Modern Physical Design:

Algorithm
Technology
Methodology
(Part II)

Andrew B. Kahng UCLA

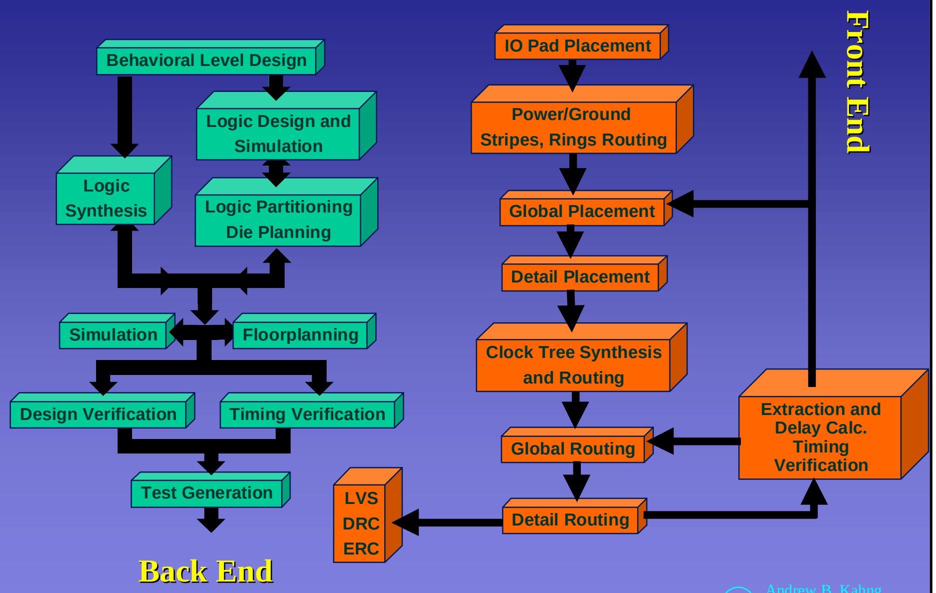
Majid Sarrafzadeh Northwestern

PART II: Fundamental Physical Design Formulation and Algorithms

- **Prediction**
  - Paradigms
  - Cost Functions
  - An example: FP
- **Placement**
  - Motivation
  - Formulation
  - Algorithms
  - Complexity management
  - Challenges
- **Routing**
  - Motivation
  - Formulation
  - Algorithms
  - Complexity management
  - Challenges
- **Placement / Synthesis**
  - Part III

Front End Flow

Back End Flow



Prediction and Cost Functions

Prediction

- **What is prediction ?**
  - every system has some critical cost functions: Area, wirelength, congestion, timing etc.
  - Prediction aims at estimating values of these cost functions without having to go through the time-consuming process of full construction.
- Allows quick space exploration, localizes the search
- For example:
  - statistical wire-load models
  - Wirelength in placement

Paradigms of Prediction

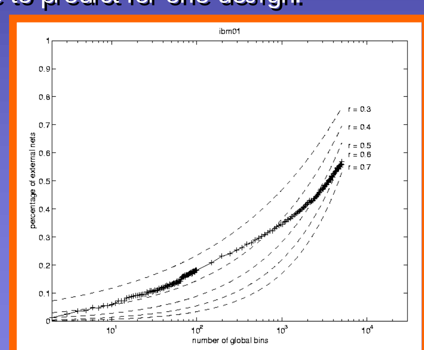
- **Two fundamental paradigms**
  - statistical prediction
    - #of two-terminal nets in all designs
    - #of two-terminal nets with length greater than 10 in all designs
  - constructive prediction
    - #of two-terminal nets with length greater than 10 in this design
  - … and everything in between, e.g.,
    - #of critical two-terminal nets in a design based on statistical data and a quick inspection of the design in hand.
- "Absolute truth" or "I need it to make progress"
- SLIP (System Level Interconnect Prediction) community.

Statistical Prediction

- W. E. Donath , A. B. Kahng, J. Mehndl , et al.
- Developing theoretical/statistical/observational models for interconnect estimation.
- Basic types
  - Estimation of Global Parameters. Assumes homogenous designs.
    - Rent's rule
    - Average multiplicity of a netlist is 2.2 - 2.6
  - Design Specific. Assumes localized homogeneity
    - localized Rent's rule
    - A specific Verilog block has average multiplicity 3.2

Statistical Prediction (cont)

- Shortcomings
  - circuits are not homogeneous.
  - they have been designed/defined hierarchically.
  - higher connectivity at lower levels.
  - these features are difficult to model statistically across designs and too expensive to predict for one design.
- Positive Aspects
  - very fast
  - reasonable approximation



Constructive Prediction

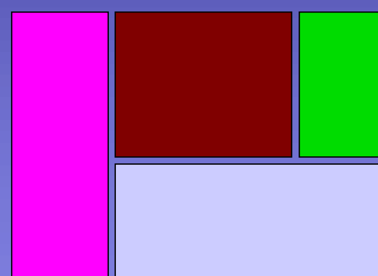
- Generally: the concept of fast algorithms
  - a quick/low-temperature annealing.
  - Final routabililty is correlated with first-level micut?
- SLIP position statement and some recent results (a few ideas will be discussed).
- E.g., Floorplan based on a given verilog hierarchy
- E.g., Construct fast layouts to predict final timing violations, routability information, etc.

Constructive Prediction (cont)

- Usefulness
  - xKy (K means knowledgeable) type of applications which may require some critical parameters from x to be fed back to y engine.
  - Allows quick exploration.
  - The predictor itself can act as the front-end for final Construction (time spent is not really wasted).
- Shortcomings
  - Slow
  - Can we trust it? (low-temperature annealing)
  - Would it localize the search too much?

A Case Study:
A Constructive floorplan predictor

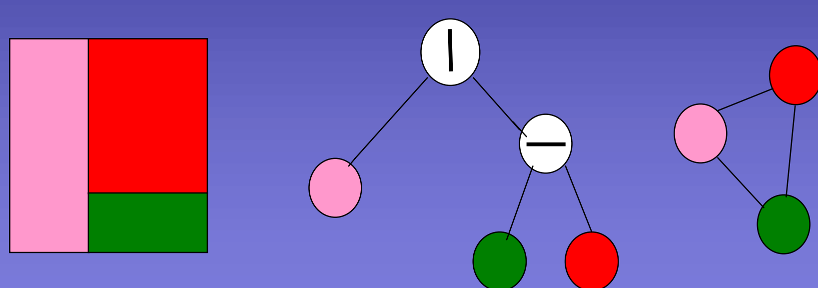
- To determine the approximate location of modules in a rectangular chip area so that a certain set of objectives (congestion/timing) is met (modules given by HDL, IP, clustering)



C Andrew B. Kahng
Majid Sarrafzadeh

Approaches to Floorplanning

- Rectangular dual graph.
- Simulated Annealing.
- Math programming
- Hierarchical floorplanning (top-down / bottom-up).



C Andrew B. Kahng
Majid Sarrafzadeh

Simulated Annealing

- A very-good floorplanner (for shape management).
- But has major limitation in terms of running time.

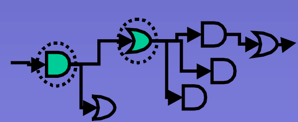
test ckt.	modules, edges	run time
prim2.s3	98, 1410	20 hrs
prim1	750, 830	31 hrs
prim2	2907, 2961	> 80 hrs

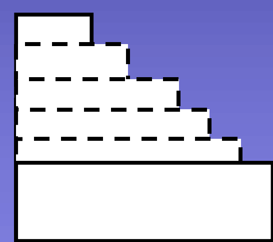
Why prediction of floorplan metrics?

- By defining more modules (more than 20-30) better estimates of the system can be obtained.
- With the reduction in device sizes, interconnect is becoming important.
- Floorplanning forms the core of many design flows.
- System-on-a-chip type applications, assessing viability of a chip at the architectural level.

A constructive floorplan prediction

- Do hierarchical floorplanning (fast).
- Exploit flexibility in the shapes of the modules.
- Flexibility means eased shape management, can concentrate on other objectives (wirelength).
  - A measure of how flexible a module is in terms of shape and number of representations.





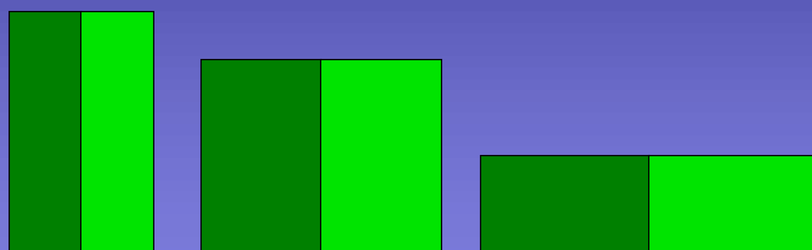
Modeling Flexibility

w1 < ....< wN
h1 <............< hN

Flexibility = sum(wi+1/wi + hi/hi+1) * N

Observations on flexible modules

- Same area, same flexibility (unique) modules combine to produce zero dead-space. (large flexibility modules combine to produce large flexibility module)



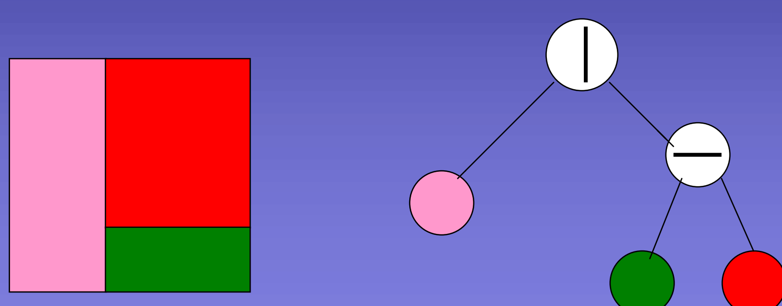
17

ICCAD Tutorial: November 11, 1999

© Andrew B. Kahng
Majid Sarrafzadeh

Hierarchical floorplanning

- Based on a divide-and-conquer paradigm, at each level of the hierarchy only a small number of modules are considered.



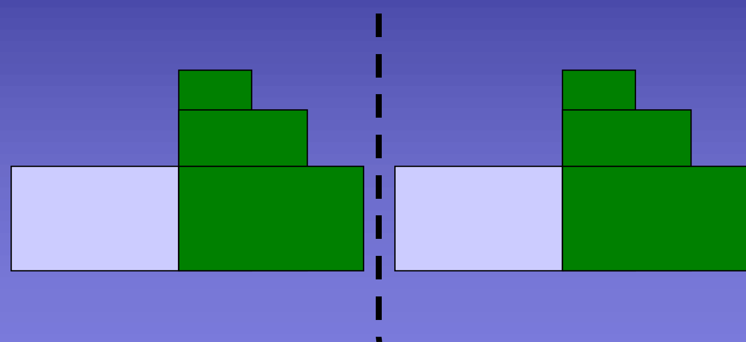
18

ICCAD Tutorial: November 11, 1999

© Andrew B. Kahng
Majid Sarrafzadeh

Observations on flexible modules (contd.)

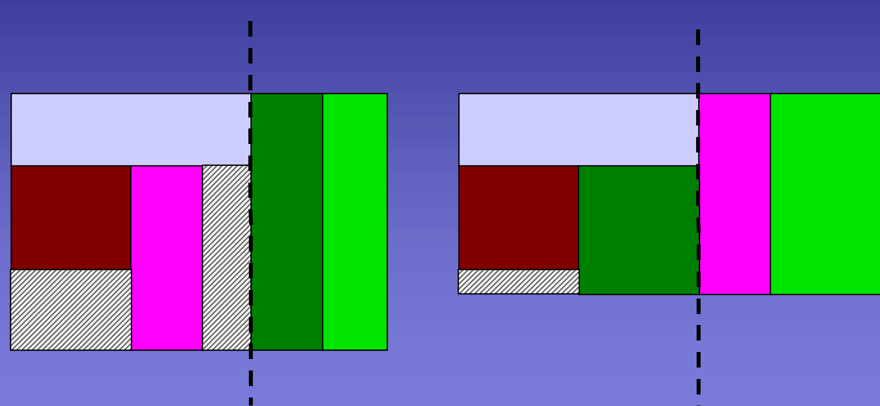
- Combining a rigid module with a flexible one is better (in general) than combining two rigid modules.



© Andrew B. Kahng
Majid Sarrafzadeh

Prediction heuristics

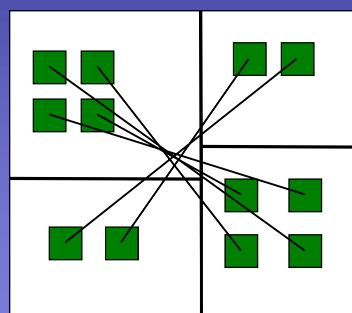
- Balance with respect to area and flexibility.

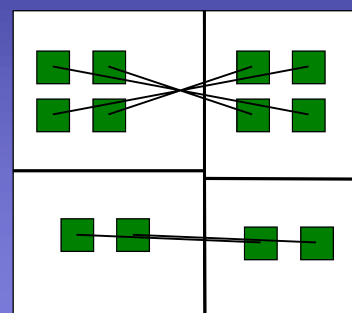


© Andrew B. Kahng
Majid Sarrafzadeh

Prediction heuristics (contd.)

- Wirelength improvement.
  - bi-partitioning may push some of the connected modules farther apart.

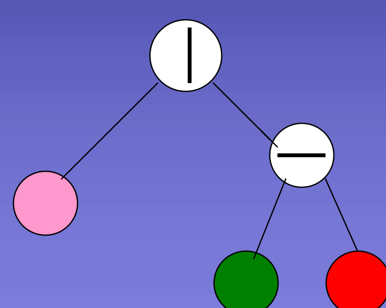




C Andrew B. Kahng
Majid Sarrafzadeh

Extension to Sizing Theorem

- Sizing theorem asks to keep non-redundant width, height implementations for area. What happens to the wirelength ?
- Claim: non-redundant area implementations also contain non-redundant wirelength implementations.



C Andrew B. Kahng
Majid Sarrafzadeh

Prediction algorithm

- Similar to top-down hierarchical floorplanning but with added heuristics.
  - partition the circuit recursively (we use hMetis), balance on area and flexibility.
  - aspect-ratio heuristic decides the cut of internal nodes.
  - Wirelength improvement applied as post-processing.

Test results

ckt(%rigid)	Wong-Liu		Predictor			
	cost	time(sec)	cost	% diff	time(s)	speedup
ind1(10)	12039	177	13256	10.11	0.3	590
ind1(20)	12168	173	12871	5.78	0.3	577
ind1(30)	14971	122	18535	23.8	0.3	407
ind1(50)	16033	88	19935	24.34	0.3	293
hway(10)	70571	3420	71981	2.00	1.7	2011
hway(20)	71838	3515	73390	2.16	1.7	2068
hway(30)	72274	3524	75530	4.5	1.6	2202
hway(50)	77653	2564	80273	3.37	1.6	1602
fract(10)	131431	15651	129187	-1.7	4.5	3478
fract(20)	137044	12803	139125	1.52	4.6	2783
fract(30)	137084	14694	140192	2.27	4.6	3194
fract(50)	144072	9268	152436	5.81	4.7	1972
prim1(10)	831329	110491	863012	3.81	40.0	2762
prim1(20)	867690	100010	864245	-0.4	39.4	2538
prim1(30)	870456	95299	881813	1.3	37.1	2569
prim1(50)	897120	68303	957847	6.77	36.6	1866
prim2_s1(10)	230703	11899	216478	-6.2	16.7	713
prim2_s1(20)	235694	11141	227186	-3.6	16.9	659
prim2_s1(30)	248317	9306	230093	-7.3	16.6	561
prim2_s1(50)	249489	7445	254568	2.03	16.5	451
prim2_s2(10)	323017	38416	299704	-7.21	29.8	1289
prim2_s2(20)	319897	30062	318005	-0.59	30.4	989
prim2_s2(30)	333313	29632	322356	-3.28	29.7	998
prim2_s2(50)	354387	21045	344175	-2.88	29.5	713

Extension of Predictor to Constructor

- Why is annealing good (and slow)?
  - Focused shape management

- What is important in floorplanning with eased shape management ?

  \- It is not important to properly fit all the modules from the very beginning like Annealing.

  \- Important to avoid long wires, bad for congestion/timing etc.

  \- However in a small subset of floorplanning area-management is important.

  \- Use partitioning early on but switch to annealing at later stage.

C Andrew B. Kahng
Majid Sarrafzadeh

Results

ckt(%rigid)	Wong-Liu		Constructor			
	cost	time(sec)	cost	% diff	time(s)	speedup
ind1(10)	12039	177	12182	1.18	8.6	21
ind1(20)	12168	173	12324	1.28	7.5	23
ind1(30)	14971	122	15400	2.86	5.9	21
ind1(50)	16033	88	17578	9.64	5.1	17
hway(10)	70571	3420	69611	-1.36	194	18
hway(20)	71838	3515	70448	-1.93	183	19
hway(30)	72274	3524	71900	-0.51	210	17
hway(50)	77653	2564	78696	1.34	105	24
fract(10)	131431	15651	128388	-2.32	897	18
fract(20)	137044	12803	130984	-4.42	704	18
fract(30)	137084	14694	135869	-0.88	723	20
fract(50)	144072	9268	145392	0.91	549	17
prim1(10)	831329	110491	832365	1.2	4629	24
prim1(20)	867690	100010	862657	-0.6	4911	20
prim1(30)	870456	95299	871623	0.13	4214	23
prim1(50)	897120	68303	931694	3.85	3617	19
prim2_s1(10)	230703	11899	215193	-6.72	455	26
prim2_s1(20)	235694	11141	217542	-7.70	439	25
prim2_s1(30)	248317	9306	229349	-7.46	411	23
prim2_s1(50)	249489	7445	248566	-0.37	328	23
prim2_s2(10)	323017	38416	283188	-12.33	1477	26
prim2_s2(20)	319897	30062	301768	-5.67	1389	22
prim2_s2(30)	333313	29632	308023	-7.59	1506	20
prim2_s2(50)	354387	21045	323188	-8.80	1138	18

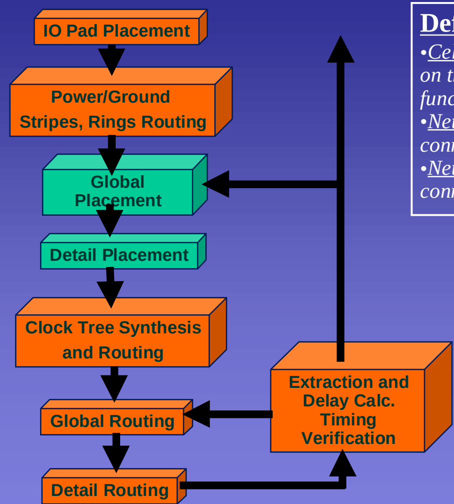
C Andrew B. Kahng
Majid Sarrafzadeh

Conclusion

- Statistical and Constructive predictors are extremely valuable
- We need a lot more research to really understand them
- SLIP is a focused group looking at these issues

Placement Paradigms

VLSI Design Flow and Physical Design Stage



Definitions:

•*Cell*: a circuit component to be placed on the chip area. In placement, the functionality of the component is ignored.
•*Net*: specifying a subset of terminals, to connect several cells.
•*Netlist*: a set of nets which contains the connectivity information of the circuit.

Placement Problem

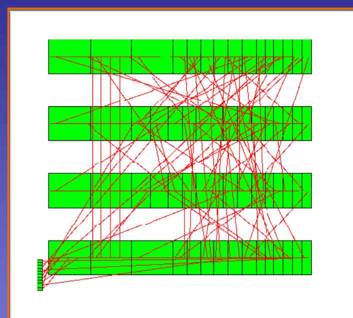
Input:
•A set of cells and their complete information (a cell library).
•Connectivity information between cells (netlist information).

Output:
A set of locations on the chip: one location for each cell.

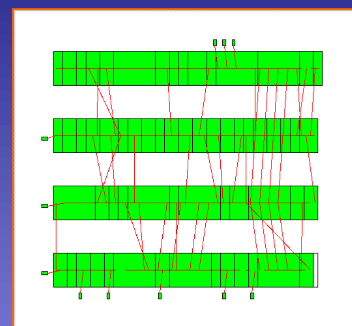
Goal:
The cells are placed to produce a routable chip that meets timing (low-power, …)

Challenge:
•The number of cells in a design is very large (> 1 million).
•The timing constraints are very tight.

Placement Problem



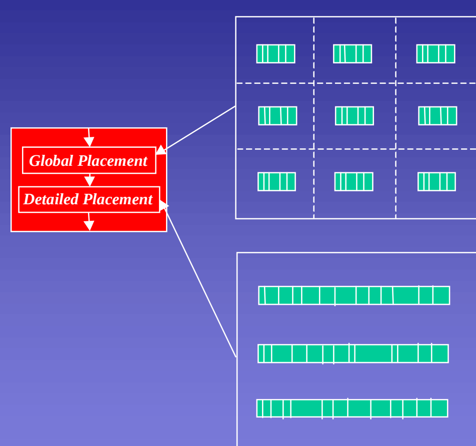
A bad placement



A good placement

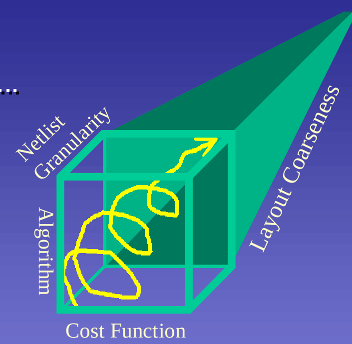
Global and Detailed Placement

In global placement, we decide the approximate locations for cells by placing cells in global bins. In detailed placement, we make some local adjustment to get the final non-overlapping placement.



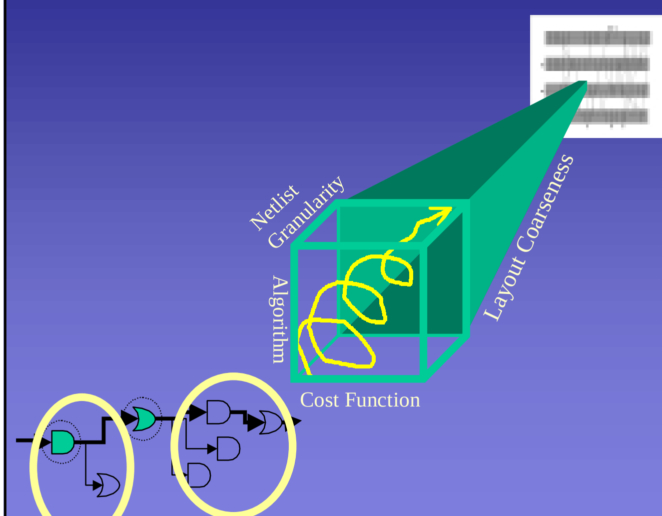
Placement Cube (4d)

- Cost Function(s) to be used
  - Cut, wirelength, congestion, crossing, ...
- Algorithm(s) to be used
  - FM, Quadratic, annealing, ….
- Granularity of the netlist
- Coarseness of the layout domain
  - 2x2, 4x4, ….



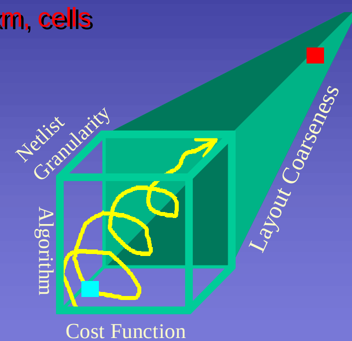
- An effective methodology picks the right mix from the above and knows when to switch from one to next.
- Today: Ad-hoc

What does the cube represent?



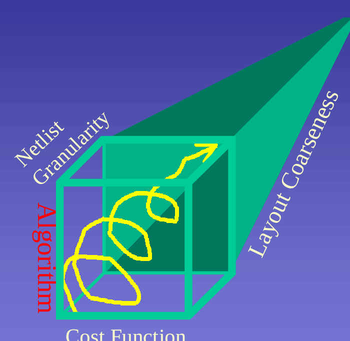
Twolf Global Placement

- Two points in the entire cube
  - Annealing, WL, mxm, clusters
  - Annealing, WL, mxm, cells



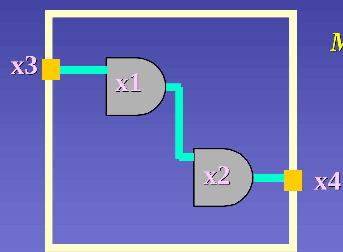
Traditional Algorithms

- Quadratic Placement
- Simulated Annealing
- Bi-Partitioning / Quadrisection
- Force Directed Placement
- Hybrid



Quadratic Placement

☞ Analytical Technique



$$\mathit{Min}\ [(x1-x3)^2 + (x1-x2)^2 + (x2-x4)^2] : F$$

$\delta F/\delta x1 = 0;$

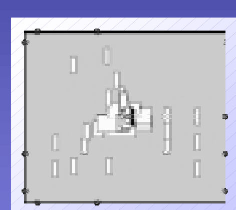
$\delta F/\delta x2 = 0;$

$\Rightarrow$ $Ax = B$

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad B = \begin{bmatrix} x3 \\ x4 \end{bmatrix} \quad x = \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

Analytical Placement

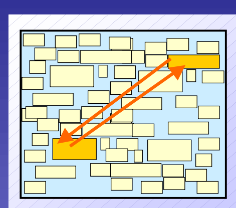
- Get a solution with lots of overlap
- What do we do with the overlap?



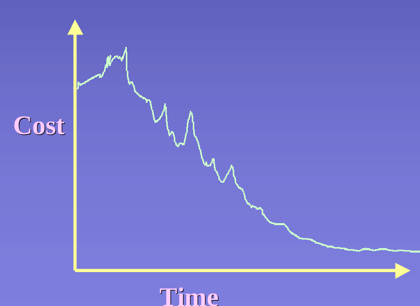
Pros and Cons of QP

- Pros:
  - Very Fast Analytical Solution
  - Can Handle Large Design Sizes
  - Can be Used as an Initial Seed Placement Engine
- Cons:
  - Not Suitable for Timing Driven Placement
  - Not Suitable for Simultaneous Optimization of Other Aspects of Physical Design (clocks, crosstalk…)
  - Gives Trivial Solutions without Pads (and close to trivial with pads)

Simulated Annealing Placement



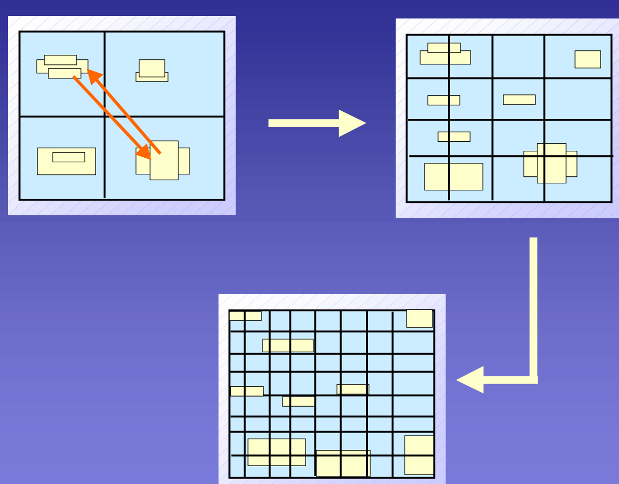
- **Initial Placement Improved through Swaps and Moves**
- **Accept a Swap/Move if it improves cost**
- **Accept a Swap/Move that degrades cost under some probability conditions**



Pros and Cons of SA

- Pros:
  - Can Reach Globally Optimal Solution (given “enough” time)
  - Open Cost Function.
  - Can Optimize Simultaneously all Aspects of Physical Design
  - Can be Used for End Case Placement
- Cons:
  - Extremely Slow Process of Reaching a Good Solution

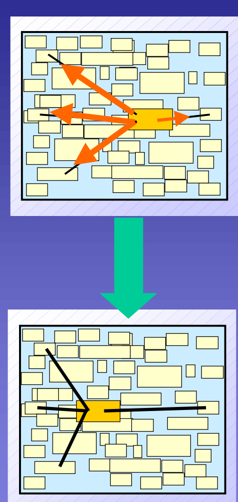
Bi-Partitioning/Quadrisection



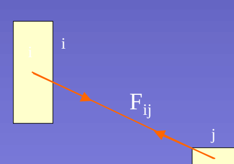
Pros and Cons of Partitioning Based Placement

- Pros:
  - More Suitable to Timing Driven Placement since it is Move Based
  - New Innovation (hMetis) in Partitioning Algorithms have made this Extremely Fast
  - Open Cost Function
  - Move Based means Simultaneous Optimization of all Design Aspects Possible
- Cons:
  - Not Well Understood
  - Lots of "indifferent" moves
  - May not work well with some cost functions.

Force Directed Placement



- **Cells are dragged by forces.**
- **Forces are generated by nets connecting cells. Longer nets generate bigger forces.**
- **Placement is obtained by either a constructive or an iterative method.**



Pros and Cons of Force Directed Placement

- Pros:
  - Very Fast Analytical Solution
  - Can Handle Large Design Sizes
  - Can be Used as an Initial Seed Placement Engine
  - The force
- Cons:
  - Not sensitive to the non-overlapping constraint
  - Gives Trivial Solutions without Pads
  - Not Suitable for Timing Driven Placement

Hybrid Placement

- Mix-matching different placement algorithms
- Effective algorithms are always hybrid

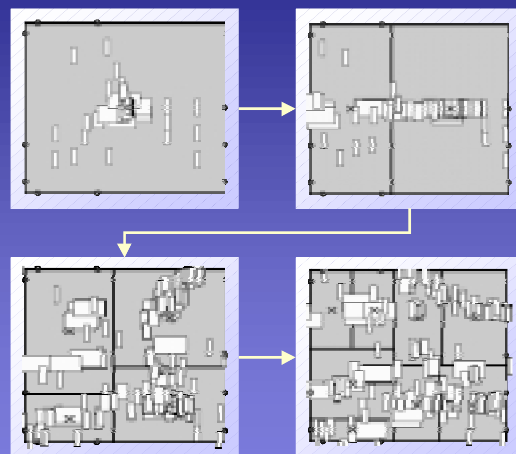
GORDIAN (quadratic + partitioning)

Initial Placement

$$min\{\sum(x_i - x_j)^2\}$$

$$min\{\sum(y_i - y_j)^2\}$$

Partition and Replace



Hybrid Placement

- It is not yet understood how to combine various algorithms to construct an effective flow: when to use which algorithm and for how long (and why)?
- Current methods are ad-hoc
- In fact, the flow should be instance based
  - in one circuit annealing is good enough
  - in another quadratic + partitioning

Paritioning in Placement Algorithms

- Used in many placement algorithms
- In its most basic form or cost function / algorithmic variations of it
- Here, we used partitioning in discussion of algorithms and will used it later in discussion of cost functions
- While describing partitioning, we will touch on a few fundamental concepts such as clustering

Standard Min-Cut Formulation

- Given a vertex- and hyperedge-weighted hypergraph $H = (V, E)$, partition V into disjoint partitions $C_1, \ldots, C_k$, $\bigcup_{i=1}^{k} C_i = V$ such that the number of cut hyperedges is minimized.
- Edge $e \in E$ is *cut* if there exist C_i, C_j with $e \cap C_i, e \cap C_j \neq \emptyset$
- $k = 2$ most often studied
- Partition sizes must satisfy balance constraints
- Focus on cut: communication, interaction between subproblems

Variant Formulations

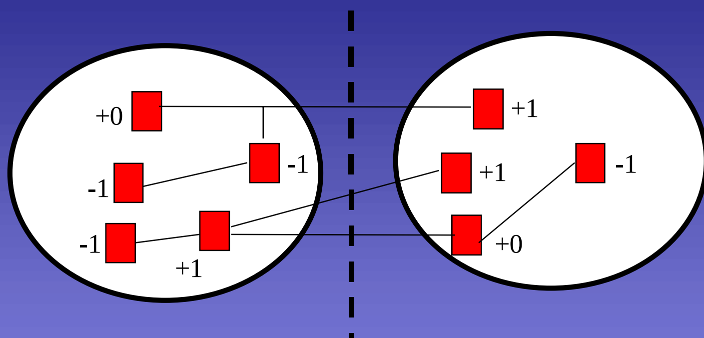
- Constraints
  - I/O, area
  - path delay
  - multi-balance (area, power)
  - multi-dimensional balance
  - hierarchy
- Degrees of freedom
  - replication
- Objectives
  - min-cut ?

Iterative Algorithms

- Neighborhood operator
  - single-shift
  - pair-swap
- Accept/reject ?
  - pass
    - start with all vertices unlocked
    - do
    - make the best move of unlocked vertices
    - lock moved vertices
    - until all vertices locked
    - find best prefix of this move sequence
    - actually make this "compound move"

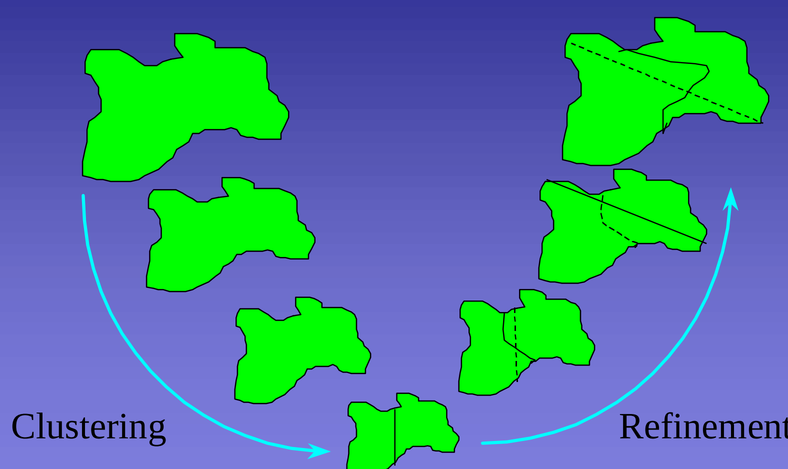
26

FM



Advantages: Simple, efficient and fast
Disadvantage: Poor quality for large instances

Multilevel Partitioning



Iterative Algorithms

- Folklore   greedy shift, swap
- 1970 Kernighan-Lin
- 1982 Fiduccia-Mattheyses
- 1983 Metaheuristics… - GA, SA, LSMC
- 1984 Goldberg-Burstein, two-level
- 1989 Krishnamurthy, Sanchis,
- 1993 Quick Cut
- 1995 Metis, Chaco, … Multilevel
- 1996 PROP, CLIP, CDIP
- 1997 MLc, HMetis
- 1998 Deep Prop, ISPD98 suite

Iterative Algorithms

Practicality/creativity:

- relaxed balance constraints
- multiple unlocks
- early termination
- dual representation

Spectral / Geometric

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x_i - x_j\right)^2 = x^{T} Q x$$

- Hall70

$$Qx = \lambda x \quad \left(\lambda = WL \ \text{ if } \ x^{T} x = 1\right)$$

  - closest legal partitioning to k smallest eigenvectors
- Barnes85, HagenK91, ChanSZ93
- Blanks85, FrankleK86, ArunR91, AlpertY95

Ordering-based: 1-D place (order) then partition
Embedding-based: legalization of analytic placement

F. Johannes (TU Munich)
B. Korte (U Bonn)

Combinatorial Algorithms

- Min-delay clustering (replication) LawlerLT69
- Network flows
  - cut
  - replication
- Mathematical programming
  - replication/retiming
  - constrained partitioning (MCM's)

C. K. Cheng (UCSD)
D. F. Wong (UT Austin)

A Synthesis of Placer Evolution

- Top-down (partitioning) technology

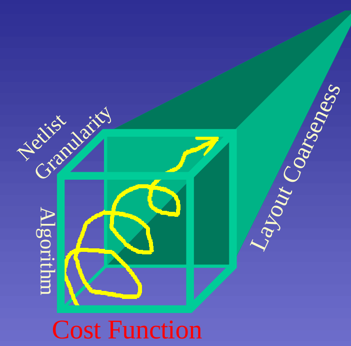
top-down bisection = placement onto 2 points!
terminal propagation DunlopK85
quadrisection SuarisK87
quadrisection with exact placement objective HuangK97

Cost Functions for Placement

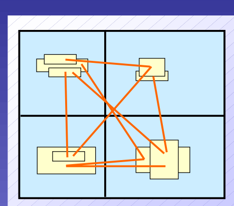
- The final goal of placement is to achieve routability and meet timing constraints
- Constraints are very hard to use in optimization, thus we use cost functions (e.g., Wirelength) to predict our goals.
  - We will show what happens when you try constraints directly
  - The main challenge is a technical understanding of various cost functions and their interaction.

Cost Functions for Placement

- Net-cut
- Linear wirelength
- Quadratic wirelength
- Congestion
- Timing
- Coupling
- Other performance related cost functions
- Undiscovered: crossing

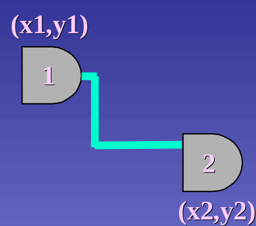


Net-cut Cost for Global Placement



- **The net-cut cost is defined as the number of external nets between different global bins**
- **Minimizing net-cut in global placement tends to put highly connected cells close to each other.**

Linear Wirelength Cost

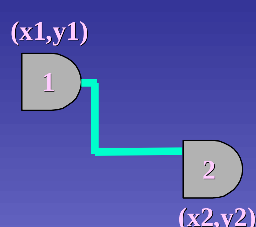


The linear length of a net between cell 1 and cell 2 is

$$l_{12} = |x1-x2| + |y1-y2|$$

The linear wirelength cost is the summation of the linear length of all nets.

Quadratic Wirelength Cost

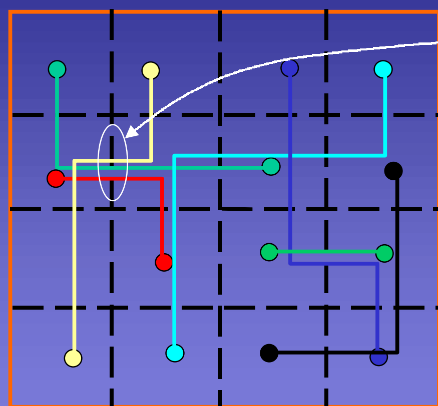


The quadratic length of a net between cell 1 and cell 2 is

$$l_{12} = (x1-x2)^2 + (y1-y2)^2$$

The quadratic wirelength cost is the summation of the quadratic length of all nets.

Congestion Cost



Routing demand = 3
Assume routing supply is 1,
overflow = 3 - 1 = 2 on this edge.

Overflow on each edge =

- Routing Demand - Routing Supply (if Routing Demand > Routing Supply)
- 0 (otherwise)

$$\text{Congestion Overflow} = \sum_{\text{all edges}} \text{overflow}$$

Cost Functions for Placement

- Various cost functions (and a mix of them) have been used in practice to model/estimate routability and timing
- We have a good "feel" for what each cost function is capable of doing
- We need to understand the interaction among cost functions

C Andrew B. Kahng
Majid Sarrafzadeh

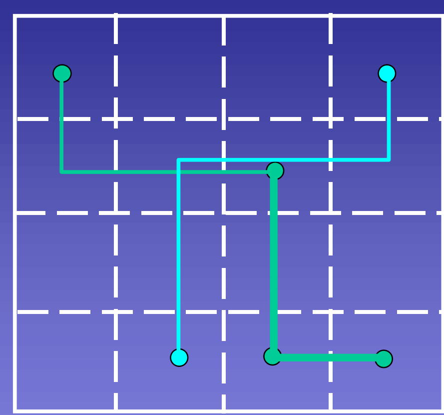
Cut vs. Wirelength

- Globally: In a 2x1 bin, wirelength is the same as cut

Congestion Minimization and Congestion vs Wirelength

- Congestion is important because it closely represents routability (especially at lower-levels of granularity)
- Congestion is not well understood
- Ad-hoc techniques have been kind-of working since congestion has never been severe
- It has been observed that length minimization tends to reduce congestion.
- Goal: Reduce congestion in placement (willing to sacrifice wirelength a little bit).

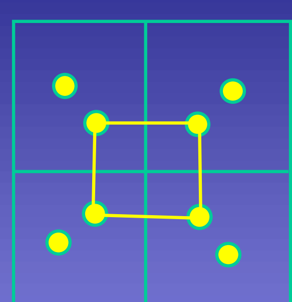
Correlation between Wirelength and Congestion



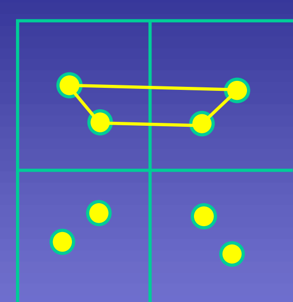
Total Wirelength = Total Routing Demand

ICCAD Tutorial: November 11, 1999
(C) Andrew B. Kahng
Majid Sarrafzadeh

Wirelength ≠ Congestion



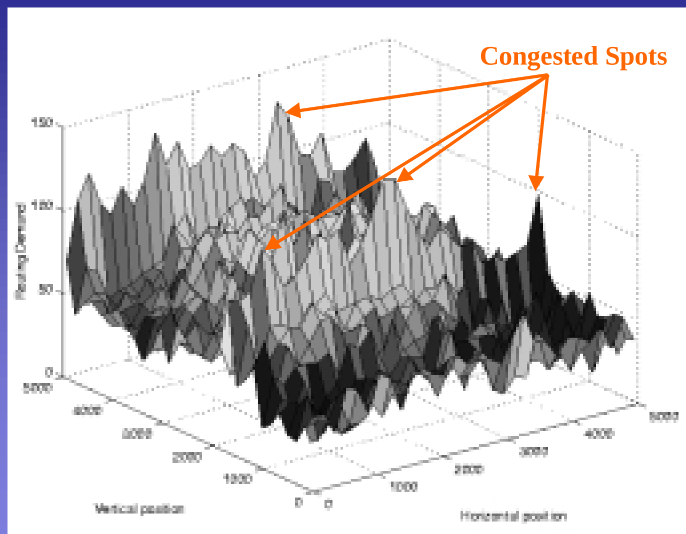
A congestion minimized placement



A wirelength minimized placement

ICCAD Tutorial: November 11, 1999
(C) Andrew B. Kahng
Majid Sarrafzadeh

Congestion Map of a Wirelength Minimized Placement

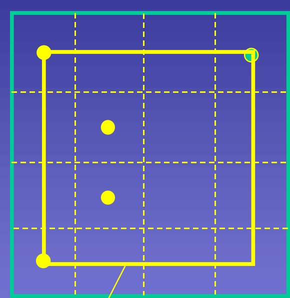


Different Routing Models for modeling congestion

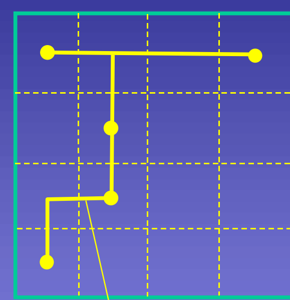
- Bounding box router: fast but inaccurate.
- Real router: accurate but slow.
- A bounding box router can be used in placement if it produces correlated routing results with the real router.

- Note: For different cost functions, answer might be different (e.g., for coupling, only a detailed router can answer).

Different Routing Models



A bounding box routing model



A MST+shortest_path routing model

Correlation Test Between Different Routers

Circuit	RoutModel	A	B	C	D	E	F
Primary1	BBox	14	36	26	27	40	30
	Real	27	9	7	4	5	4
Primary2	BBox	562	163	594	680	147	631
	Real	331	63	378	407	73	378
struct	BBox	949	459	1086	1091	665	1119
	Real	92	294	121	142	414	154
biomed	BBox	4098	2522	7458	7335	3790	6711
	Real	188	48	706	760	180	474

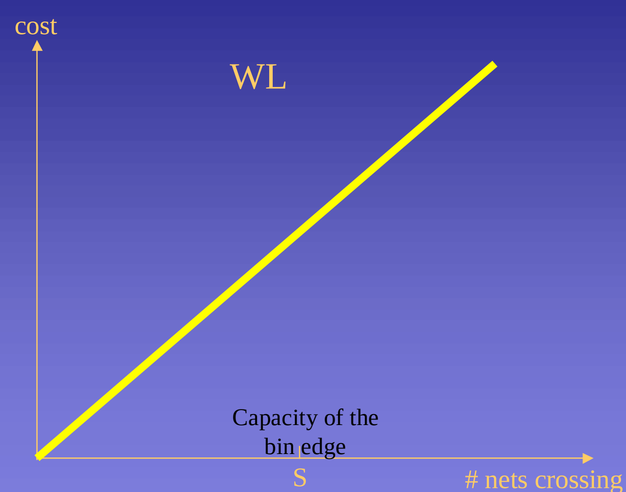
Evaluate overflow value using different routers.
(A, B, C, D, E and F are six independent placements)

Conclusion:
Bounding box router cannot be used in placement to evaluate congestion.

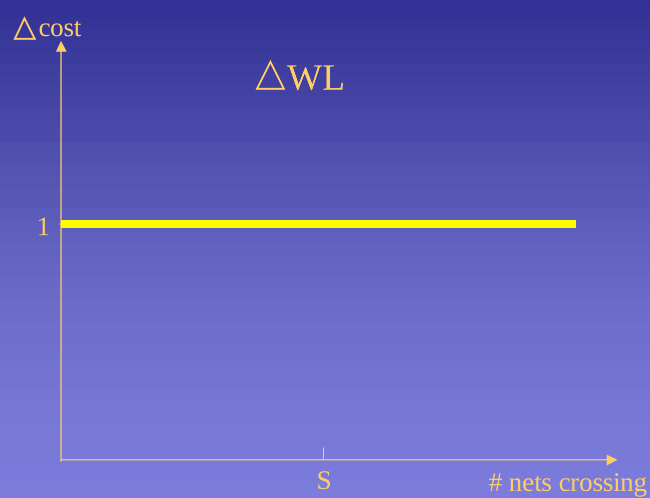
Objective Functions Used in Congestion Minimization

- WL: Standard total wirelength objective.
- Ovrflw: Total overflow in a placement (a direct congestion cost).
- Hybrid: $(1- \alpha)$WL + α Ovrflw
- QL: A quadratic plus linear objective.
- LQ: A linear plus quadratic objective.
- LkAhd: A modified overflow cost.
- $(1- \alpha_T)$WL + α_T Ovrflw: A time changing hybrid objective which let the cost function gradually change from wirelength to overflow as optimization proceeds.

Wirelength Cost

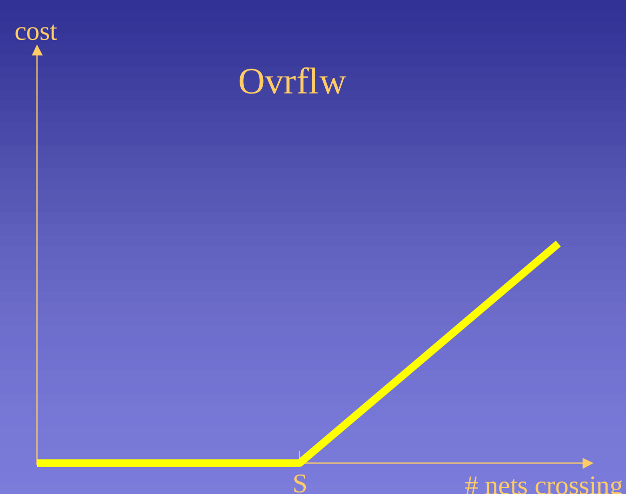


Delta Wirelength Cost



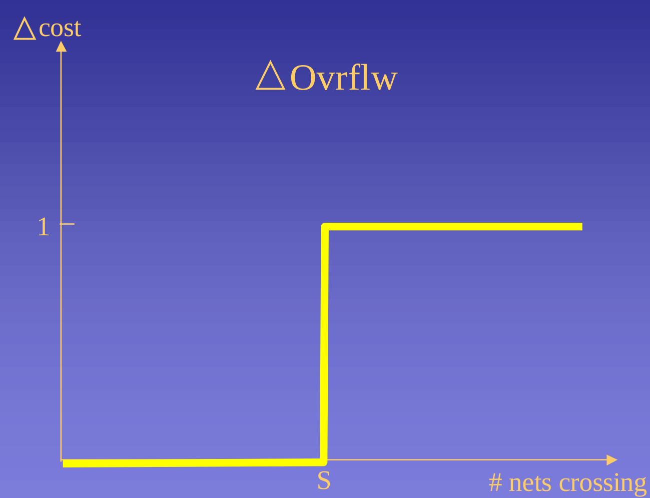
ICCAD Tutorial: November 11, 1999

Overflow Cost

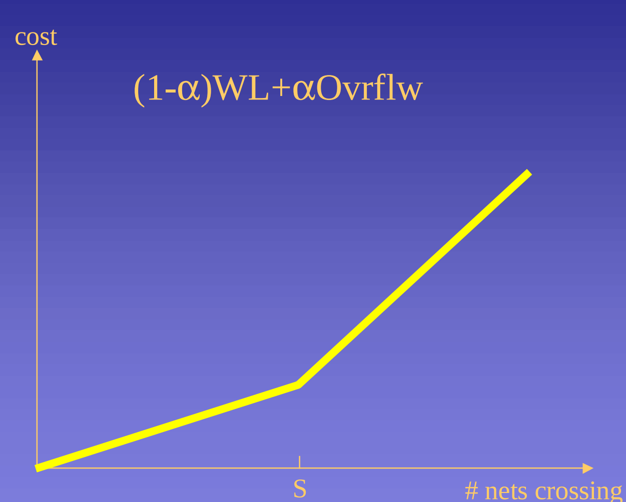


ICCAD Tutorial: November 11, 1999

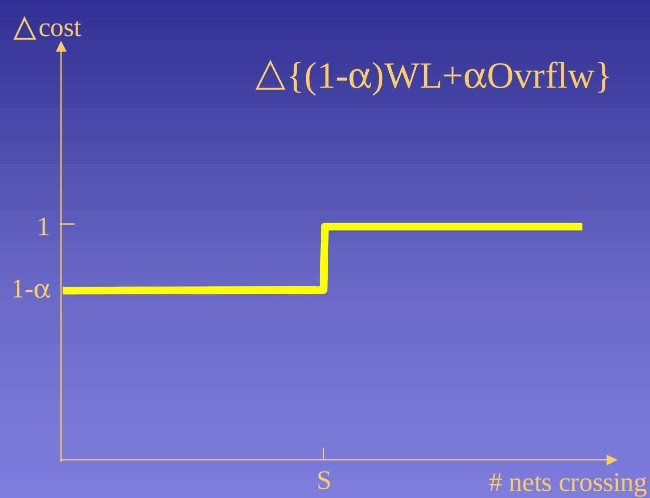
Delta Overflow Cost



Hybrid Cost



Delta Hybrid Cost



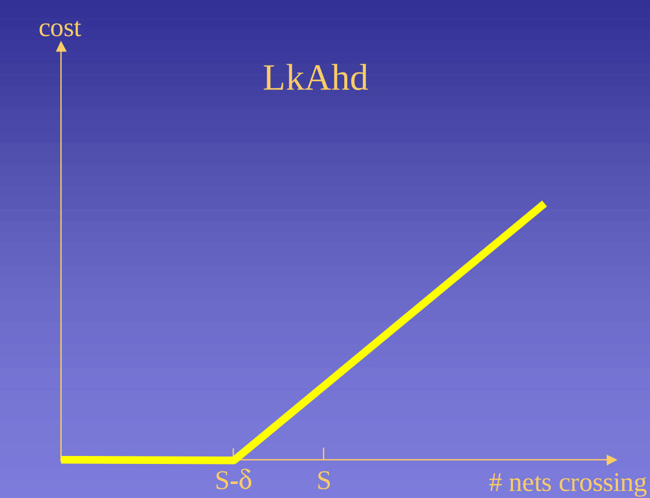
Look-ahead overflow cost

Overflow on each edge = { Routing Demand - Routing Supply (if Routing Demand > Routing Supply); 0 (otherwise) }

Total Overflow = $\sum_{\text{all edges}}$ overflow

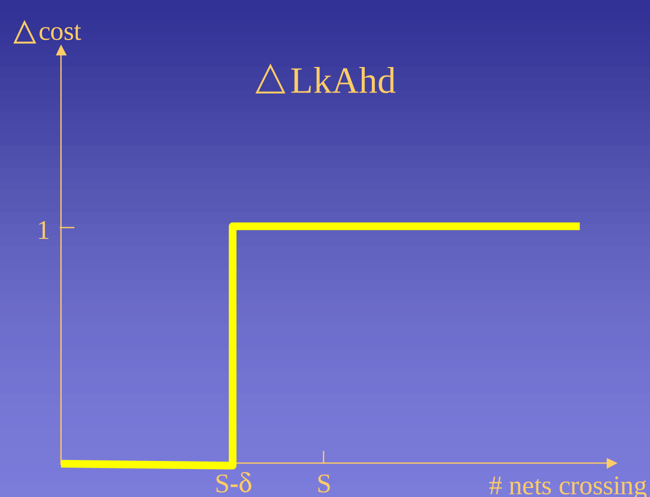
Overflow with look-ahead = { Routing Demand - (Routing Supply - δ) (if Routing Demand > Routing Supply - δ); 0 (otherwise) }

Look-ahead Cost



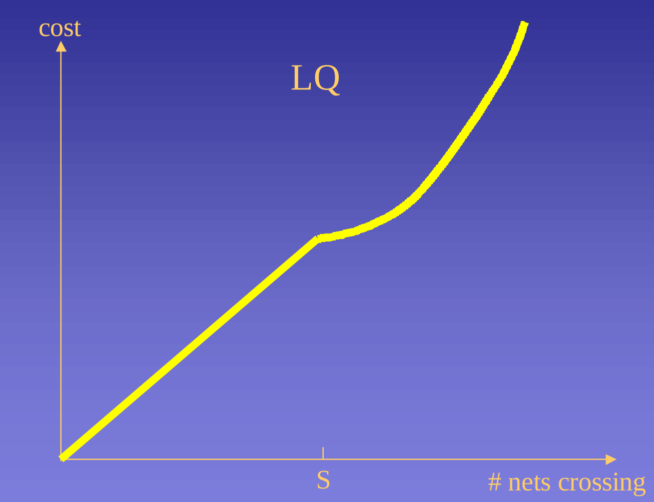
C Andrew B. Kahng
Majid Sarrafzadeh

Delta Look-ahead Cost

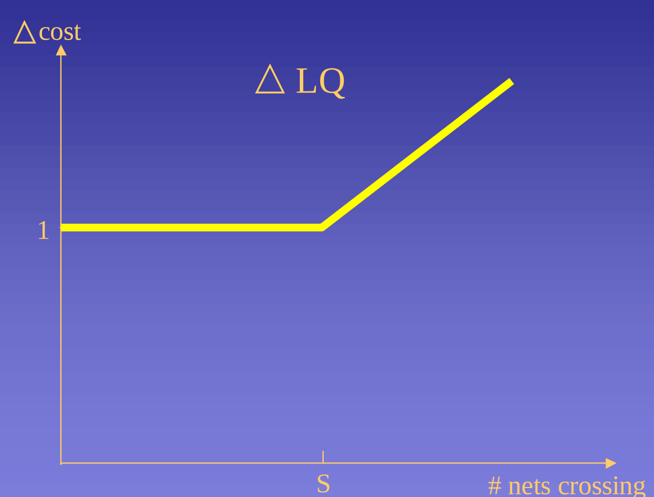


C Andrew B. Kahng
Majid Sarrafzadeh

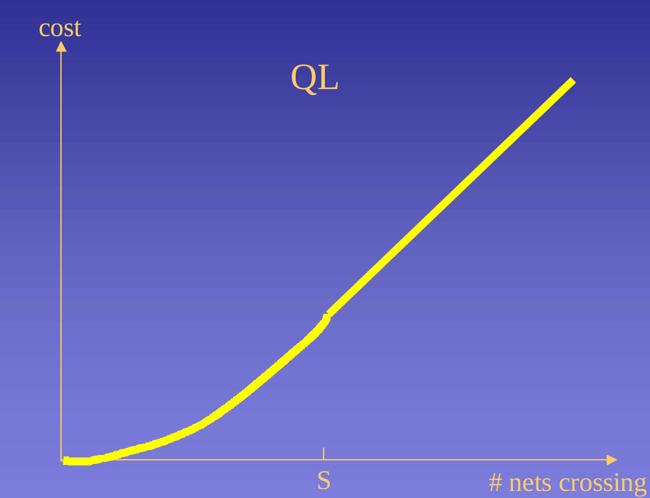
Linear-Quadratic Cost



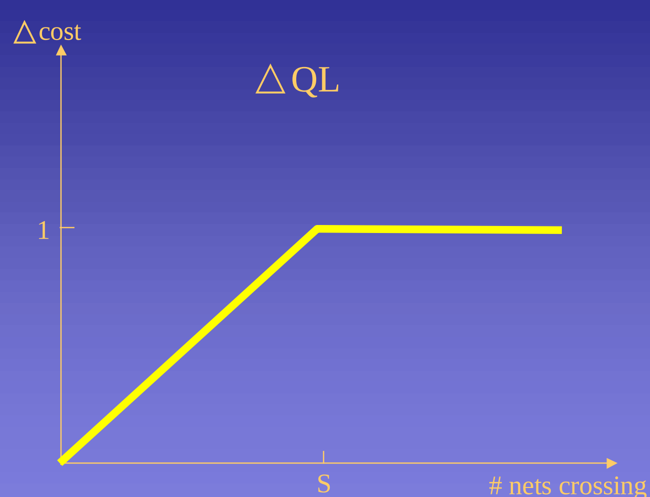
Delta Linear-Quadratic Cost



Quadratic-Linear Cost



Delta Quadratic-Linear Cost



ICCAD Tutorial: November 11, 1999
C Andrew B. Kahng
Majid Sarrafzadeh

Comparison Between Different Objectives

	wirelength	overflow	runtime
WL	27885	3011	643
Ovrflw	57992	20400	116050
$0.8WL + 0.2Ovrflw$	53289	20982	51001
$0.6WL + 0.4Ovrflw$	56993	23399	53398
$0.5WL + 0.5Ovrflw$	58016	23768	50074
$0.4WL + 0.6Ovrflw$	59434	24954	49283
$0.2WL + 0.8Ovrflw$	62450	27063	49884
$(1 - \alpha_T)WL + \alpha_T Ovrflw$	65233	29486	47300
LkAhd	70346	32367	43523
QL	65532	27738	47426
LQ	67786	30846	48212

Comparison between different objectives for circuit biomed.

Comparison Between Different Objectives

	wirelength	overflow	runtime
WL	26918	151	269
Ovrflw	80425	6391	9116
$0.8WL + 0.2Ovrflw$	79918	9406	17103
$0.6WL + 0.4Ovrflw$	81704	9149	17108
$0.5WL + 0.5Ovrflw$	84586	9660	17145
$0.4WL + 0.6Ovrflw$	89734	10883	17167
$0.2WL + 0.8Ovrflw$	96108	12052	17517
$(1 - \alpha_T)WL + \alpha_T Ovrflw$	100869	13055	17761
LkAhd	77823	5613	9267
QL	66086	4231	11600
LQ	75090	6298	10284

Comparison between different objectives for circuit Primary2.

Comparison Between Different Objectives

	wirelength	overflow	runtime
WL	9110	159	5839
Ovrflw	718451	130410	93381
$0.8WL + 0.2Ovrflw$	651406	117992	89283
$0.6WL + 0.4Ovrflw$	655704	118569	93330
$0.5WL + 0.5Ovrflw$	658943	118994	89081
$0.4WL + 0.6Ovrflw$	660134	119084	90385
$0.2WL + 0.8Ovrflw$	661199	119243	90469
$(1-\alpha_T)WL + \alpha_T Ovrflw$	698035	126173	60884
LkAhd	711535	128970	61417
QL	669985	120612	59896
LQ	718701	130538	61840

Comparison between different objectives for circuit avqs.

Comparison Between Different Objectives

	wirelength	overflow	runtime
WL	107261	802	7934
Ovrflw	879751	160520	113085
$0.8WL + 0.2Ovrflw$	832858	153260	110778
$0.6WL + 0.4Ovrflw$	838492	159306	119350
$0.5WL + 0.5Ovrflw$	839052	159465	113754
$0.4WL + 0.6Ovrflw$	842840	153849	117805
$0.2WL + 0.8Ovrflw$	849358	159374	110485
$(1-\alpha_T)WL + \alpha_T Ovrflw$	859994	156729	72723
LkAhd	881915	161172	71997
QL	840739	152345	72526
LQ	879860	160625	72593

Comparison between different objectives for circuit avql.

Post Processing to Reduce Congestion

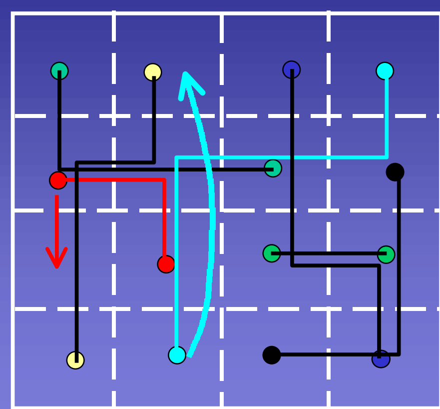
Reduce congestion globally by minimizing the traditional wirelength

Post process the wirelength optimized placement using the congestion objective

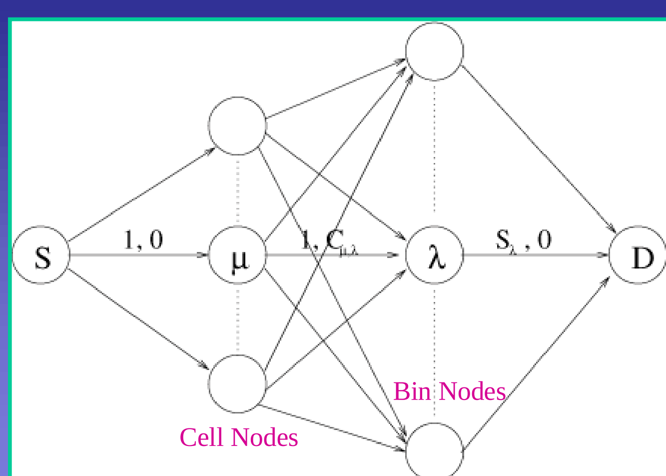
Post Processing Heuristics

- Greedy cell-centric algorithm: Greedily move cells around and greedily accept moves.
- Flow-based cell-centric algorithm: Use a flow-based approach to move cells.
- Net-centric algorithm: Move nets with bigger contributions to the congestion first.

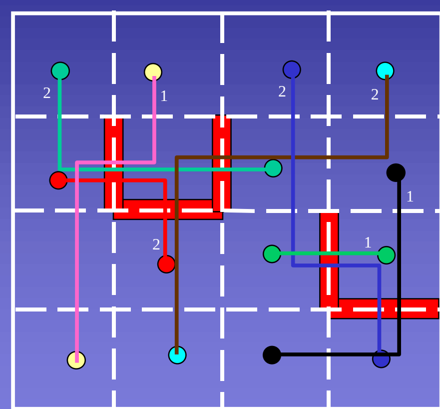
Greedy Cell-centric Heuristic



Flow-based Cell-centric Heuristic



Net-centric Heuristic

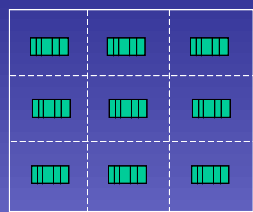


Post Processing Results

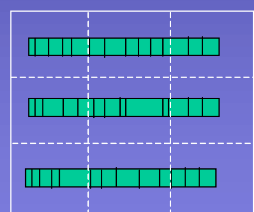
TestCase	before PP	cell-centric	flow-based	net-centric	% imp. net-centric vs. before PP
highway2	12	7	7	7	41.7%
fract	16	14	14	14	12.5%
Primary1	34	9	17	4	88.2%
Primary2	151	56	65	49	67.5%
struct	88	52	39	47	46.5%
biomed	3011	2646	*	2610	12.1%
avqs	159	124	*	116	27.0%
avql	802	753	*	747	6.9 %
ave.					36.9%

(* out of memory)

From Global Placement to Detailed Placement



Global Placement: Assuming all the cells are placed at the centers of global bins.



Detailed Placement: Cells are placed without overlapping.

Correlation Between Global and Detailed Placement

TestCase	WL_g	CON_g	WL_d	CON_d
highway2	12	8	18	13
fract	16	14	24	23
Primary1	140	125	151	141
Primary2	710	586	917	867
struct	150	110	261	227
biomed	667	1115	605	1084
avqs	180	149	258	214
avql	898	791	1032	909

- WL_g: Wirelength optimized global placement.
- CON_g: Wirelength optimized detailed placement.
- WL_d: Congestion optimized global placement.
- CON_d: Congestion optimized detailed placement.

Conclusion: Congestion at detailed placement level is correlated with congestion at global placement level. Thus reducing congestion in global placement helps reduce congestion in final detailed placement.

Conclusion

- Wirelength minimization can minimize congestion globally. A post processing congestion minimization following wirelength minimization works the best to reduce congestion in placement.
- We tested a number of congestion-related cost functions including a hybrid length plus congestion (commonly believed to be very effective). Experiments prove that they do not work very well.
- Net-centric post processing techniques are very effective to minimize congestion.
- Congestion at the global placement level, correlates well with congestion of detailed placement.

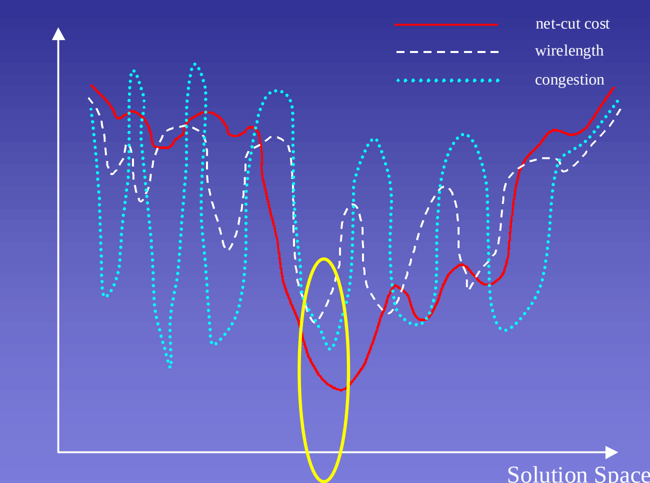
C Andrew B. Kahng
Majid Sarrafzadeh

Summary: Relationships Between the Three Cost Functions

- ☞ The net-cut objective function is more smooth than the wirelength objective function
- ☞ The wirelength objective function is more smooth than the congestion objective function
- ☞ Local minimas of these three objectives are in the same neighborhood.

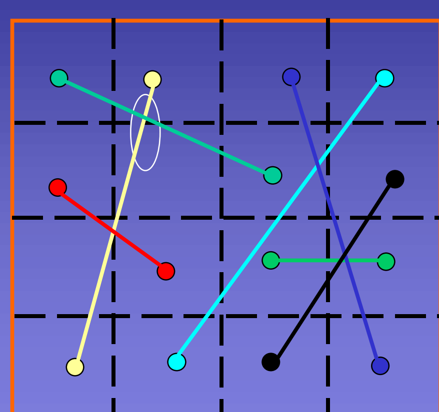
C Andrew B. Kahng
Majid Sarrafzadeh

Shapes of Cost Functions

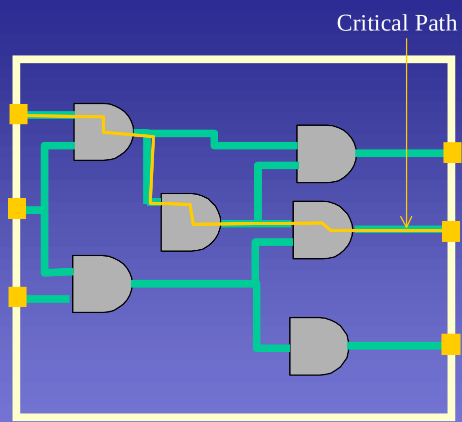


Crossing: A routability estimator?

- Replace each crossing with a “gate”
- A planar netlist
- Easy to place

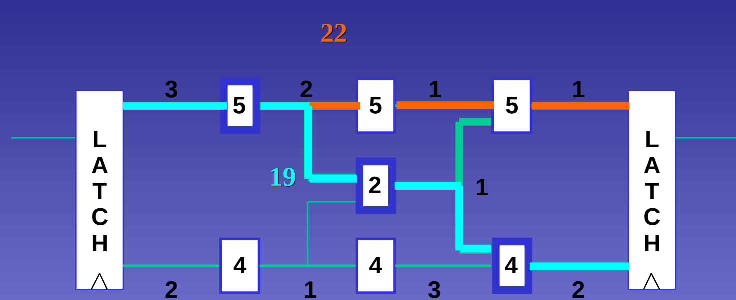


Timing Cost



- Delay of the circuit is defined as the longest delay among all possible paths from primary inputs to primary outputs.
- Interconnection delay becomes more and more important in deep sub-micron regime.

Timing Analysis



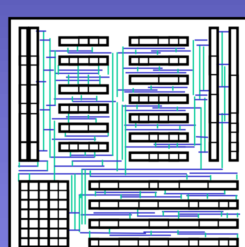
How do we get the delay numbers on the gate/interconnect?

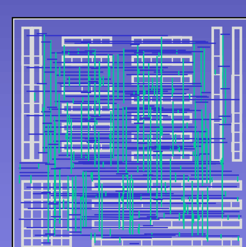
Approches

- Budgeting
  - In accurate information
  - Fast
- Path Analysis
  - Most accurate information
  - Very slow
- Path analysis with infrequent path substitution
  - Somewhere in between

Timing Metrics

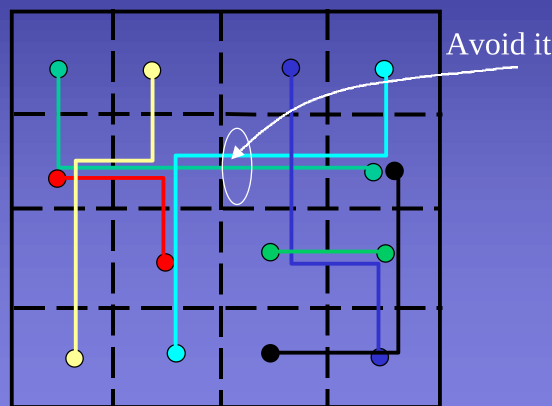
- How do we assess the change in a delay due to a potential move during physical design?
- Whether it is channel routing or area routing, the problem is the same
  - translate geometrical change into delay change





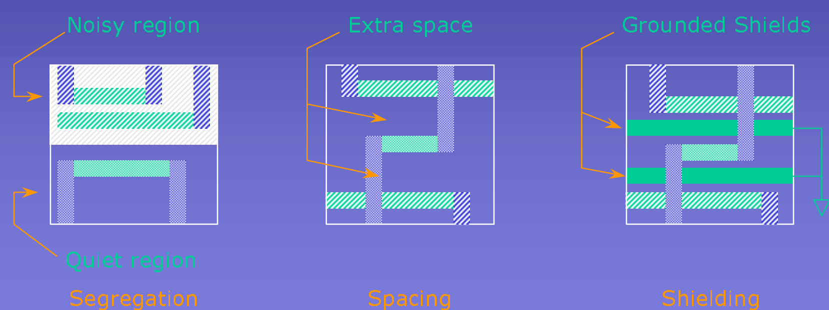
Others costs: Coupling Cost

- Hard to model during placement
- Can run a global router in the middle of placement
- Even at the global routing level it is hard to model it



Coupling Solutions

- Once we have some metrics for coupling, we can calculate sensitivities, and optimize the physical design...



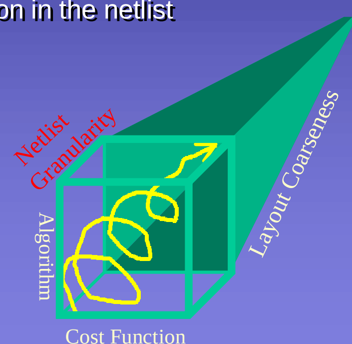
Other Performance Costs

- Power usage of the chip.
  - Weighted nets
  - Dual voltages (severe constraint on placement)
- Very little known about these cost functions and their interaction with other cost functions
- Fundamental research is needed to shed some light on the structure of them

© Andrew B. Kahng
Majid Sarrafzadeh

Netlist Granularity:
Problem Size and Solution Space Size

- The most challenging part of the placement problem is to solve a huge system within given amount of time
- We need to effectively reduce the size of the solution space and/or reduce the problem size
  - Netlist clustering: Edge extraction in the netlist



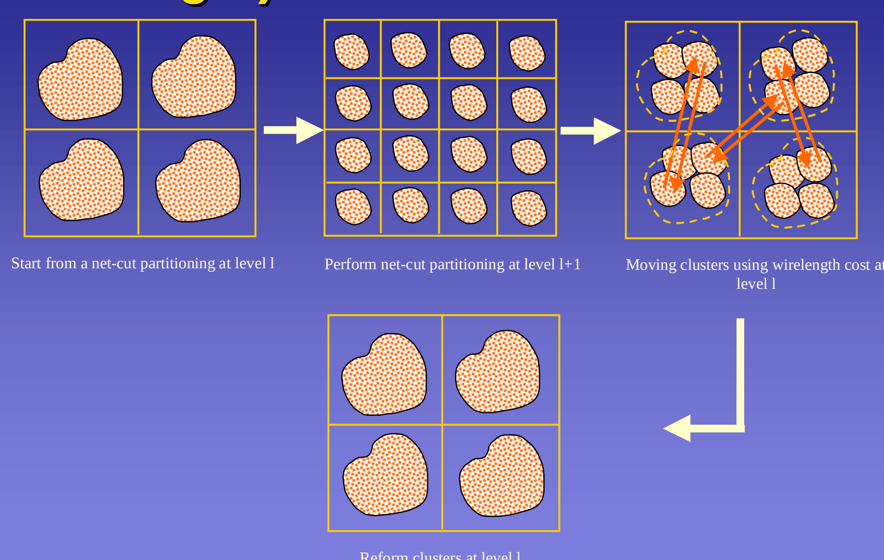
© Andrew B. Kahng
Majid Sarrafzadeh

Clustering (net-cut vs. wirelength)

- Big clusters should be formed based on net-cut cost
- Small clusters should take wirelength into account.
- According to the target size of the clusters, we should be able to choose the appropriate cost function for clustering

- Reuse a partitioner:

+1 level clustering (net-cut and wirelength)

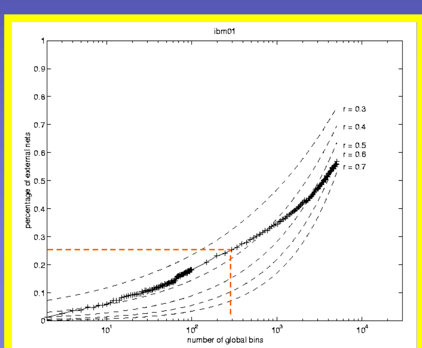


+1 Level Clustering Heuristics

- +1 level A: Use hMetis to get the net-cut optimized cell clusters at level h+1. Then perform the wirelength optimization at level h: Flat hMetis
- +1 level B: Use hMetis to get the net-cut optimized placement at level h. Then use hMetis to partition the subcircuit in each global bin into clusters. Then perform the wirelength optimization at level h: hierarchical (two-level) hMetis
- +1 level C: Use hMetis to get the net-cut optimized placement at the first level h_1. Then use hMetis to keep partitioning until we reach level h+1. Then do clustering at level h+1 and perform the wirelength optimization back at level h. hierarchical (multi) hMetis

	2 × 2		4 × 4		8 × 8		16 × 16		32 × 32	
technique	*WL*	runtime	*WL*	runtime	*WL*	runtime	*WL*	runtime	*WL*	runtime
WL.fast	1472	501	1517	542	2125	511	3588	490	3505	542
WL.slow	629	15709	703	16535	972	3834	879	14962	924	14846
cut opt.	384	499	596	542	847	668	1082	961	1339	1453
+1level A	409	262	737	244	1047	300	1313	384	1739	561
+1level B	385	281	649	244	876	282	1051	326	1204	524
+1level C	384	254	790	251	858	267	1028	323	1436	463

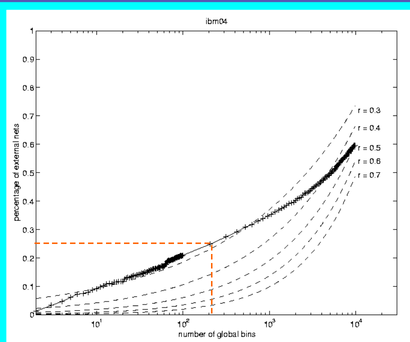
Wirelength and runtime comparison between different approaches for ibm01



Percentage of external nets vs. number of global bins: cut-only is good early-on, cut+WL later

technique	2 × 2		4 × 4		8 × 8		16 × 16		32 × 32	
	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	8635	570	11537	2433	12917	1460	15138	1065	15103	1080
WL.slow	6418	3921	8139	9088	6946	9344	7008	23068	7292	17156
cut opt.	5749	1377	6730	1409	7190	1658	7670	2552	8297	2276
+1level A	5816	884	6806	751	7731	824	7976	986	8820	1365
+1level B	5762	886	6714	700	7241	766	7436	846	7939	1123
+1level C	5789	882	6979	801	7437	754	7752	790	9543	1036

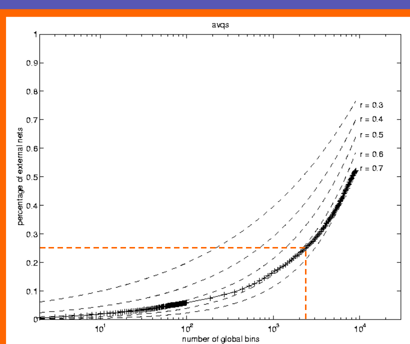
Wirelength and runtime comparison between different approaches for ibm04



Percentage of external nets vs. number of global bins

technique	2 × 2		4 × 4		8 × 8		16 × 16		32 × 32	
	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	1325	814	1472	716	1452	742	1472	724	1581	654
WL.slow	1077	1672	1173	1739	1064	1578	1350	1712	905	3241
cut opt.	225	355	319	363	409	411	496	424	656	533
+1level A	272	219	486	186	763	207	1017	260	1456	280
+1level B	310	217	406	179	516	196	656	215	765	262
+1level C	253	212	405	188	495	179	699	227	778	265

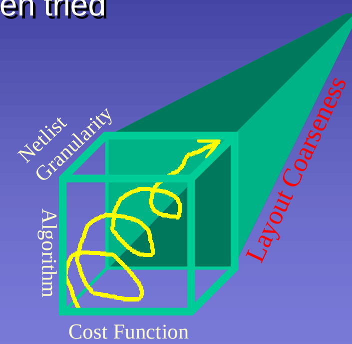
Wirelength and runtime comparison between different approaches for avqs



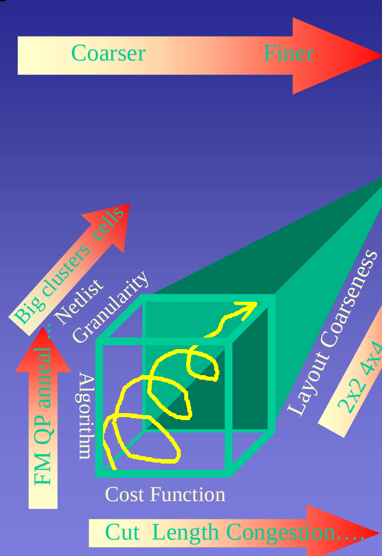
Percentage of external nets vs. number of global bins

Layout Coarseing

- Reduce Solution Space
- Edge extraction in the solution space
- Only simple things have been tried
  - GP, DP (Twolf)
  - 2x1, 2x2, ….
- Coarsen only "easy" parts



Summary of the Placement Cube

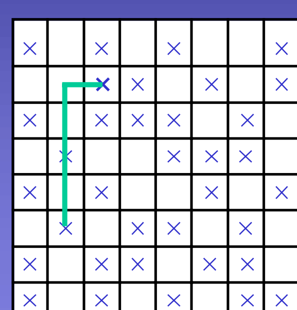


Incremental Placement

- Given an optimal placement for a given netlist, how to construct optimal placements for netlists modified from the given netlist.
- Very little research in this area.
  - Different type of incremental changes (in one region, or, all over)
  - Methods to use
  - How global should the method be
- An extremely important problem.

Incremental Placement

- A placement move changes the interconnect capacitance and resistance of the associated net
- A net topology approximation is required to estimate these changes



Routing Algorithms

Maze Routing

- Point by point routing of nets
- Route from source to sink
- Objective is to route all nets according to some cost function
- Most often, cost function tries to minimize congestion, route length, coupling, etc

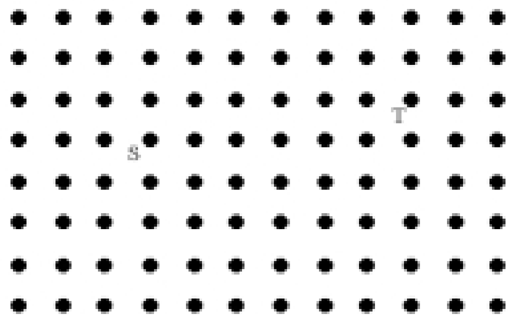
Maze routing algorithm

- Initialize priority queue Q, source S and sink T
- Place S in Q
- Get the lowest cost point X from Q, put neighbors of X in Q
- Repeat last step until lowest cost point X is equal to the sink T
- Rip and reroute nets

Rip and Reroute

- After all nets are routed, rip and reroute will select a number of nets based on a cost function to reroute
- Maze router focuses on minimizing congestion, therefore the rip and reroute finds nets that are routed through congested areas, removes net's routing and reroutes the net
- Rip and reroute is very important. It greatly improves the solution

Maze routing example



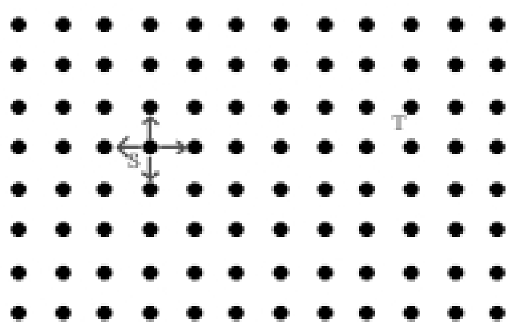
S denotes the source point
T is the sink point

127

ICCAD Tutorial: November 11, 1999

C Andrew B. Kahng
Majid Sarrafzadeh

Maze routing example



S denotes the source point
T is the sink point

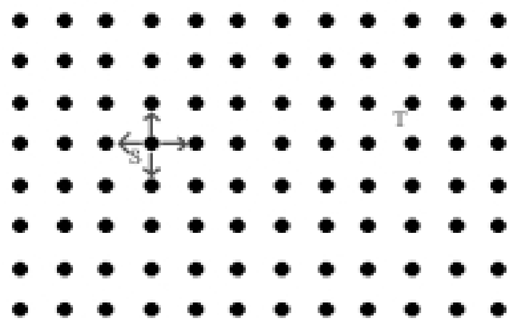
Search the neighbors of S

128

ICCAD Tutorial: November 11, 1999

C Andrew B. Kahng
Majid Sarrafzadeh

Maze routing example

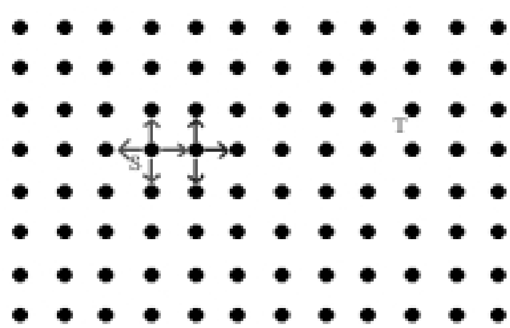


S denotes the source point
T is the sink point

Search the neighbors of S

Put neighbors into priority queue

Maze routing example



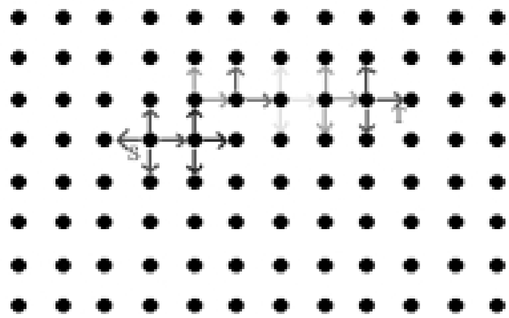
S denotes the source point
T is the sink point

Search the neighbors of S

Put neighbors into priority queue

Get the lowest cost point from the queue and search it's neighbors

Maze routing example



S denotes the source point
T is the sink point

Search the neighbors of S

Put neighbors into priority queue

Get the lowest cost point from the queue and search it's neighbors

Continue searching until sink is popped from the queue

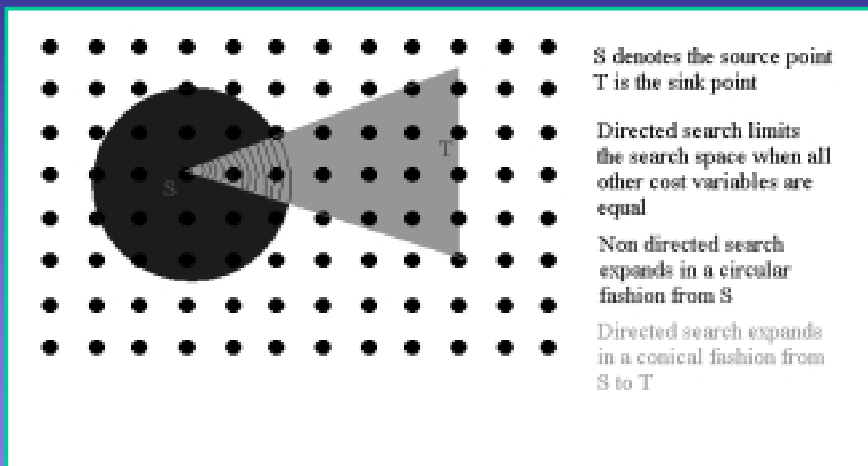
131 ICCAD Tutorial: November 11, 1999 (C) Andrew B. Kahng Majid Sarrafzadeh

Maze routing cost function

- Points can be popped from queue according to a multivariable cost function
- cost = function(overflow,coupling,wire length, …)
- By adding a distance to sink variable to the cost function, you get a directed search
- Directed search allows the maze router to explore the points around the direct path from source to sink first
- Directed search is optimal if “estimate” is a lower-bound to actual distance

132 ICCAD Tutorial: November 11, 1999 (C) Andrew B. Kahng Majid Sarrafzadeh

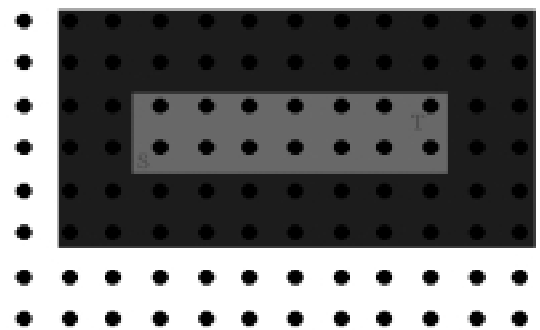
Directed search



Limiting the search region

- Since the majority of nets are routed within the bounding box defined by S and T, you can limit the number of points that the maze router will search to those within the bounding box
- This allows the maze router to finish sooner with little to no negative impact on the final routing cost
- Intuitively, you can see how this will decrease the runtime since the router will not consider points which are not likely to be on the route path. As stated before, any point outside the bounding box is unlikely to appear on the routing path

Restricting the search region



S denotes the source point
T is the sink point

Bounding box of S and T

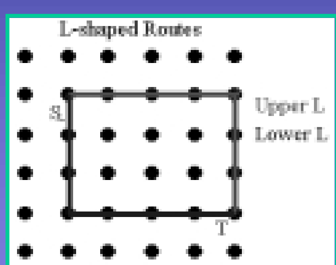
Normally, the search region is restricted to the bounding box + X

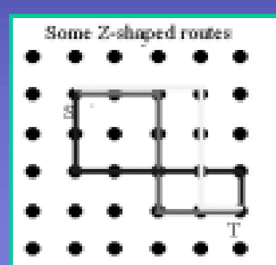
In this example, X = 2

The points outside of blue area are not considered by the maze router

Routing patterns

- Idea: restrict the routing of a net to certain basic templates
- Basic templates are L-shaped (1 bend) or z-shaped (2 bends) routes between a source and sink
- Templates allow fast routing of nets since you only consider certain edges and points





Experimental Results

- In our experiments, we tried using these templates for routing the n
- Our results showed that you can route x% of the small nets in l-shaped fashion without los congestion costs
- Unfortunetly, you can not route the largest nets using l-shaped routing without a dram increase in congestion (as compared to the congestion when you maze route every n Therefore, segment trees provide no advantage for l-shaped routin
- Below, the overflow results when the x% largest nets are locked and l-shape route

overflow costs - maze routing with x% 1-bend routing large nets -> small nets. Split into 2 terminals by mst							
filename	0%	5%	increase	5% ratio	10%	increase	10% ratio
p1.3	383	409	26	1.0678851	418	35	1.091383812
p2.3	675	874	199	1.2948148	947	272	1.402962963
avqs.2	3148	3870	722	1.229352	4150	1002	1.318297332
biomed.2	22	148	126	6.7272727	236	214	10.72727273
p1.2	70	104	34	1.4857143	105	35	1.5
p2.2	117	196	79	1.6752137	247	130	2.111111111
struct	233	280	47	1.2017167	331	98	1.420600858
biomed	3024	3307	283	1.0935847	3309	285	1.094246032
avqs	115	264	149	2.2956522	313	198	2.72173913
total	7787	9452	1665	n/a	10056	2269	n/a

Results

- The overflow results when the smallest x% (determined by bounding box) of the nets are l-shape routed. The remaining (1-x)% nets are maze routed. The smallest x% nets are locked i.e. they are not available for rip and reroute.
- The total time to route is reduced since l-shaped routing completes faster than maze routing and the locked nets allow rip and reroute to finish quickly
- When x% = 0, this is pure maze routing. When x% = 100, every net is l-shape routed
- An intuitive explanation: Since most of the routing of a net is done within it's bounding box, the nets with a small bounding box have little freedom with their routing. Thus, one of the l-shaped routes will be a good solution. On the other hand, a large bounding box gives more freedom to find a low cost path and the restrictive l-shaped route only considers two paths. Therefore, the l-shaped route is likely a bad routing choice.

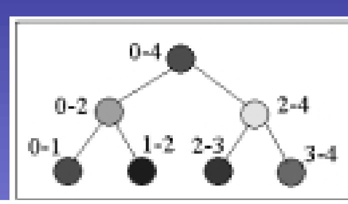
filename/x%	0	5	10	15	20	30	50	80	90	100
p1.3	379	379	379	379	379	376	376	380	378	460
p2.3	665	665	665	665	665	665	665	686	712	1099
avqs.2	2960	2960	3033	3033	3012	3016	2926	3170	2907	5224
biomed.2	15	15	15	15	15	15	15	15	16	644
p1.2	71	71	71	71	71	71	74	74	71	154
p2.2	109	109	109	109	109	109	109	115	108	315
struct	210	210	210	212	212	213	227	309	408	618
biomed	2994	2994	2994	2994	2994	2994	2985	2965	2951	3454

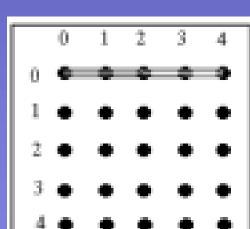
Quality of pattern routing

- If pattern route x% of the nets (where x is very small) same quality results is obtained
- Not much faster if the search graph is the same as maze routing graph (I.e. need to number grid points one-by-one)
- Need “super” data structures that allow quick jumps

Segment Trees

- Store routing segments in binary trees for fast of segments and congestion
- Routing area is divided into m horizontal trees and n vertical segment trees where m and n are the width and height, respectively, of the routing area





The above segment tree represents the horizontal track 0

Example: red node denotes a segment from (0,0) to (4,0)

C Andrew B. Kahng
Majid Sarrafzadeh

Segment Trees

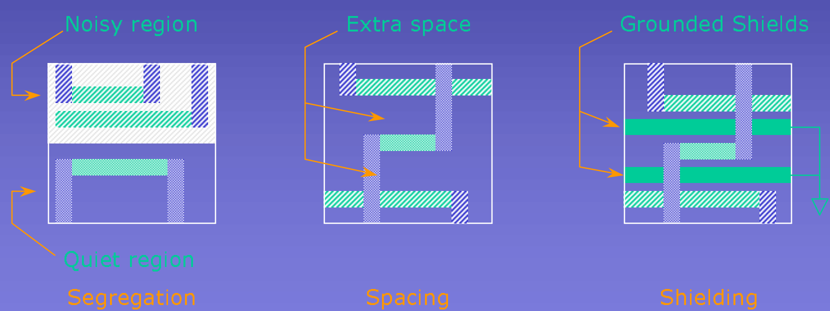
- The number of segments of any node can be retrieved in log n time where n is the length of the routing track
- Segment trees give you a quick global view at the routing of the nets
- Allows you to route long z or L-shaped nets much faster than traditional grid approach
- On a 100x100 grid, segment trees will route an L-shaped net faster if the net has a bounding box perimeter greater than 40
- Therefore, we want to route long nets with segment trees if it yields a cost similar to that of traditional maze routing

Coupling

- As fabrication sizes get smaller, coupling plays a larger role in timing
- Therefore, we want to minimize the number of long nets that are close to each other (on same route track)
- Segment trees keep this information
- Ways of using segment trees to reduce coupling during global routing?

Post-processing to avoid coupling

- Need good incremental router



Incremental Routing

- Very little work done in this area
- For example, with maze routing, if x (10) nets are changed, how many do we rip-up and reroute?

- Segment tree for incremental routing?

Routing

- Requirements for the DSM Router:
  - N-layer shape-based router
  - Supports gridless and gridded routing
  - Variable wire width for optimal delay constraints
  - Cross-talk avoidance, antenna effects
  - Clock tree sizing for tree balancing
  - Power routing sizing for voltage drop and electromigration
  - Power and clock routing resources reserved early
  - Activity-based optimization

Conclusion

- Maze routing is very effective and used in industry
- Need to “pattern” route many nets
- Need the right data structure to support pattern routing

(C) Andrew B. Kahng Majid Sarrafzadeh

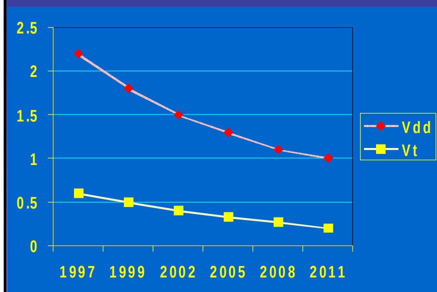
Placynthesis Algorithms

How to combine Placement & Synthesis

- Algorithmic challenges
- Does it make sense?
- Details in Part III

- There are “pure synthesis” operations
- And, there are placynthesis moves
- And there are “preliminary” placement moves (for cost function “smoothness”).

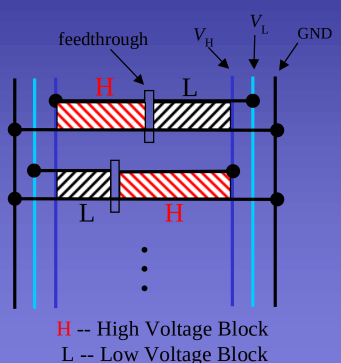
Many other Design Metrics: Power Supply and Total Power



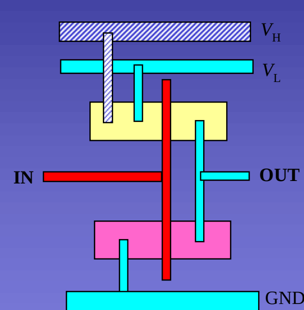
Source: The Incredible Shrinking Transistor, Yuan Taur, T. J. Watson Research Center, IBM, IEEE Spectrum, July 1999

Dual Voltages: A harder problem

- Layout synthesis with dual voltages: major geometric constraints



Layout Structure



Cell Library with Dual Power Rails

Conclusion

- There are so many problems that we do not understand.
- Innovation (in algorithms, methodology, tools, etc) needed in all facets.
- Modern physical design challenges are in fundamentals.