{41} = D_4 + D_1 - 2C_{41} = (+1) + (+1) - 2(0) = +2$
  $g_{45} = D_4 + D_5 - 2C_{45} = (+1) + (+0) - 2(+1) = -1$
  $g_{46} = D_4 + D_6 - 2C_{46} = (+1) + (+0) - 2(+1) = -1$

- The largest g value is $g_{41} = +2$

  ⇒interchange 4 and 1 $(a_1, b_1) = (4, 1)$
  A' = A' - {4} = {2, 3}
  B' = B' - {1} = {5, 6} both not empty

[©Kang]

Example: KL (cont.)

- Step 4 - update D values of node connected to vertices (4, 1)

  $D_2' = D_2 + 2C_{24} - 2C_{21} = (-1) + 2(+1) - 2(+1) = -1$
  $D_5' = D_5 + 2C_{51} - 2C_{54} = +0 + 2(0) - 2(+1) = -2$
  $D_6' = D_6 + 2C_{61} - 2C_{64} = +0 + 2(0) - 2(+1) = -2$

- Assign $D_i = D_i'$, repeat step 3 :

  $g25 = D_2 + D_5 - 2C_{25} = -1 - 2 - 2(0) = -3$
  $g26 = D_2 + D_6 - 2C_{26} = -1 - 2 - 2(0) = -3$
  $g35 = D_3 + D_5 - 2C_{35} = -1 - 2 - 2(0) = -3$
  $g36 = D_3 + D_6 - 2C_{36} = -1 - 2 - 2(0) = -3$

- All values are equal;
  arbitrarily choose $g_{36} = -3$ ⇒ (a2, b2) = (3, 6)

  A' = A' - {3} = {2}, B' = B' - {6} = {5}

  New D values are:

  $D_2' = D_2 + 2C_{23} - 2C_{26} = -1 + 2(1) - 2(0) = +1$
  $D_5' = D_5 + 2C_{56} - 2C_{53} = -2 + 2(1) - 2(0) = +0$

- New gain with $D_2 \leftarrow D_2'$, $D_5 \leftarrow D_5'$

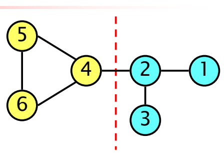
  $g_{25} = D_2 + D_5 - 2C_{52} = +1 + 0 - 2(0) = +1$ ⇒ (a3, b3) = (2, 5)

[©Kang]

Example: KL (cont.)

- Step 5 - Determine the # of moves to take

  $g_1 = +2$
  $g_1 + g_2 = +2 - 3 = -1$
  $g_1 + g_2 + g_3 = +2 - 3 + 1 = 0$



- The value of k for max G is 1

  $X = \{a_1\} = \{4\}$, $Y = \{b_1\} = \{1\}$

- Move X to B, Y to A ⇒ A = {1, 2, 3}, B = {4, 5, 6}
- Repeat the whole process:

  • • • • •

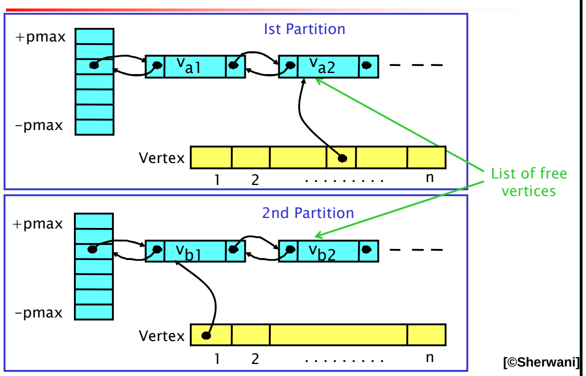
- The final solution is A = {1, 2, 3}, B = {4, 5, 6}

Fiduccia-Mattheyses (FM) Algorithm

- Modified version of KL
- A single vertex is moved across the cut in a single move
  - ➔ Unbalanced partitions
- Vertices are weighted
- Concept of cutsize extended to hypergraphs
- Special data structure to improve time complexity to $O(n^2)$
  - (Main feature)
- Can be extended to multi-way partitioning

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

The FM Algorithm: Data Structure



[©Sherwani]

The FM Algorithm: Data Structure

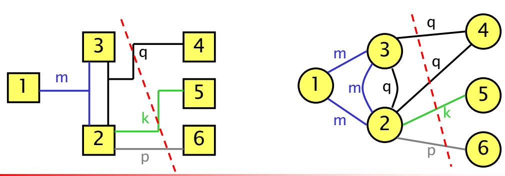
- Pmax
  - Maximum gain
  - $p_{max} = d_{max} \cdot w_{max}$, where
    d_{max} = max degree of a vertex (# edges incident to it)
    w_{max} is the maximum edge weight
  - What does it mean intuitively?
- -Pmax .. Pmax array
  - Index i is a pointer to the list of unlocked vertices with gain i.
- Limit on size of partition
  - A maximum defined for the sum of vertex weights in a partition
    (alternatively, the maximum ratio of partition sizes might be defined)

The FM Algorithm

- Initialize
  - Start with a balance partition A, B of G
    (can be done by sorting vertex weights in decreasing order, placing them in A and B alternatively)
- Iterations
  - Similar to KL
  - A vertex cannot move if violates the balance condition
  - Choosing the node to move:
    pick the max gain in the partitions
  - Moves are tentative (similar to KL)
  - When no moves possible or no more unlocked vertices available, the pass ends
  - When no move can be made in a pass, the algorithm terminates

Why Hyperedges?

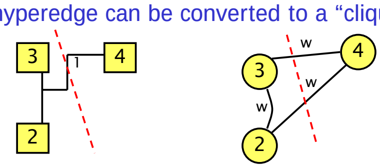
- For multi terminal nets, K-L may decompose them into many 2-terminal nets, but not efficient!
- Consider this example:
- If A = {1, 2, 3} B = {4, 5, 6}, graph model shows the cutsize = 4 but in the real circuit, only 3 wires cut
- Reducing the number of nets cut is more realistic than reducing the number of edges cut



[©Kang]

Hyperedge to Edge Conversion

- A hyperedge can be converted to a "clique".



"Real" cut=1 "net" cut=2w

- w=?
  - w=2/(n-1) has been used, also w=2/n
  - Best: $w=4/(n^2 - \text{mod}(n,2))$
    for n=3, w=4/(9-1)=0.5
- Always necessary to convert hyper-edge to edge?

[©Keutzer]

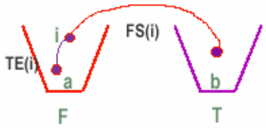
FM Gain Calculation: Direct Hyperedge Calc

- FM is able to calculate gain directly using hyperedges (➔ not necessary to convert hyperedges to edges)
- Definition:
  - Given a partition (A|B), we define the *terminal 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 (how fast can be computed?)
  - 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)=0,1 or B(n)=0,1

[©Keutzer]

FM Gain Calc: Direct Hyperedge Calc (cont.)

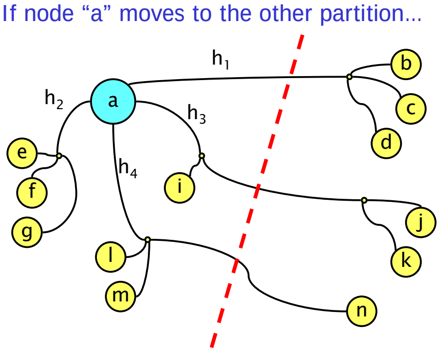
- Gain of cell depends only on its critical nets:
  - If a net is not critical, its cutstate cannot be affected by the move
  - A net which is not critical either before or after a move cannot influence the gains of its cells
- Let F be the "from" partition of cell i and T the "to":
- g(i) = FS(i) - TE(i), where:
  - FS(i) = # of nets which have cell i as their only F cell
  - TE(i) = # of nets connected to i and have an empty T side



[©Keutzer]

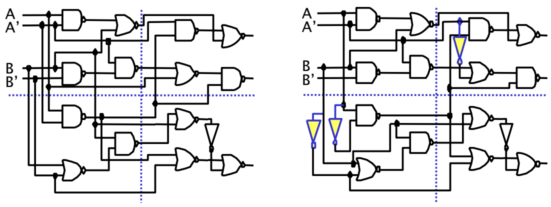
Hyperedge Gain Calculation Example

- If node "a" moves to the other partition…



Subgraph Replication to Reduce Cutsize

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

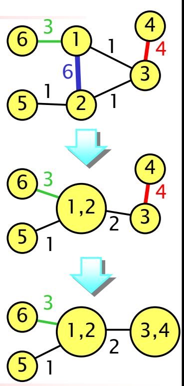


C. Kring and A. R. Newta, ICCAD, 1991.

[©Sherwani]

Clustering

- Clustering
  - Bottom-up process
  - Merge heavily connected components into clusters
  - Each cluster will be a new "node"
  - "Hide" internal connections (i.e., connecting nodes within a cluster)
  - "Merge" two edges incident to an external vertex, connecting it to two nodes in a cluster
- Can be a preprocessing step before partitioning
  - Each cluster treated as a single node



Other Partitioning Methods

- KL and FM have each held up very well
- Min-cut / max-flow algorithms
  - Ford-Fulkerson – for unconstrained partitions
- Ratio cut
- Genetic algorithm
- Simulated annealing

To Probe Further...

- B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning of Electrical Circuits", Bell System Technical Journal", pp291-307, 1970.
- C. M. Fiduccia and R. M. Mattheyses. "A linear-time heuristic for improving network partitions", Proceedings of the Design Automation Conference, pp 174-181, 1982.
- George Karypis, Rajat Aggarwal, Vipin Kumar and Shashi Shekhar, "Multilevel hypergraph partitioning: application in VLSI domain", Design Automation Conference, pp. 526-529, 1997.
- George Karypis and Vipin Kumar, "Multilevel k-way hypergraph partitioning", Design Automation Conference, pp. 343-348, 1999.
- A. E. Caldwell, A. B. Kahng and I. L. Markov, "Hypergraph Partitioning With Fixed Vertices", Design Automation Conference (DAC), pp. 355-359, 1999.
- A. E. Caldwell, A. B. Kahng, I. L. Markov, "Design and Implementation of Move-Based Heuristics for VLSI Hypergraph Partitioning", ACM Journal on Experimental Algorithms, Vol. 5, 2000.