Placement: Key Step for Design Closure

Gi-Joon Nam
IBM Austin Research Lab
Credits: IBM PDS team, Placement team

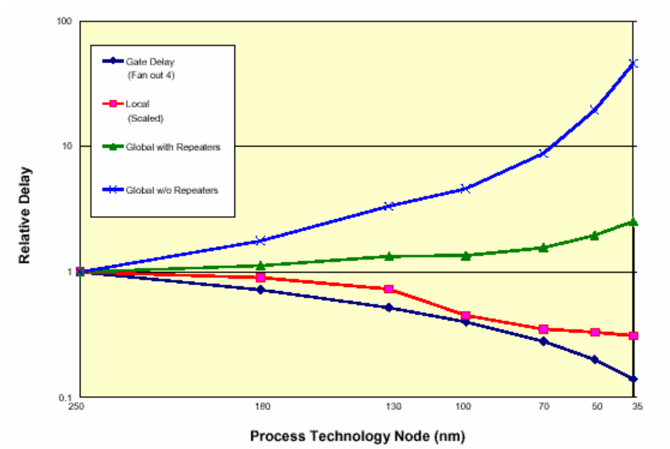
Outline of talk

- Placement in design closure flow
- Brief review of placement technology
- ISPD 2005 / 2006 placement contest & benchmark suite.
- Modern placement methodology
- Conclusion and future work

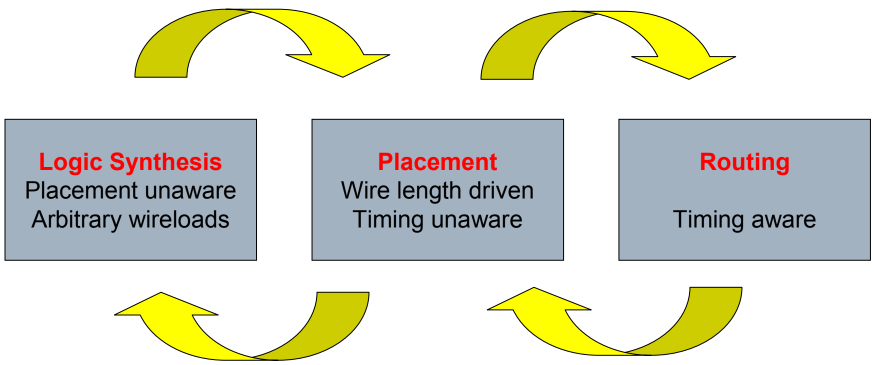
Design Closure Flow

- Microelectronics - ASICs.
  - Continued #1 industry ranking
  - Maintaining high-end technology
    - Highest gate count, advanced materials, IP
    - Record 35M-gate customer design
    - New cores (eDRAM, eFPGA, serial link) for SoC
    - Continued EDA tool innovations
    - Proven first-time-right methodology
  - Driven down into mid-range applications
    - IP collaboration adding 3rd party cores
    - Low-power technology for consumer designs
    - Easier design through standard tools
    - Semi-custom platform availability

Global Interconnect Dominance



Design Closure Problem

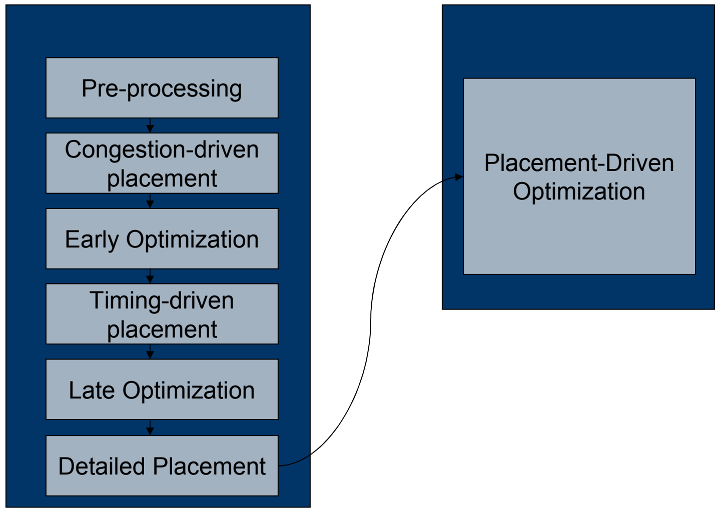


Convergence???

Design Closure Flow

- Eliminate iterations
- Reduce TAT
- Tight integration of relevant tools
  - Floorplanning/Placement
  - Logic transformations to correct timing
    - Gate Sizing
    - Buffering
    - Logic restructuring
    - Interconnection restructuring
    - Physical location change
  - Routing
    - Congestion-aware
    - Noise-aware

Design Closure Flow



Outline of talk

- Placement in design closure flow
- Brief review of placement technology
- ISPD 2005 / 2006 placement contest & benchmark suite.
- Modern placement methodology
- Conclusion and future work

Key Themes in Placement

- Placement problem consists of optimizing three orthogonal components:
  - Relative order
  - Spacing
  - Global position
- All within the context of routability, timing and signal integrity
- Placement within timing closure system is especially sensitive to stability

Placement Stability

- Algorithm produces similar results given similar input data
- Algorithm produces “scaled” results across a range of problem parameters (like density)
- Results are predictable

Placement Objective

- Find optimal relative ordering of cells
  - Minimize wire length and congestion
  - Maximize timing slack
- Find optimal spacing of cells
  - Eliminate wiring congestion problems
  - Provide space for post placement synthesis
    - Clock tree
    - Buffer insertion
    - Timing correction
- Find global optimal position

Overview of Common Placement Algorithm

- Simulated Annealing
- Recursive Partitioning
- Quadratic Placement
- Force-directed Placement

Simulated Annealing

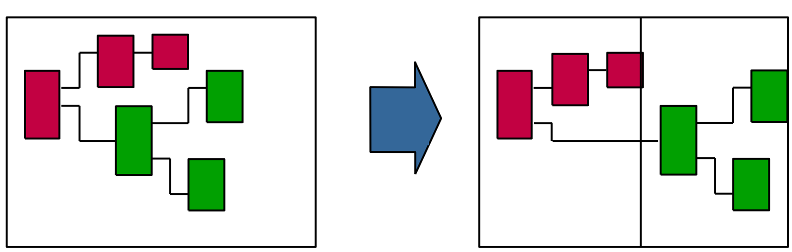
```
for (temp=high; temp > absolute_zero; temp -= increment)
{
        make a random move
        score the move
        use temp dependent probability to decide to accept or reject
}
```

Note: Clustering can be used to improve performance

- Great quality of results
  - Excellent relative ordering
  - Excellent spacing
  - Excellent global position
- Algorithm runtime is a problem
- Difficult to integrate with timing closure tools

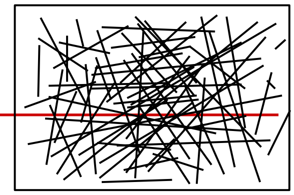
Recursive Partitioning-based Placement

- Given a set of interconnected blocks, produce two (four) sets that are of equal size such that the number of nets connecting the two sets is minimized

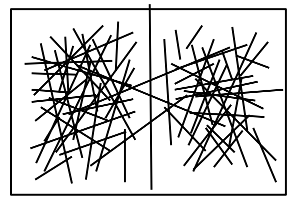


Recursive Partitioning-based Placement

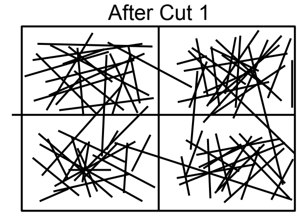
```
list_of_sets = entire_chip;
while(any_set_has_2_or_more_objects(list_of_sets))
{
    for_each_set_in(list_of_sets)
    {
        partition_it();
    }
    /* each time through this loop the number of   */
    /* sets in the list doubles.                   */
}
```



Initial Random Placement



After Cut 1



After Cut 2

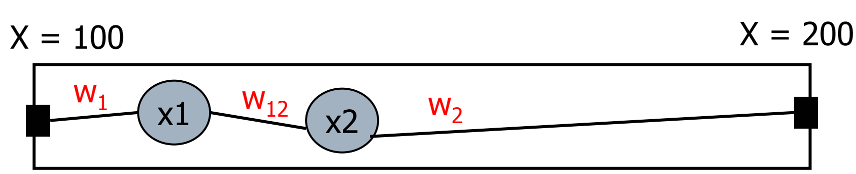
Recursive Partitioning-based Placement

- Finds correct cell ordering
- Poor spacing
- Poor position
- Lack of stability

Quadratic Placement

- PROUD [DAC88], GORDIAN [TCAD91], GORDIAN-L [DAC91], Vygen [DAC 97] implementation
- Minimize total squared wire length
  - $\Sigma\, w_{ij}\, \{ (x_i - x_j)^2 + (y_i - y_j)^2 \}$
  - Form and solve large Ax=b linear system
  - Called Quadratic Placement optimization (QP)
- But… Solutions will have overlaps
- Quadrisection will eliminate overlaps
- Recursive algorithm with Repartitioning refinements

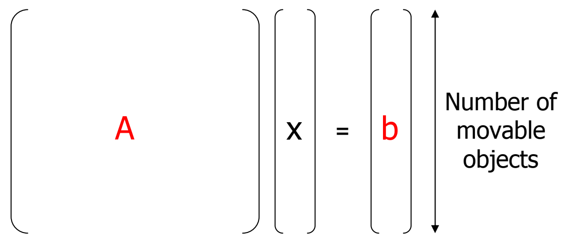
QP Mathematical Formulation



- Objective function: squared wire length
  - $f(X) = \underbrace{(x_1 - 100)^2}_{W_1} + \underbrace{(x_2 - 200)^2}_{W_2} + \underbrace{(x_1 - x_2)^2}_{W_{12}}$
- Set the derivative of $f(X)$ to 0, $df(X)/dX = 0$
  - $df(X)/dx_1 = 2(x_1 - 100) + 2(x_1 - x_2) = 0$
  - $df(X)/dx_2 = 2(x_1 - x_2) + 2(x_2 - 200) = 0$

QP Resulting Matrix Equations

- Thus Ax=b where $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, $b = \begin{pmatrix} 100 \\ 200 \end{pmatrix}$
- In general, a sparse linear system equation

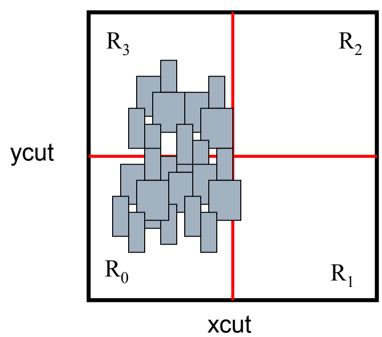


Quadratic Placement

- Why formulate the problem this way?
  - Known techniques make solution easy to find
  - There is only one solution (convex solution space)
  - The solution conveys "relative order" information
  - The solution conveys "global position" information
- However…
  - Solution is not legal, generally densely overlapping
    - Needs to be spread out
  - Solution depends on fixed anchor points
  - Solution minimize squared wire length, not linear wire length, congestion or timing

Quadrisection

- Geometric Partitioning

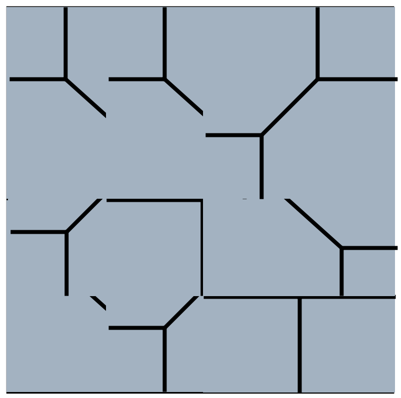


Given:
- cut spec
- a set of cells V
- for each $v \in V$
  (x(v), y(v))
  size(v)
- capacity for each sub-region
  $\kappa_0, \kappa_1, \kappa_2, \kappa_3$

Quadrisection: Geometric Partitioning

- Partitioning formulation
  - $f: V \rightarrow i \in \{0, 1, 2, 3\}$ such that
  - Meet capacity constraints
    - $\Sigma\{ \text{size}(v) \mid v \in V, f(v) = i \} \leqslant \kappa_i$ for $i=0,1,2,3$
  - Minimize weighted total movement
    - $\Sigma\ \text{size}(v) \cdot \text{distance}((x(v), y(v)), R_{f(v)})$

QP Refinements: Repartitioning



- 2 x 2 sliding window local refinement
- One pass goes over the entire placement region
- Keep iterating until the improvement is insignificant
- Sequence dependent

Outline of talk

- Placement in design closure flow
- Brief review of placement technology
- ISPD 2005 / 2006 placement contest & benchmark suite.
- Modern placement methodology
- Conclusion and future work

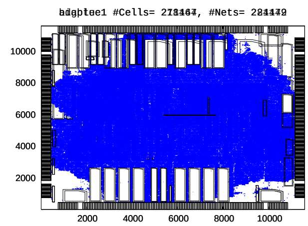
Trends in Placement

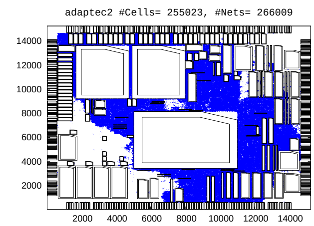
- Chips are larger
- Footprints are more diverse
- Free space is growing
- Interconnect delays are larger percentage of chip cycle time
- Placement is no longer a point tool
  - Core part of a timing closure system

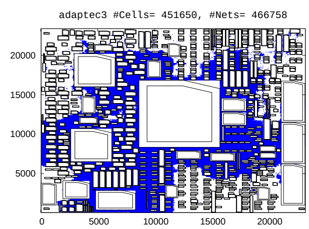
ISPD 2005 Placement Benchmark Suite

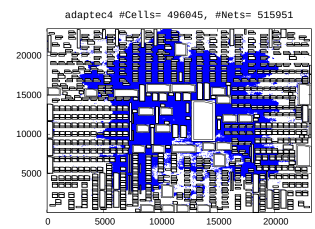
Name	#Objs	#Movs	#Fixed	#Nets	#Total Pins	#Pins from M	#Pins from F	Peri. IOs	Dens-ity%
adaptec1	211K	210904	543	221142	944053	923513	20540	480	75.71
adaptec2	255K	254457	566	266009	1069482	1045699	23783	407	78.59
adaptec3	452K	450927	723	466758	1875039	1843852	31187	0	74.53
adaptec4	496K	494716	1329	515951	1912420	1876563	35857	0	62.67
bigblue1	278K	277604	560	284479	1144691	1131856	12835	528	54.19
bigblue2	558K	534782	23084	577235	2122282	1979597	142685	0	61.80
bigblue3	1097K	1095519	1293	1123170	3833218	3790107	43111	0	85.65
bigblue4	2177K	2169183	8170	2229886	8900078	8710667	189411	0	65.30

ISPD 2005 Placement Benchmark Suite

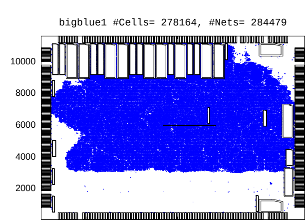


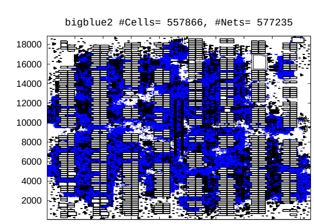


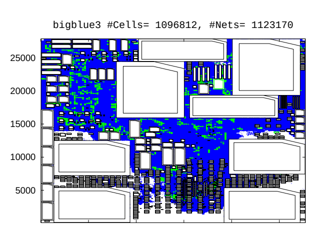


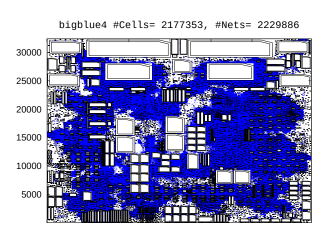


ISPD 2005 Placement Benchmark Suite









ISPD 2005 Placement Benchmark Suite

- Real industrial ASIC designs
- Free space
  - 54% - 85%
  - Affects wire-length significantly
- Macros
  - Wider distribution of cell sizes
- I/Os: perimeter and area-array I/Os
- Various row configuration
- Clock logic included

ISPD 2005 Placement Contest

- Open contest primarily for academic physical community
- Covers majority of placement tools
  - FastPlace, Capo, mPL, FengShui, APlace, NTUPlace, mFAR, Kraftwerk&Domino, Dragon
- Goals
  - To provide new modern placement benchmarks
  - To encourage to expose placement tools and results
  - To provide an educational forum on the state-of-the-art placement algorithms

ISPD 2005 Placement Contest

- Each team is given 5 days to come up with the best results
  - Fixed window of time
  - No limit on CPU resources
- Quality metrics
  - Legality
  - Half-perimeter bounding box wire length
  - No timing metric
  - No congestion metric

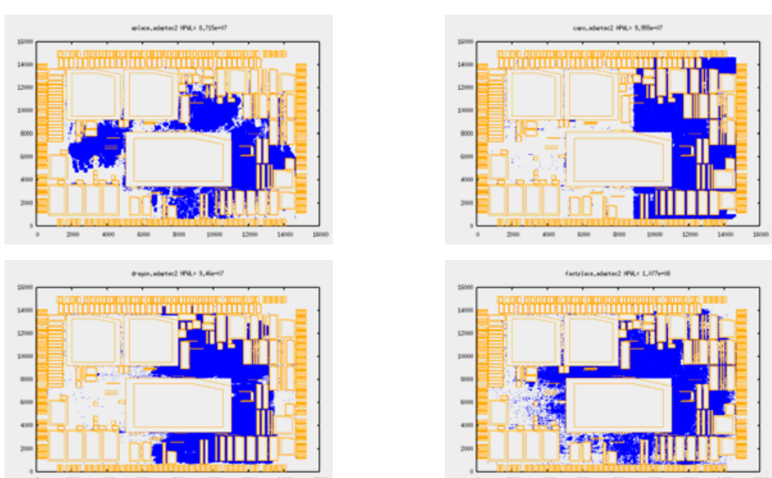
ISPD 2005 Placement Contest Result

	adaptec2	adaptec4	bigblue1	bigblue2	bigblue3	bigblue4	Ratio
APlace	87.31	187.65	94.64	143.82	357.89	833.21	1.00
mFAR	91.53	190.84	97.70	168.70	379.95	876.28	1.06
dragon	94.72	200.88	102.39	159.71	380.45	903.96	1.08
mPL	97.11	200.94	98.31	173.22	369.66	904.19	1.09
FastPlace	107.86	204.48	101.56	169.89	458.49	889.87	1.16
Capo	99.71	211.25	108.21	172.30	382.63	1098.76	1.17
NTUP	100.31	206.45	106.54	190.66	411.81	1154.15	1.21
fs50	122.99	337.22	114.57	285.43	471.15	1040.05	1.50
K&D	157.65	352.01	149.44	322.22	656.19	1403.79	1.84

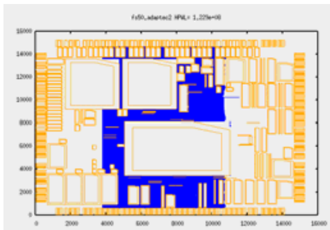
ISPD 2005 Placement Contest

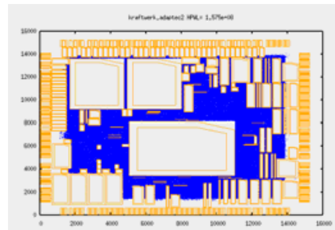
- New placement benchmark suite: ISPD 2005
  - Expected to be a standard benchmark suite
- Analytical placement dominance
  - Abundant free space
- Better quality metrics for future contest
  - Routability & congestion

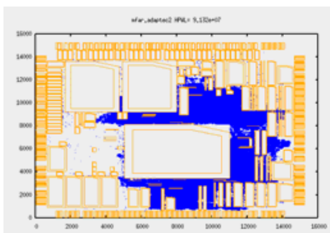
ISPD 2005 Placement Contest: adaptec2

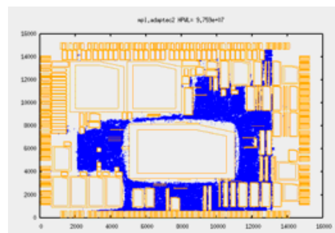


ISPD 2005 Placement Contest: adaptec2

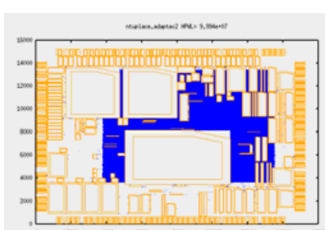








ISPD 2005 Placement Contest: adaptec2



Summary of ISPD 2005 Placement Contest

- 9 academic placement tools participated
  - Good coverage of placement tools
- 8 new placement benchmarks were released.
  - All were derived from real industrial ASIC designs
  - Extensively being used in placement research
- HPWL was used as sole quality metric
  - No routability estimation
  - No timing analysis
  - No runtime measurement
- Analytic placement tools dominated

A bit of Criticism

- "The contest, however, evaluated legality and wire length, not routability, which is a key concern for commercial placement tools"… EETimes 04/06/2005

- Rather high free space in benchmarks (i.e., low utilization)
  - Sort of favors analytic placement algorithm

ISPD 2006 Placement Contest

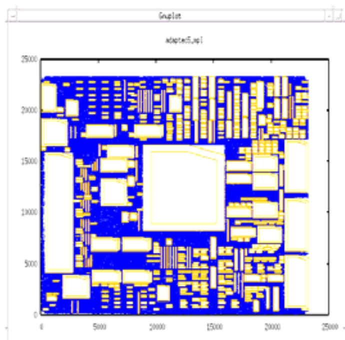
- 9 teams again
  - APlace3, Capo, DPlace, Dragon, FastPlace, Kraftwerk, mFAR, mPL6, Ntuplace
- Provide another suite of real placement benchmarks
- More advanced form of quality of metric
  - Legality
  - HPWL
  - Routability estimation via density target
  - Runtime
- Contestants submit executables and administrator runs them on new benchmarks

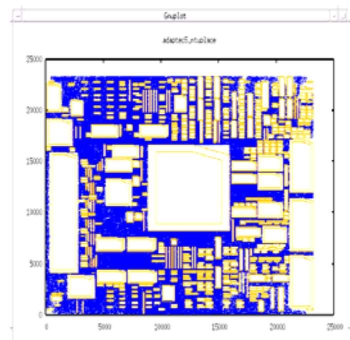
ISPD 2006 Benchmark Suite

Name	#Objs	#Movs	#Fixed	#Nets	Density %	Utilization %	Density Target%
adaptec5	843128	842482	646	867798	78.64	49.98	50
newblue1	330474	330137	337	338901	85.73	83.20	80
newblue2	441516	330239	1277	465219	86.14	61.66	90
newblue3	494011	482833	11178	552199	84.70	26.31	80
newblue4	646139	642717	3422	637051	65.72	46.45	50
newblue5	1233058	1228177	4881	1284251	74.54	49.56	50
newblue6	1255039	1248150	6889	1288443	59.27	38.78	80
newblue7	2507954	2481372	26582	2636820	76.46	49.31	80

adaptec5

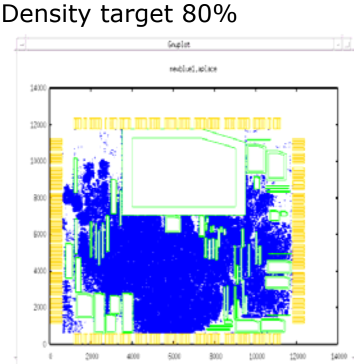
- 843K objects
- Density 79%, Utilization 50%
- Density target 50%

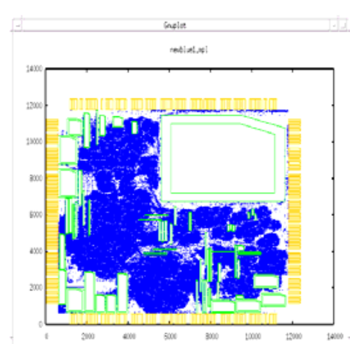




newblue1

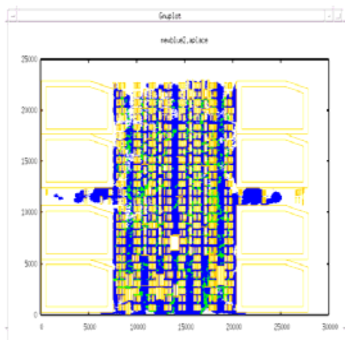
- 330K objects
- Lots of large movable macros
- Density 86%, Utilization 71%
- Density target 80%

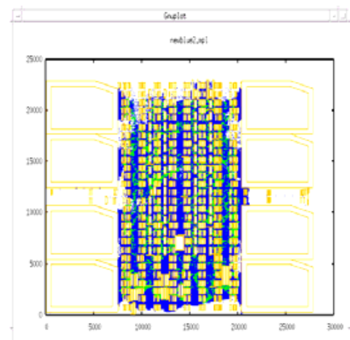




newblue2

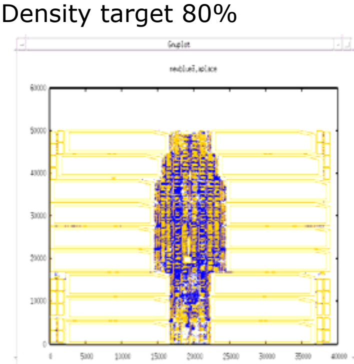
- 442K objects
- All standard cells were inflated by 2x
- 3.7K small movable macros (a few circuit row height)
- Density 86%, Utilization 62%
- Density target 90%

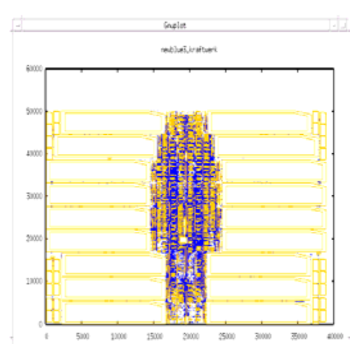




newblue3

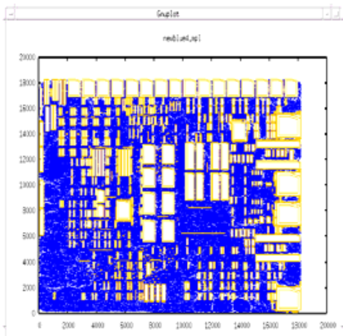
- 494K objects
- Interesting floorplan
- Density 85%, Utilization 26%
- Density target 80%

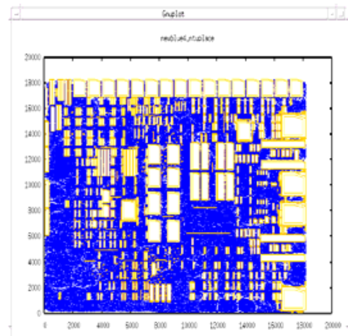




newblue4

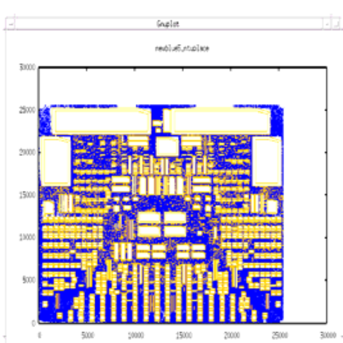
- 646K objects
- Density 66%, Utilization 46%
- Density target 50%

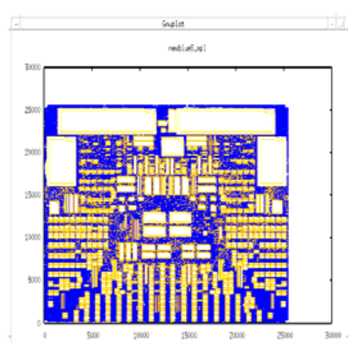




newblue5

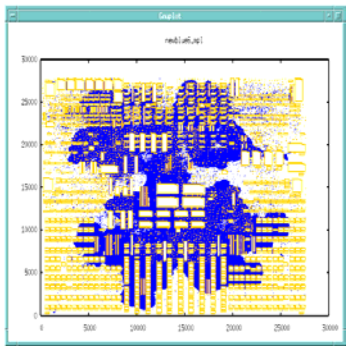
- 1233K objects
- Density 75%, Utilization 50%
- Density target 50%

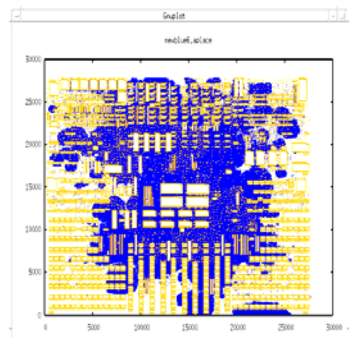




newblue6

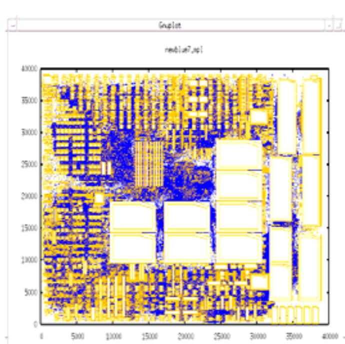
- 1255K objects
- Density 60%, Utilization 39%
- Density target 80%

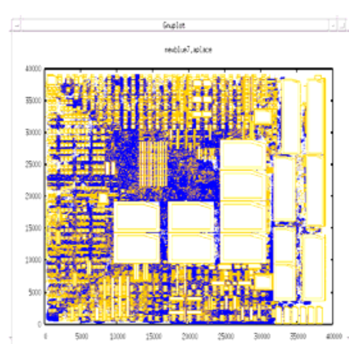




newblue7

- 2508K objects
- Density 76%, Utilization 49%
- Density target 80%





ISPD 2006 Placement Contest Results

	ad5	nb1	nb2	nb3	nb4	nb5	nb6	nb7	Avg.
kraftwerk	1.01	1.19	1.00	1.00	1.01	1.04	1.00	1.00	1.03
mPL6	1.00	1.06	1.07	1.17	1.00	1.02	1.00	1.00	1.04
ntuplace	1.02	1.00	1.07	1.16	1.03	1.00	1.04	1.07	1.05
mFAR	1.09	1.23	1.09	1.16	1.09	1.13	1.03	1.04	1.11
APlace3	1.26	1.20	1.05	1.13	1.35	1.21	1.06	1.05	1.16
Dragon	1.08	1.21	1.29	1.90	1.05	1.13	1.03	1.23	1.24
FastPlace	1.82	1.22	1.02	1.37	1.35	1.76	1.04	1.05	1.33
DPlace	1.26	1.55	1.77*	1.36	1.14	1.35	1.23	1.25	1.36
Capo	1.16	1.57	1.64	1.44	1.22	1.28	1.32	1.46	1.39

*Illegal solution with few overlaps on AMD platform, Legal solution on Intel platform

ISPD 2006 Placement Contest Results

	Avg. WL	Avg. Overflow Penalty%	Avg. CPU Factor%
kraftwerk	1.09	1.68	-5.04
mPL6	1.03	1.36	1.58
ntuplace	1.02	4.10	1.66
mFAR	1.11	2.71	-0.12
APlace3	1.10	3.82	5.31
Dragon	1.33	0.12	-5.90
FastPlace	1.18	22.09	-5.62
DPlace	1.34	9.32	-4.54
Capo	1.38	0.32	2.69

Results: What if CPU_factor is not included….

	ad5	nb1	nb2	nb3	nb4	nb5	nb6	nb7	Avg.
mPL6	1.00	1.06	1.01	1.05	1.00	1.04	1.00	1.00	1.02
ntuplace	1.00	1.00	1.03	1.06	1.02	1.00	1.03	1.09	1.03
kraftwerk	1.06	1.24	1.05	1.03	1.05	1.10	1.05	1.08	1.08
APlace3	1.21	1.15	1.00	1.00	1.28	1.19	1.01	1.01	1.11
mFAR	1.10	1.22	1.07	1.11	1.08	1.16	1.03	1.07	1.11
Dragon	1.16	1.27	1.32	1.92	1.14	1.26	1.10	1.30	1.31
Capo	1.15	1.55	1.56	1.32	1.21	1.27	1.29	1.40	1.34
FastPlace	1.87	1.33	1.07	1.33	1.43	1.86	1.11	1.14	1.39
DPlace	1.33	1.62	1.66	1.39	1.22	1.45	1.32	1.33	1.42

ISPD 2005 / 2006 Placement Contest

- Total 16 new placement benchmarks
  - All derived from real ASIC designs
  - Variety of floorplans
  - 5 benchmarks with more than million objects
- ISPD 2006 Contest
  - Indirectly address routability issue
  - Turn-around time
  - Improvements from ISPD 2005 results
- All benchmarks are available at ISPD website
- Can we include timing analysis into this flow?

Outline of talk

- Placement in design closure flow
- Brief review of placement technology
- ISPD 2005 / 2006 placement contest & benchmark suite.
- Modern placement methodology
  - BC-clustering [Alpert ISPD05]
  - Force-directed Placement
- Conclusion and future work

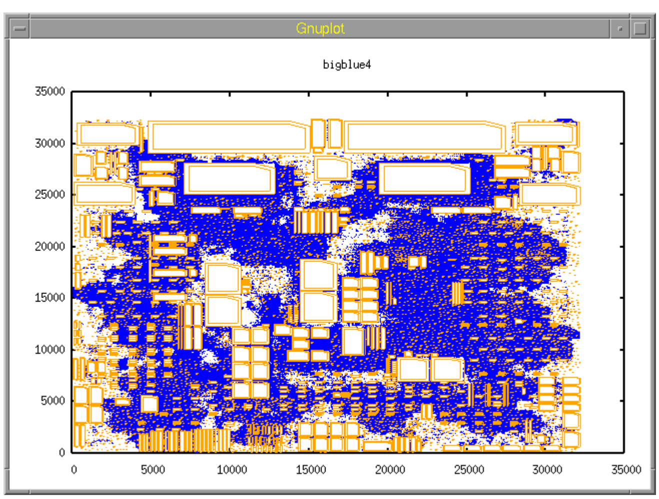
A Semi-Persistent Clustering Technique for VLSI Circuit Placement

Charles Alpert, Andrew Kahng, Gi-Joon Nam,
Sherief Reda, Paul Villarrubia
IBM Corporation & UCSD

SIA Roadmap

Year	1999	2002	2005	2008	2011	2014
Feature size(nm)	180	130	100	70	50	35
M trans/cm^2	7	14-26	47	115	284	701
Chip size (mm^2)	170	214	235	269	308	354
Singal pins/chip	768	1024	1024	1280	1408	1472
Clock rate (MHz)	600	800	1100	1400	1800	2200
Wiring levels	6-7	7-8	8-9	9	9-10	10
Power Supply(V)	1.8	1.5	1.2	0.9	0.6	0.6

Bigblue4 design from ISPD2005 Suite



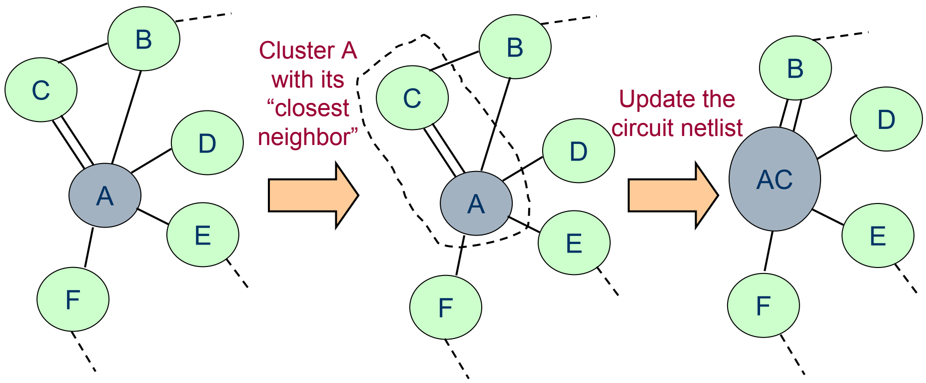
Trend Implications in Placement

- Scalability
  - Tractability
  - Runtime vs. quality trade-off
- SoC (System-on-Chip) designs
  - Mixed-size objects
  - White space

Problem Statement

- What is the most effective and efficient clustering strategy for analytic placement?
  - Quality of solution
  - CPU time

Clustering Concept

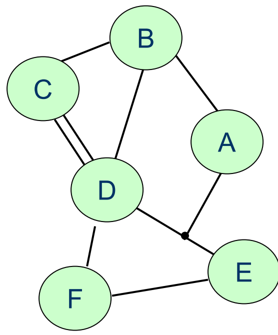


Clustering Score Function: $d(u, v) = \dfrac{\Sigma\, w_{ij} \cdot \text{conn}(u,v)}{[\ \text{size}(u) + \text{size}(v)\ ]^k}$

Clustering Literature

- Tremendous amounts of research here
  - Edge-Coarsening (EC)
  - First-Choice (FC)
  - Edge-Separability (ESC)
  - Peak-Clustering
  - Etc…
- General drawbacks
  - Clique transformation
    - Edge weight discrepancy
  - Pass-based iterations
  - Lack of global clustering view

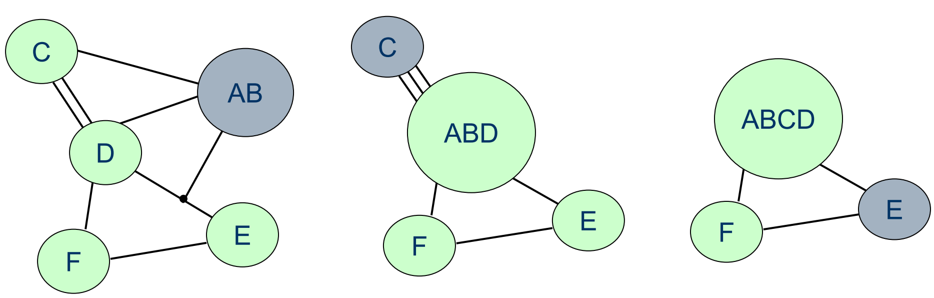
First-Choice Example



B=1/2
D=1/4
E=1/4

- Assume
  - N-pin net weight = 1 / n-1
  - Each object size = 1
  - Timing criticality is 1 for all nets
  - Random clustering sequence
    - A-B-C-D-E-F

First-Choice Example

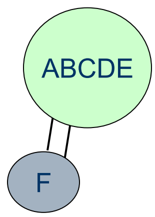


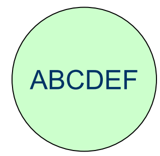
C=1/3
D=1/3 + 1/6
E=1/6

ABD=1/4+1/4+1/4

ABCD=1/5
F=1/2

First-Choice Example





ABCDE=1/5+1/5

Σ clustering_score = 2.65

Best-Choice Clustering

- Avoid clique transformation
- Avoid pass-based iterations
- More global view of clustering sequence
  - Priority-queue management
  - Lazy-update speed-up technique
- Area-controlled balanced clustering

Best-Choice Clustering

1. Initialize the priority-queue PQ:

   - For each cell *u*: calculate its clustering score *c* with its closest neighbor *v*.

   - Insert the pair (*u*, *v*) into PQ based on their cost *c.*

2. Until the target cell number is reached:

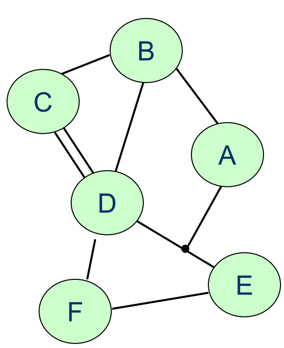
   - Pick the top of the PQ (*m, n*)

   - Cluster (*m*, *n*) into a new object *mn*; update the netlist

   - Calculate *mn* closest neighbor *k*; insert *(mn, k)* into PQ

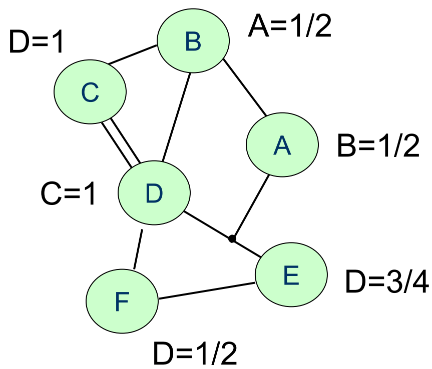
   - Recalculate the clustering cost of all the neighbors to *m* and *n*

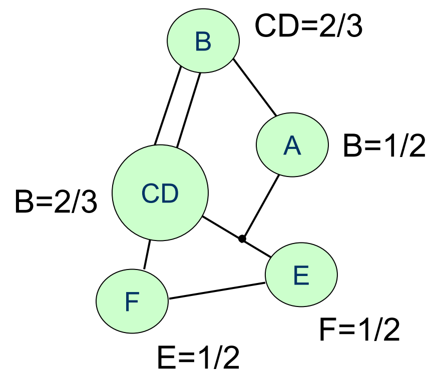
Best-Choice Example



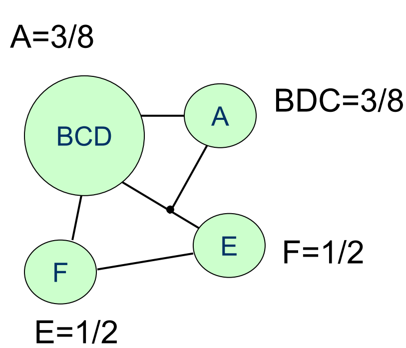
- Assume
  - N-pin net weight = 1 / n-1
  - Each object size = 1
  - Timing criticality is 1 for all nets

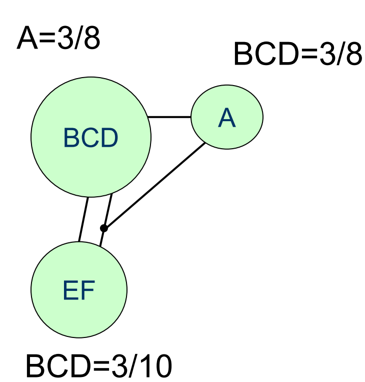
Best-Choice Example



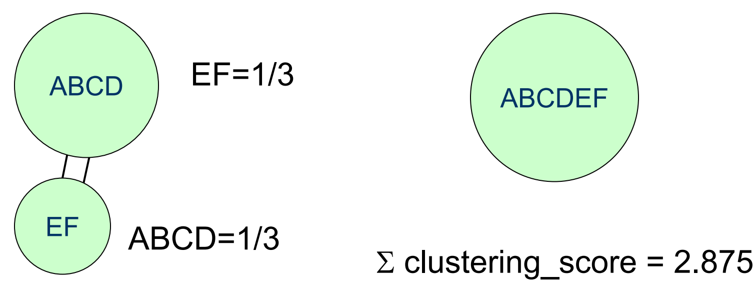


Best-Choice Example





Best-Choice Example

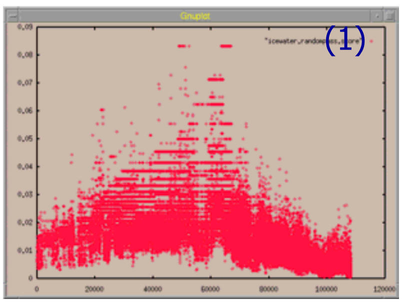


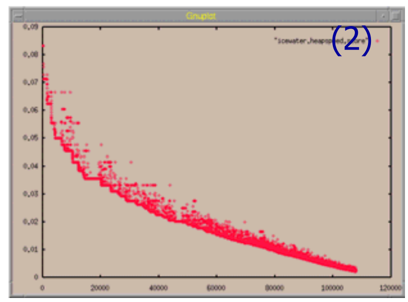
Best-Choice Clustering Summary

- Globally optimal clustering sequence via priority-queue data structure
  - Produce better quality of results
  - Clustering framework
    - Arbitrary clustering score function can be plugged in

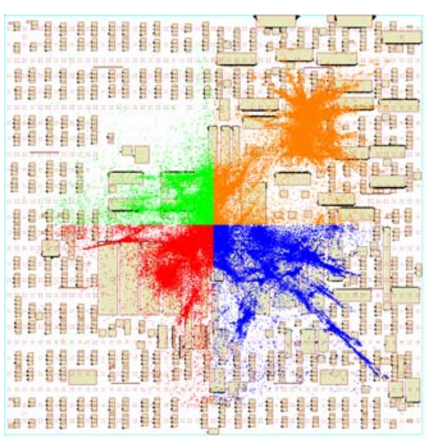
Best-Choice Clustering

- Clustering score distribution
  - First-choice (FC): Σ clustering_score = 5612.83
  - Best-choice (BC): Σ clustering_score = 6671.53

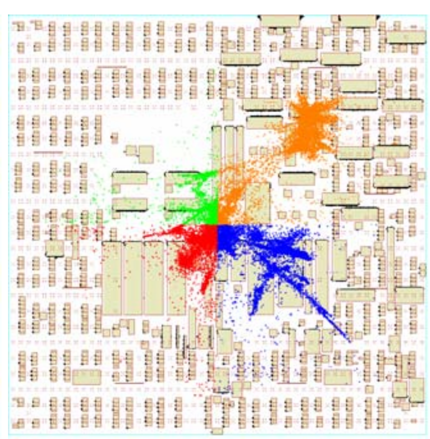




BC Clustering with Quadratic Placement

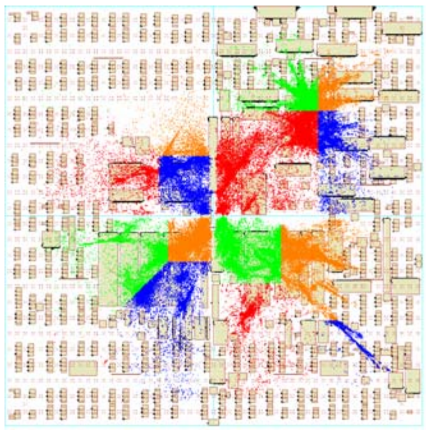


FLAT: 426K objects

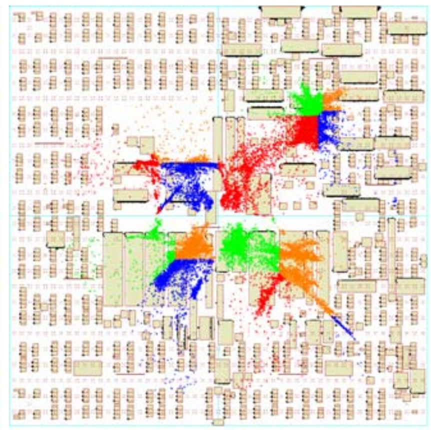


Clustering: 51K objects

BC Clustering with Quadratic Placement

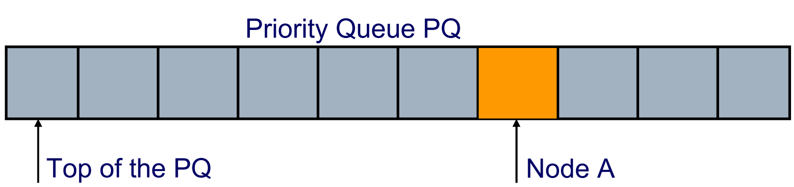


FLAT: 426K objects



Clustering: 51K objects

LazyUpdate Speed-up Technique



Observations:

1. Node A might be updated a number of times before making it to the top of the PQ (if ever), but the last update is what determines its final position in PQ
2. Statistics indicate than in 96% of our updating steps, updating node A score pushes A down in PQ

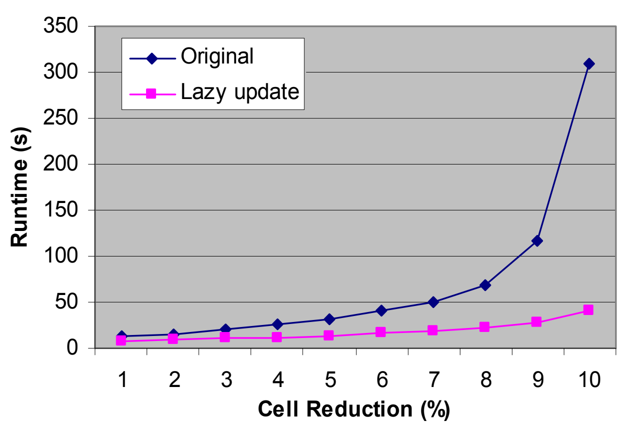
LazyUpdate Speed-up Technique

<u>Main Idea:</u> Wait until A gets to the top of the priority-queue PQ and then update its score if necessary

Until the target cell number is reached:
- Pick the top of PQ (*m, n*)
- If (*m, n*) is invalid then
  - recalculate *m* closest neighbor *n'* and insert *(m, n')* in PQ

  else
  - Cluster (*m*, *n*) into a new object *mn*; update the netlist
  - Calculate *mn* closest neighbor *k*; insert *(mn, k)* in PQ
  - Mark all neighbors of *m* and *n* invalid

LazyUpdate Runtime Characteristics



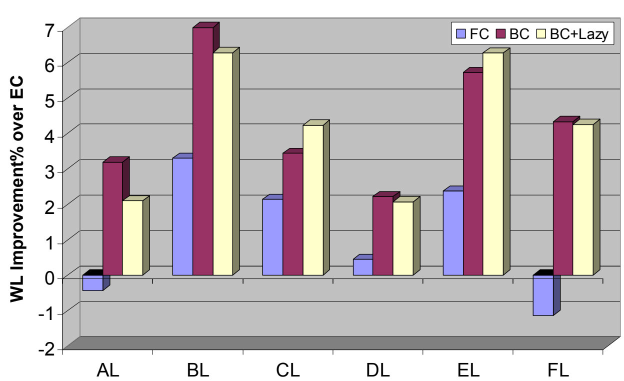
Note: Practically no impact to solution quality

BC: Experiment Setup

- IBM CPLACE
  - Analytic placement algorithm
  - Semi-persistent clustering paradigm
    - Up-front clustering
    - Selective unclustering during global placement
    - Full unclustering before detailed placement
  - Order-of-magnitude reduction by clustering
- Industrial ASIC designs
  - Size ranges from 56K to 880K placeable objects

BC: Placement Results

- Average 4.3% WL improvement over EC
- BC is x8.76 slower than EC



BC: Flat vs. BC+LazyUpdate Clustering

	WL(%)	CPU	CL-CPU%
AL(270K)	2.09%	0.40	1.17%
BL(276K)	-4.28%	0.52	1.35%
CL(351K)	3.27%	0.51	1.14%
DL(426K)	0.87%	0.45	1.35%
EL(456K)	1.59%	0.33	1.10%
FL(880K)	1.41%	0.46	1.68%
AD(389K)	8.23%	0.50	0.98%
BD(285K)	-0.34%	0.47	0.94%
CD(56K)	-0.36%	0.69	0.51%
Avg.	1.39%	0.48	1.14%

BC: Cluster Size Control

- $d(u, v) = \dfrac{\Sigma\, w_{ij} \cdot conn(u,v)}{[\, size(u) + size(v)\,]^k}$

Standard : $k = 1$

Automatic: $k = \lceil size(u) + size(v) / \mu \rceil$

where μ = expected avg. size

	Standard			Automatic		
	Max	Avg	WL%	Max	Avg	WL%
AD(84%)	14823	171.4	0.00	1140	160.4	-0.88
BD(86%)	28600	150.0	0.00	1140	114.6	3.71
CD(57%)	9060	113.5	0.00	610	109.8	30.05

BC: Conclusions

- Globally optimal clustering sequence framework
  - Independent of clustering score function
  - Better clustering sequence
  - Allow significant placement speed-up
  - Almost no loss of quality of solution
- Size control via clustering scoring function
  - Effective for dense design

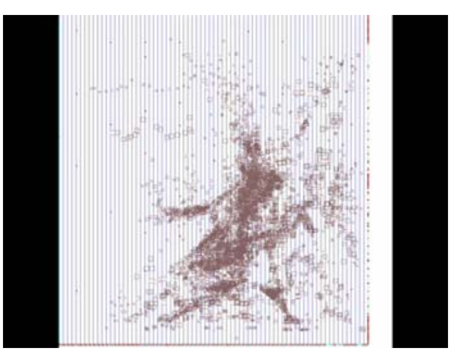
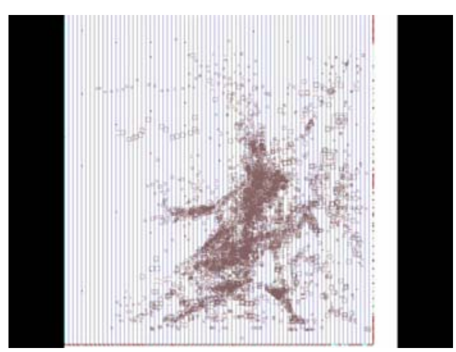
BC: Future Work

- Handling fixed blocks during clustering
  - Ignoring nets connected to fixed objects
  - Ignoring pins connected to fixed objects
  - Including fixed blocks during clustering
  - Etc…

Force-directed Placement: Brief Introduction

- Most recent breed of analytical global placement
- It has two main drivers:
  - Quadratic optimization to pull connected cells together.
  - Force computation to push cells apart.
- Different methods of applying spreading forces
  - Green function: Kraftwerk, FDP
  - Fixed point methods: mFar, FastPlace
  - Non-linear optimization functions: APlace, mPL
    - Approximation of linear wire length
    - Congestion penalty is part of objective function

Quadratic Placer vs. Force-directed Placer

Conclusion and Future Work

- Placement issues of today
  - Scalability
  - White space management
  - Mixed-sized placement
  - Congestion mitigation for routability
  - Tight timing constraints
- Be aware of known placement algorithm characteristics
  - Simulated annealing
  - MLP/FM partitioning-based approach
  - Analytical approach
  - Or combining these approaches

Conclusion and Future Work

- Know your data footprint
  - Know what footprint you're targeting, and have appropriate test case data
    - ASIC/SOC
    - Microprocessor
  - Must look at more than TWL and Congestion
    - Timing closure metrics are a must
- Don't overlook "Empty Space"
  - Introduces additional optimization dimensions
    - Spacing
    - Global position
- Tighter integration with logic synthesis
  - Puts extra emphasis on the need for "stability"