Section IV: Timing Closure Techniques

IBM Contributions to this presentation include:

- T.J. Watson Research Center
- Austin Research Lab
- ASIC Design Centers
- EDA Organization

*** For more detailed information see references at the end of this presentation, which include a wide variety of IBM and External publications covering these areas.**

Overview

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms (skip)
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques (skip)
- Congestion avoidance / mitigation techniques
- Routing optimization

Timing Closure

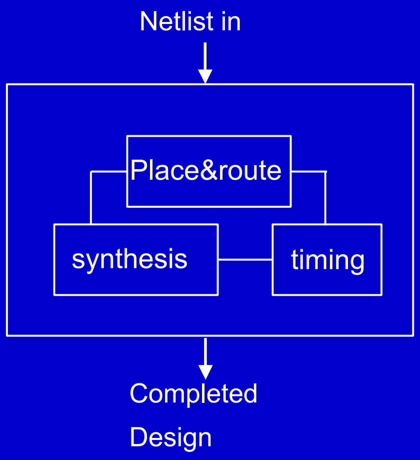
- Many aspects of a design contribute to performance, power, and density
  - Architecture / Logic Implementation
  - PD Design Style (Flat, Hierarchical, etc)
  - Clocking Paradigm / Test / Circuit Family
  - Floor Plan / Synthesis / Placement / Routing
- Design Automation for timing closure is more significant than ever before
  - Designs are larger
  - Wires are longer, invalidating statistical synthesis models, and requiring lots of buffers
  - Cycle times are more aggressive

Design Automation Tools are Individually Mature

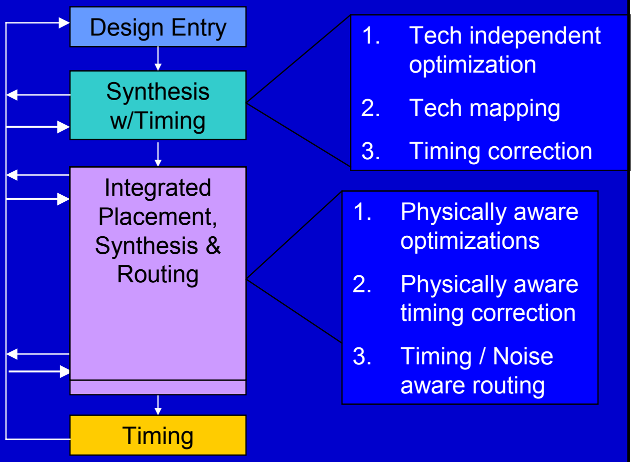
- Timing analysis
- Synthesis / Technology mapping
- Placement / Routing
- Floor Planning
- Extraction / Analysis

Challenge is to integrate them into one cooperative application

Physical Synthesis



Design Flow Evolution:



1. Tech independent optimization
2. Tech mapping
3. Timing correction

1. Physically aware optimizations
2. Physically aware timing correction
3. Timing / Noise aware routing

Purpose of this Section:

- Provide users with an intuitive feel of the inner workings of the major timing closure tools
- Demonstrate the advancements in timing closure tools technology via example designs
- Explore a variety of significant design choices

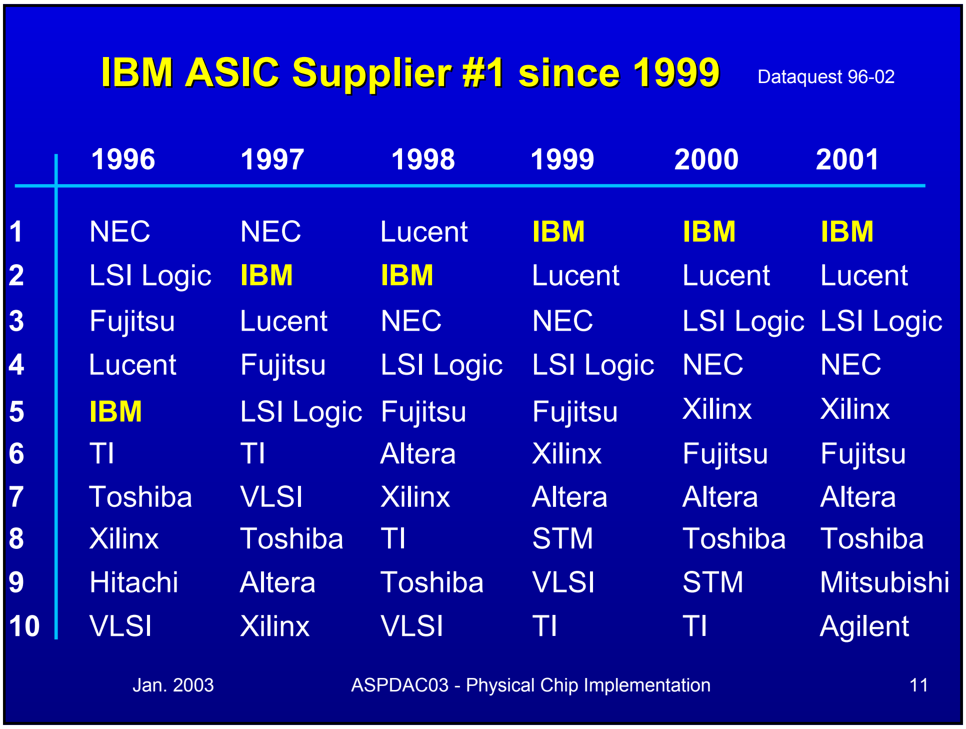
What you should expect:

- High level concepts presented are generally applicable across a wide range of tools / methodologies (ie: not IBM specific)
- Specific tool internals used in this tutorial are taken from IBM tools. They should provide a reasonable “feel” as to how things are done in the industry.

Worldwide ASIC/PLD Sales Top 5 Suppliers for 2001

IBM	$ 2758	growth	1.2%
Agere	$ 1310	growth	-43.5%
LSI	$ 1243	growth	-38.2%
NEC	$ 1243	growth	-35.2%
XLIINX	$ 1149	growth	-26.3%

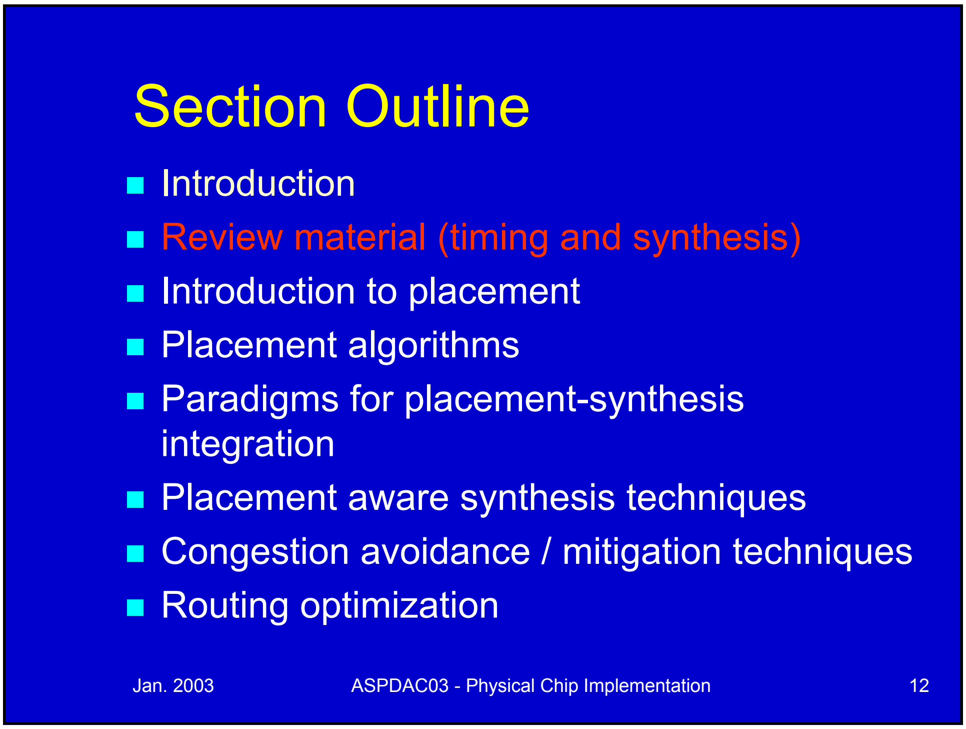
Revenue: Millions of U.S. Dollars
Source: Gartner Dataquest (March 2002)



IBM ASIC Supplier #1 since 1999

Dataquest 96-02

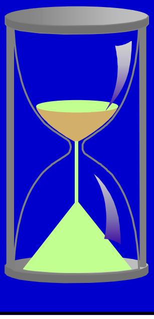
	1996	1997	1998	1999	2000	2001
1	NEC	NEC	Lucent	**IBM**	**IBM**	**IBM**
2	LSI Logic	**IBM**	**IBM**	Lucent	Lucent	Lucent
3	Fujitsu	Lucent	NEC	NEC	LSI Logic	LSI Logic
4	Lucent	Fujitsu	LSI Logic	LSI Logic	NEC	NEC
5	**IBM**	LSI Logic	Fujitsu	Fujitsu	Xilinx	Xilinx
6	TI	TI	Altera	Xilinx	Fujitsu	Fujitsu
7	Toshiba	VLSI	Xilinx	Altera	Altera	Altera
8	Xilinx	Toshiba	TI	STM	Toshiba	Toshiba
9	Hitachi	Altera	Toshiba	VLSI	STM	Mitsubishi
10	VLSI	Xilinx	VLSI	TI	TI	Agilent



Section Outline

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing optimization

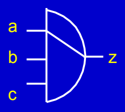
Static Timing Analysis



Timing Analysis Basics:

Why static timing since simulation is more accurate?

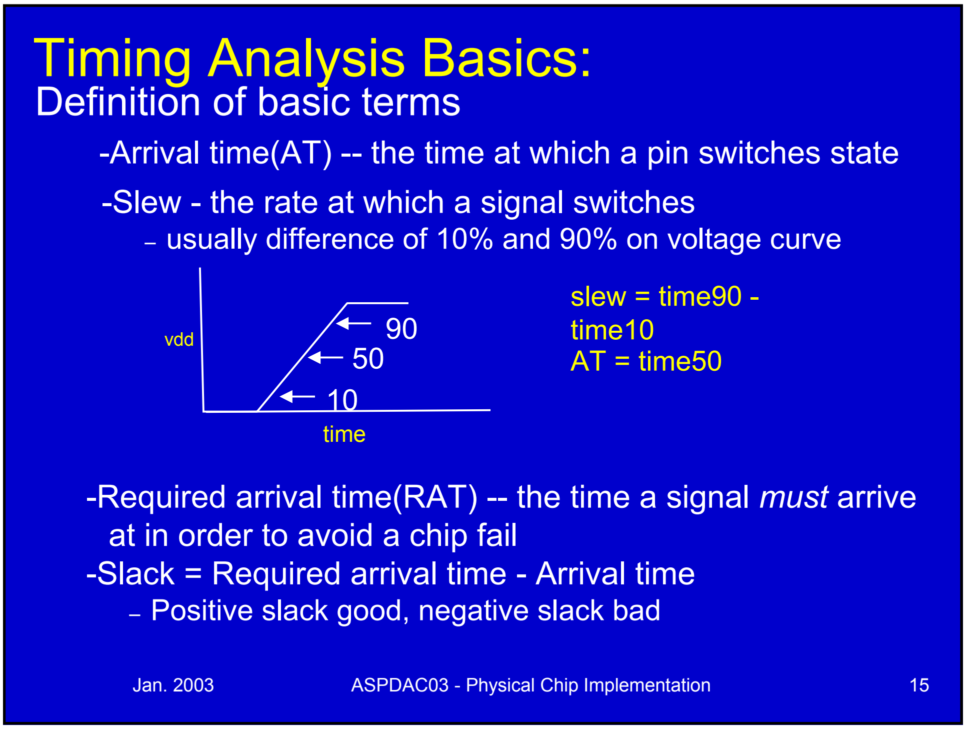
- Simulation has a number of key drawbacks
  - requires input state vectors
  - long runtimes
- A simple example:



How would one calculate the worst case rising delay from a to z?

	c=0	c=1
b=0	a-z delay1	a-z delay2
b=1	a-z delay3	a-z delay4

Exponential explosion as possible design input states grow!



Timing Analysis Basics:

Definition of basic terms

-Arrival time(AT) -- the time at which a pin switches state

-Slew - the rate at which a signal switches

- usually difference of 10% and 90% on voltage curve

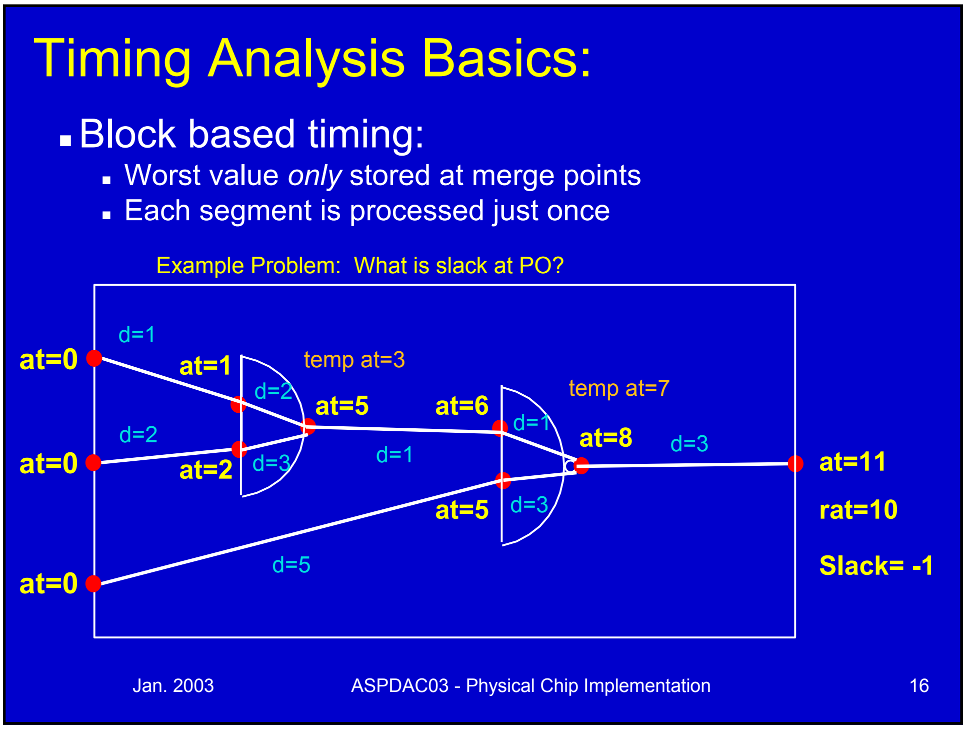
vdd
90
50
10
time

slew = time90 - time10
AT = time50

-Required arrival time(RAT) -- the time a signal *must* arrive at in order to avoid a chip fail

-Slack = Required arrival time - Arrival time

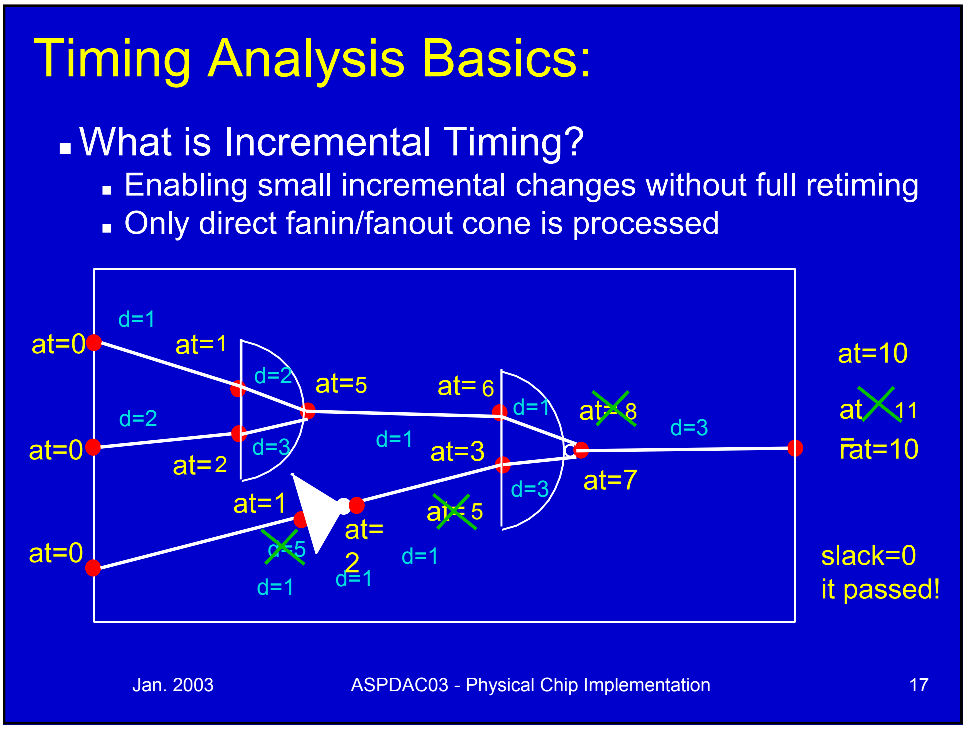
- Positive slack good, negative slack bad



Timing Analysis Basics:

- Block based timing:
  - Worst value *only* stored at merge points
  - Each segment is processed just once

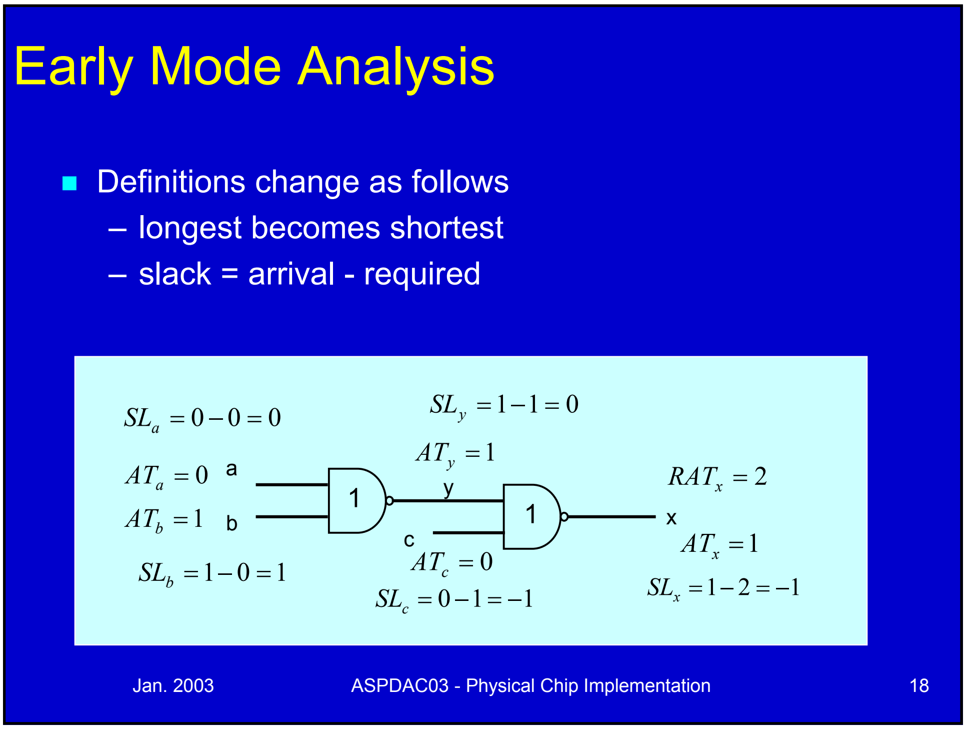
Example Problem: What is slack at PO?



Timing Analysis Basics:

- What is Incremental Timing?
  - Enabling small incremental changes without full retiming
  - Only direct fanin/fanout cone is processed

slack=0
it passed!



Early Mode Analysis

- Definitions change as follows
  - longest becomes shortest
  - slack = arrival - required

$SL_a = 0 - 0 = 0$

$AT_a = 0$

$AT_b = 1$

$SL_b = 1 - 0 = 1$

$SL_y = 1 - 1 = 0$

$AT_y = 1$

$AT_c = 0$

$SL_c = 0 - 1 = -1$

$RAT_x = 2$

$AT_x = 1$

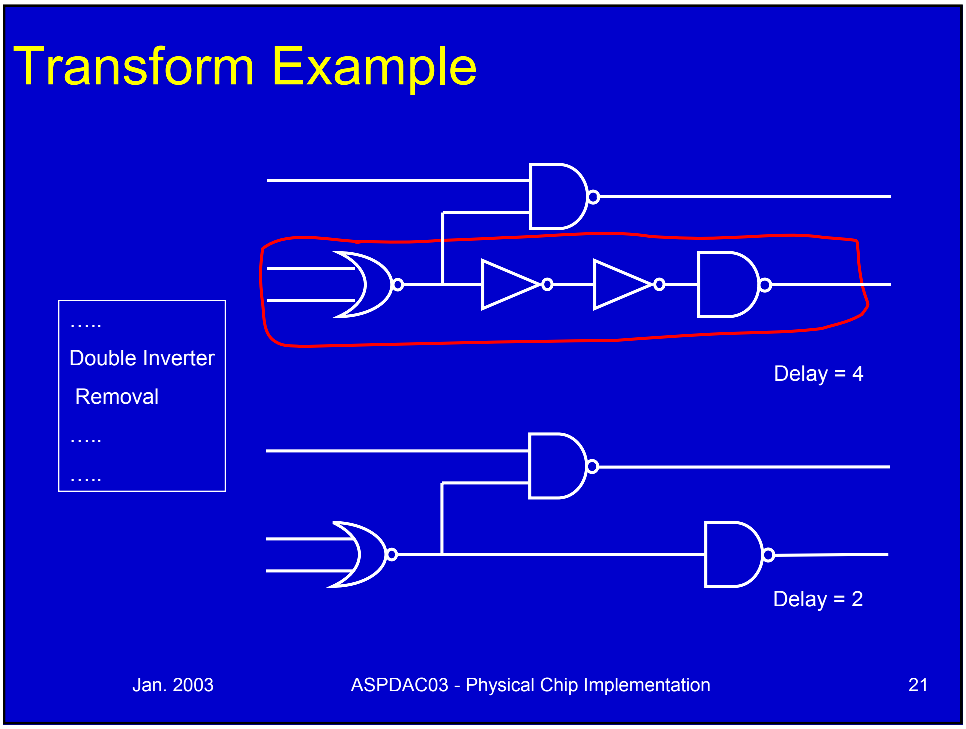
$SL_x = 1 - 2 = -1$

Timing Correction

- Fix electrical violations
  - Resize cells
  - Buffer nets
  - Copy (clone) cells
- Fix timing problems
  - Local transforms (bag of tricks)
  - Path-based transforms

Local Synthesis Transforms

- Resize cells
- Buffer or clone to reduce load on critical nets
- Decompose large cells
- Swap connections on commutative pins or among equivalent nets
- Move critical signals forward
- Pad early paths
- Area recovery



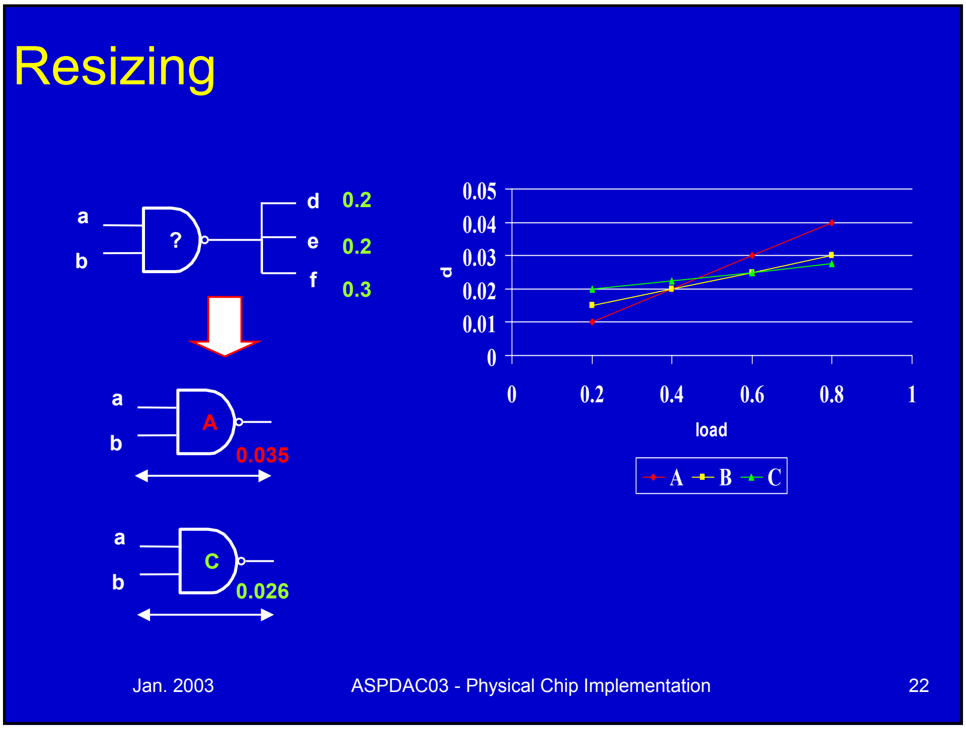
Transform Example

…..

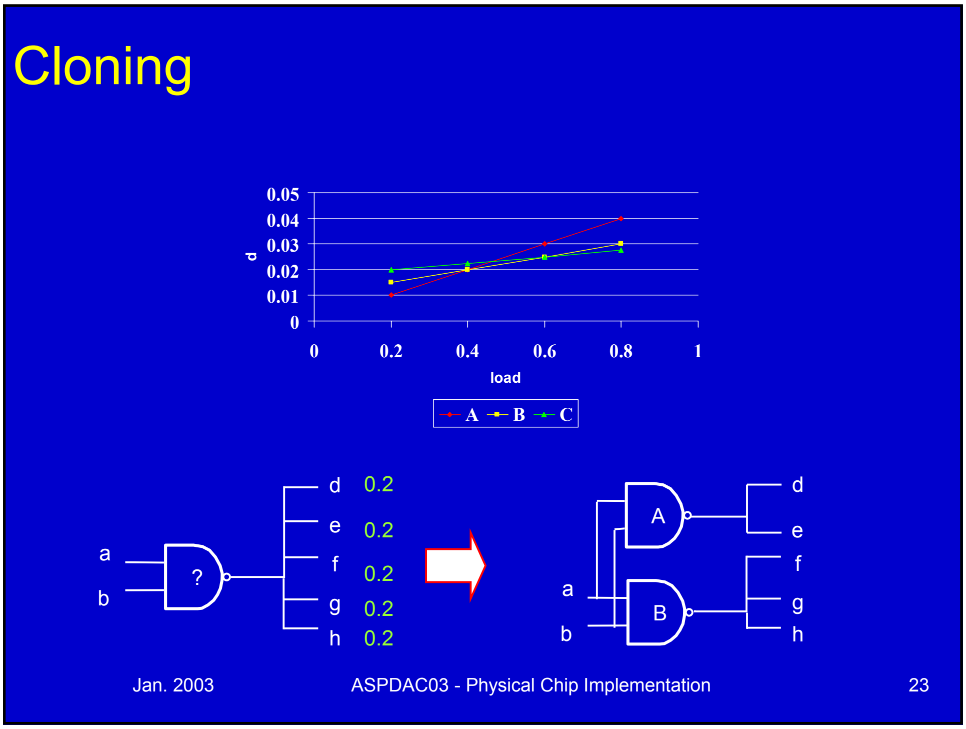
Double Inverter Removal

…..

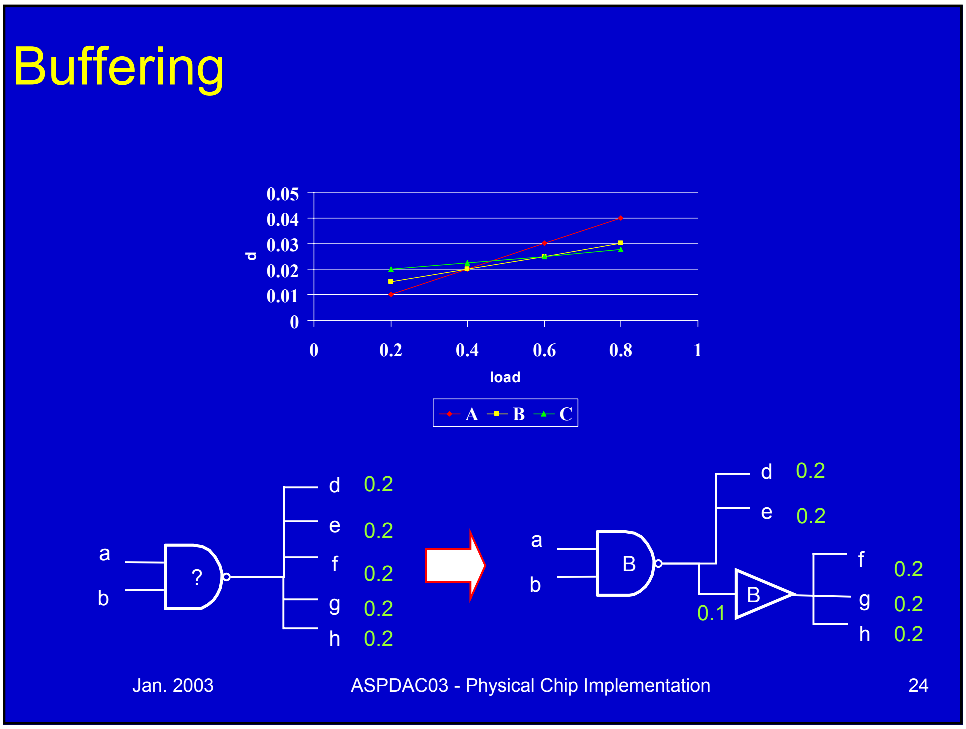
…..



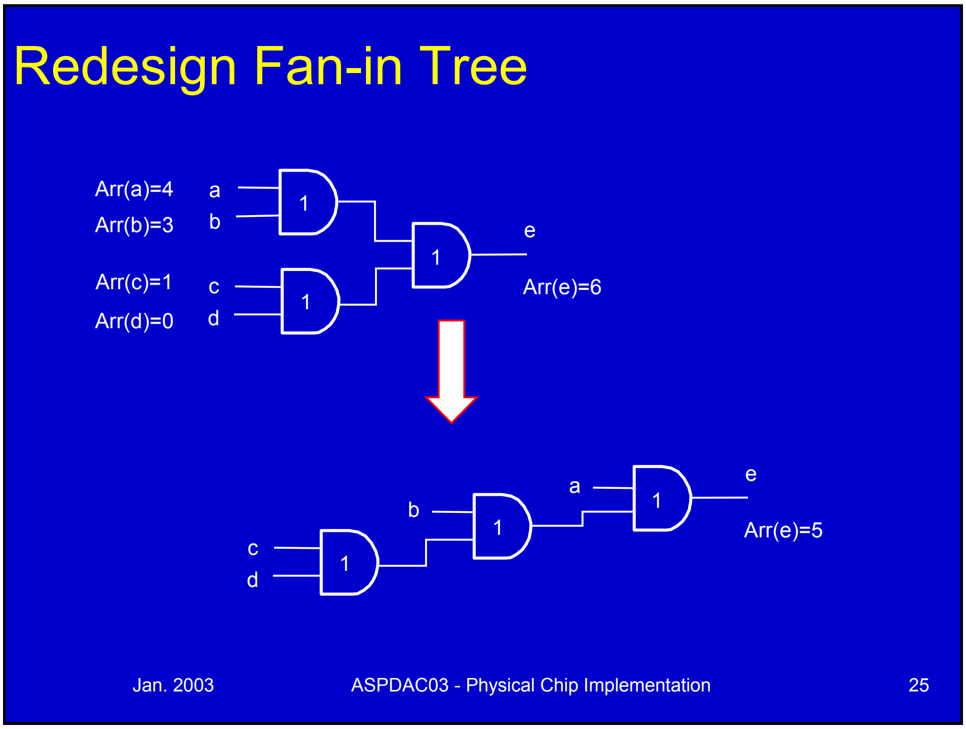
Resizing



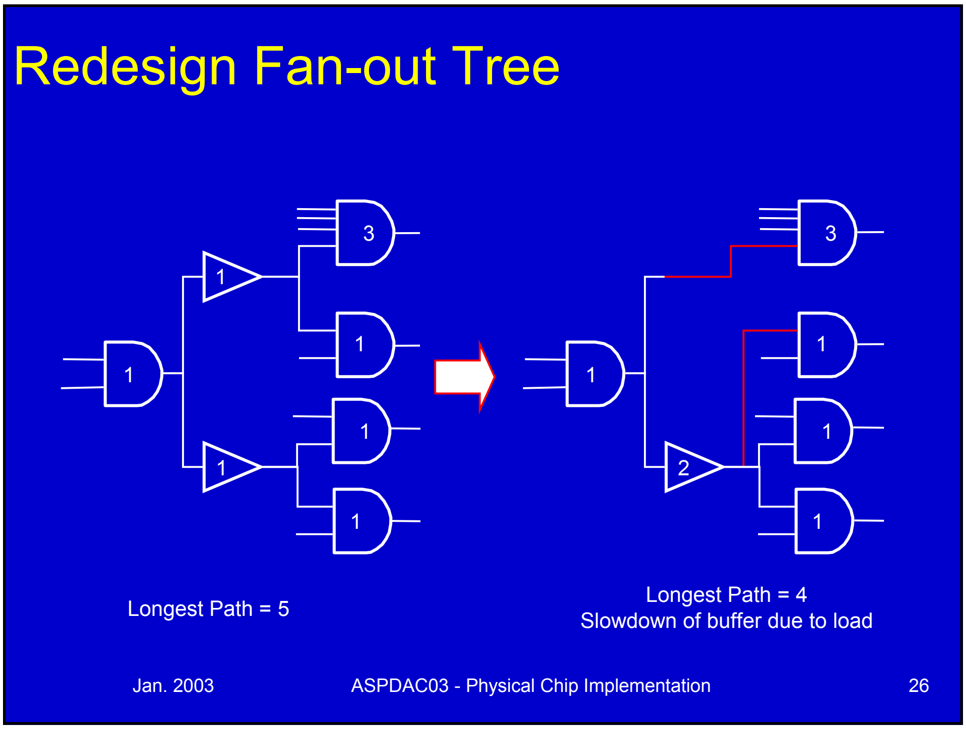
Cloning



Buffering



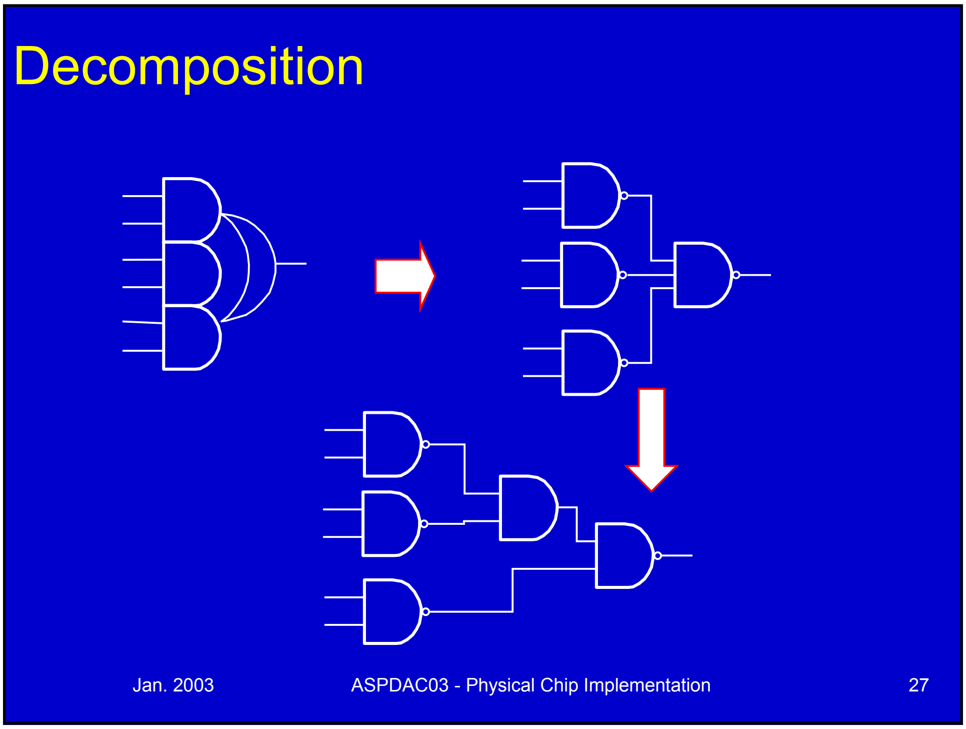
Redesign Fan-in Tree



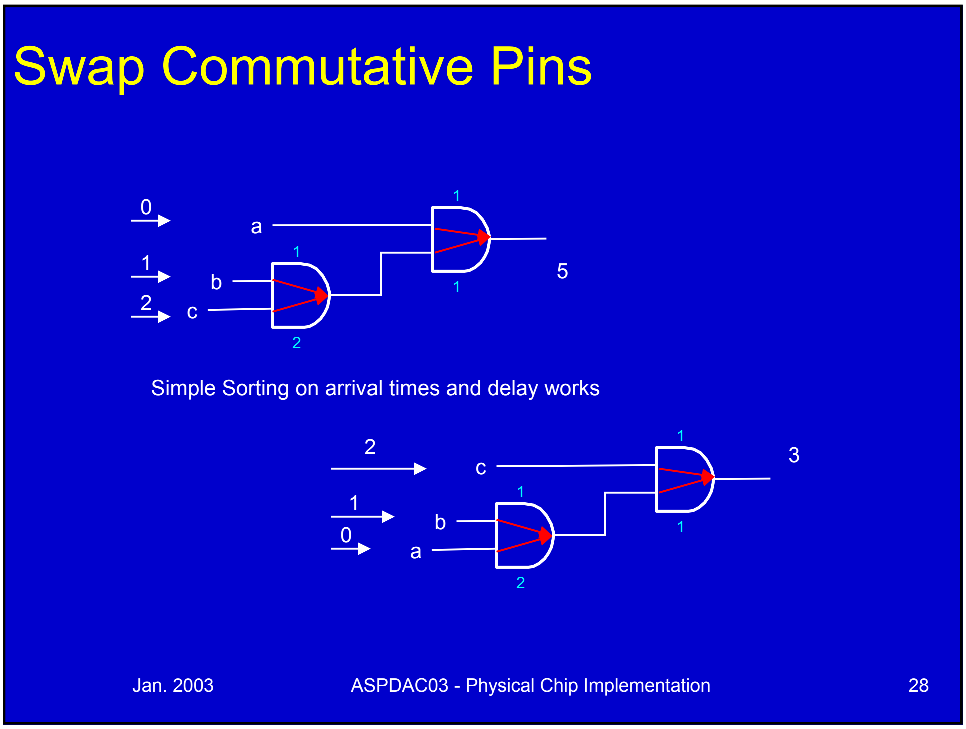
Redesign Fan-out Tree

Longest Path = 5

Longest Path = 4
Slowdown of buffer due to load

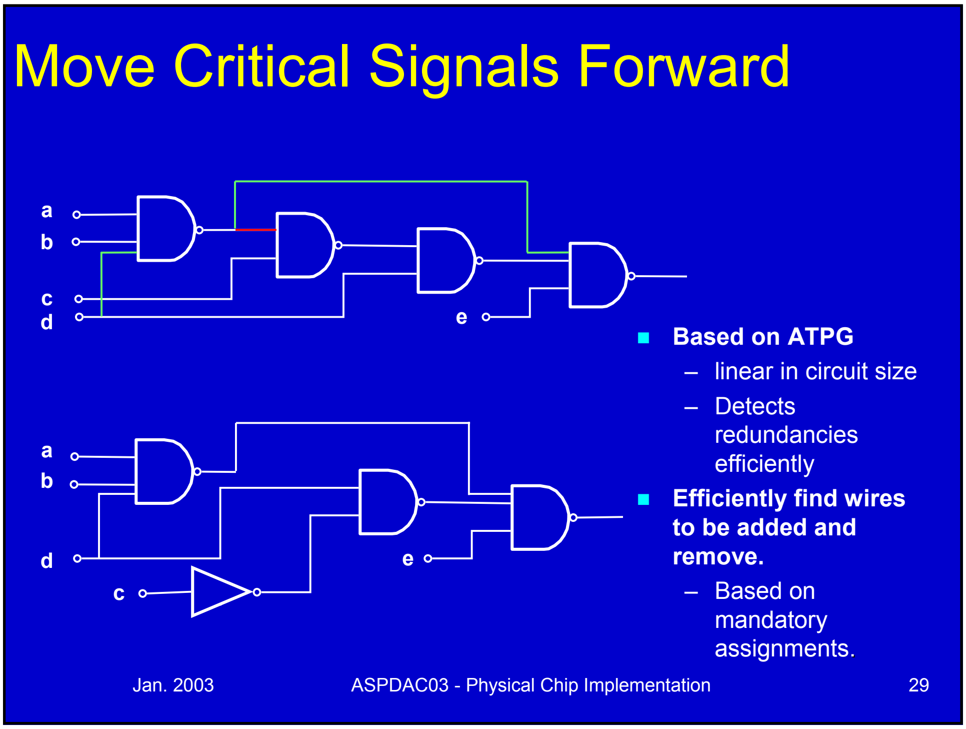


Decomposition



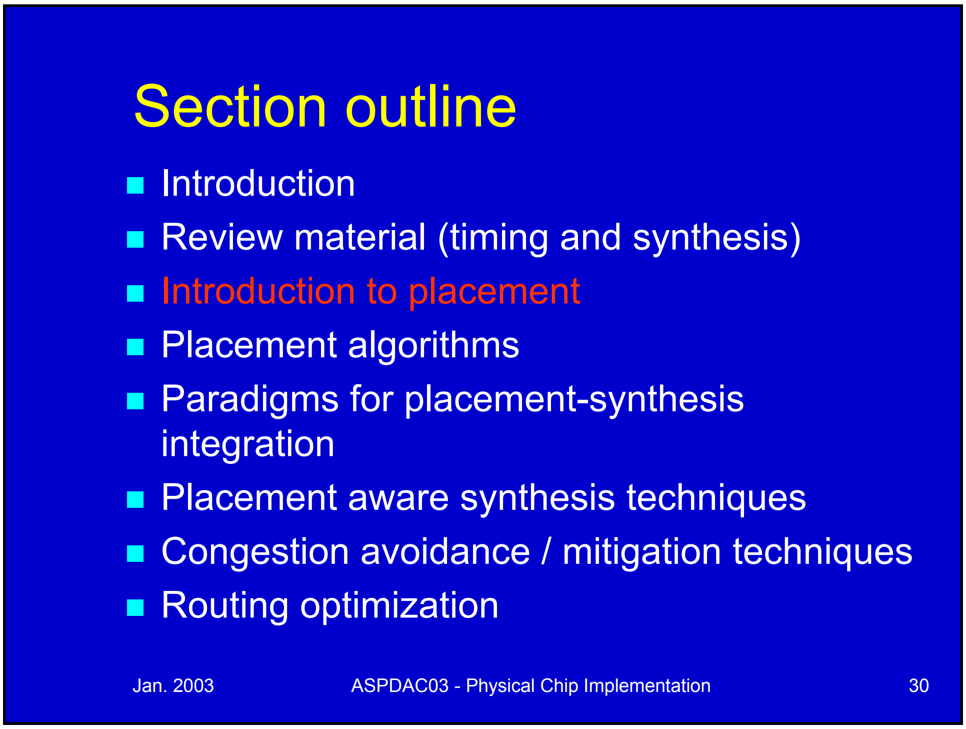
Swap Commutative Pins

Simple Sorting on arrival times and delay works



Move Critical Signals Forward

- Based on ATPG
  - linear in circuit size
  - Detects redundancies efficiently
- Efficiently find wires to be added and remove.
  - Based on mandatory assignments.



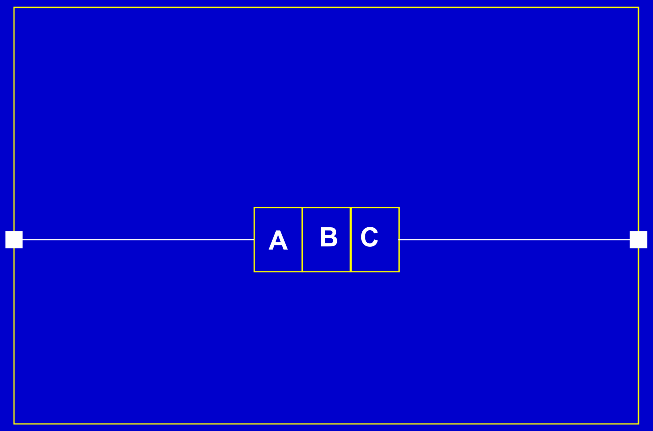
Section outline

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing optimization

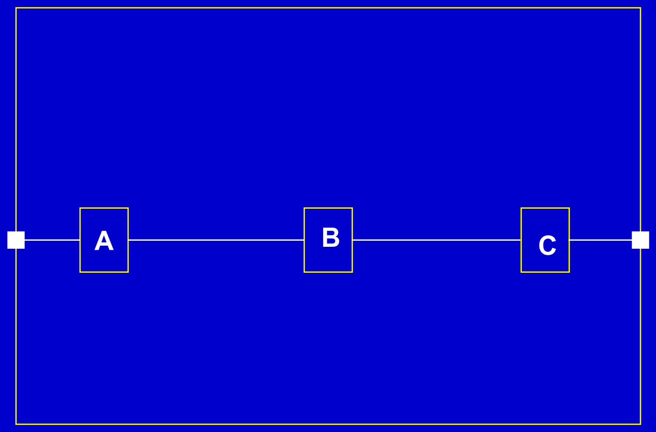
Placement Objective:

- 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 trees
    - buffer insertion
    - timing correction
- Find optimal Global Position

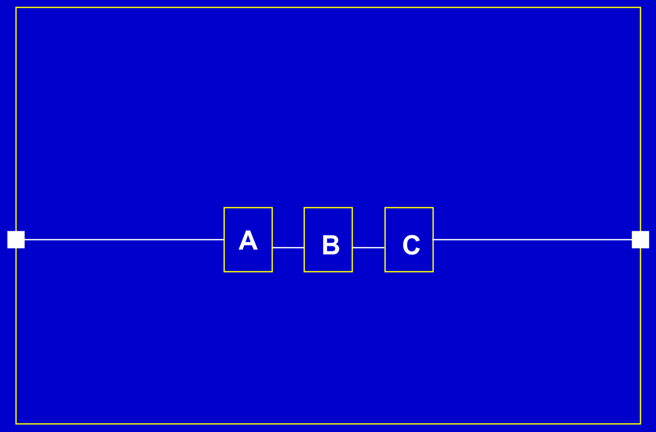
Optimal Relative Order:



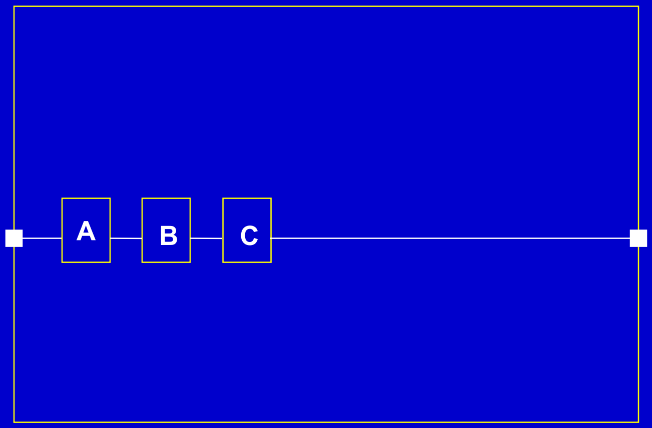
To spread ...



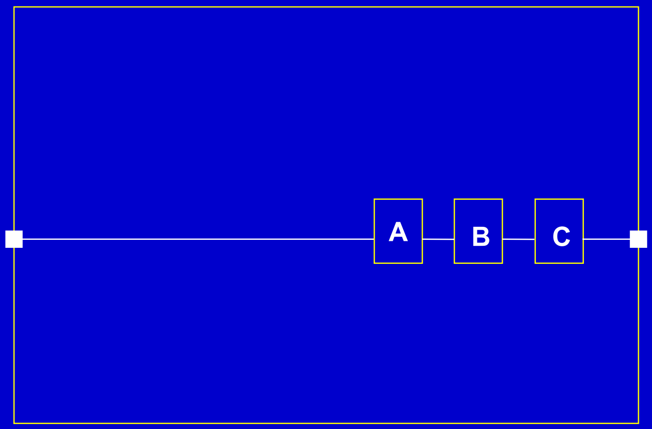
.. or not to spread

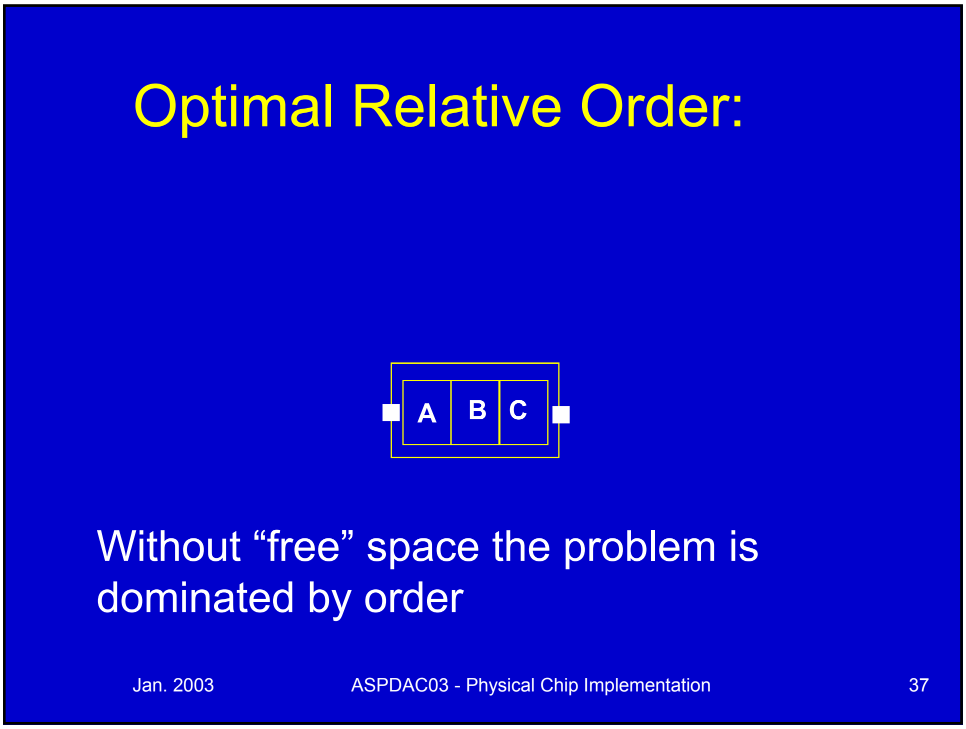


Place to the left



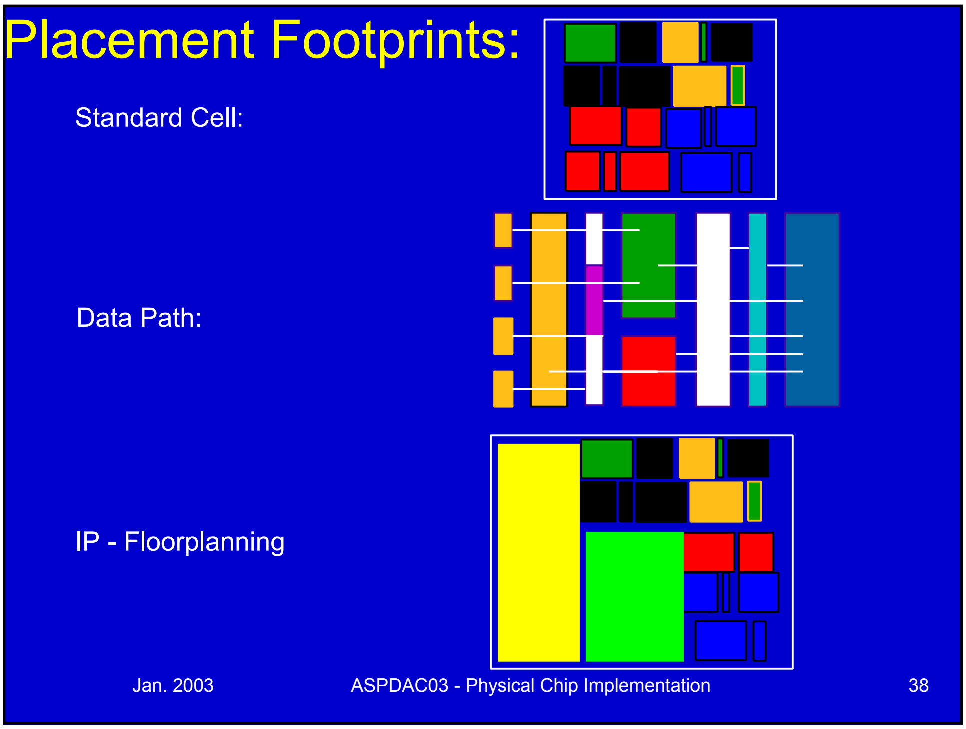
… or to the right





Optimal Relative Order:

Without “free” space the problem is dominated by order



Placement Footprints:

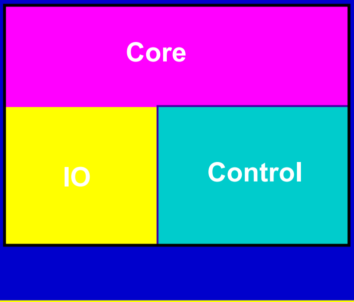
Standard Cell:

Data Path:

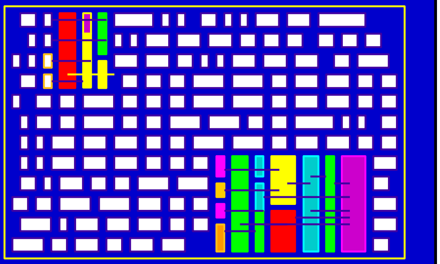
IP - Floorplanning

Placement Footprints:

Reserved areas

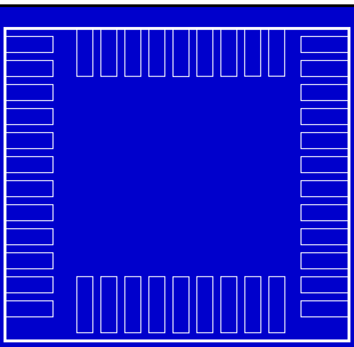


Mixed Data Path & sea of gates:

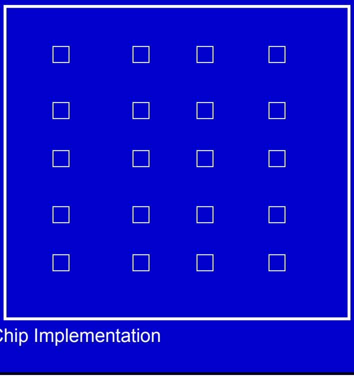


Placement Footprints:

Perimeter IO



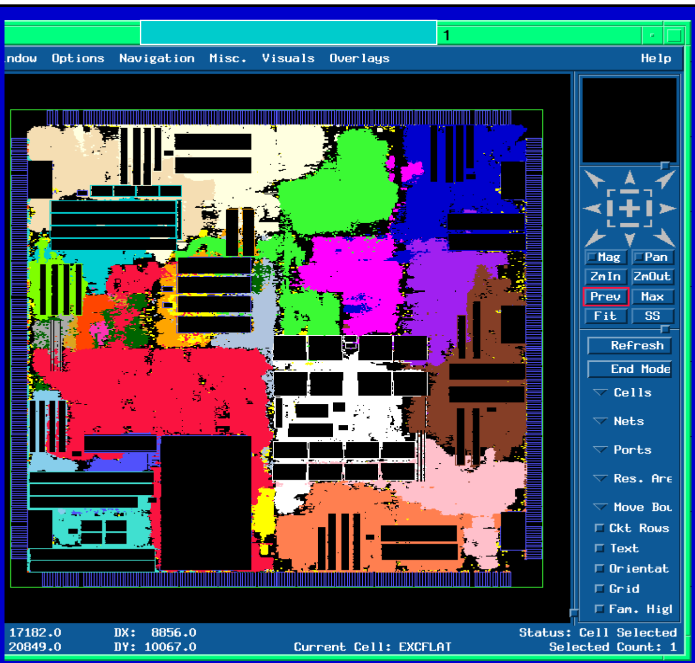
Area IO



Placement objectives are subject to User Constraints / Design Style:

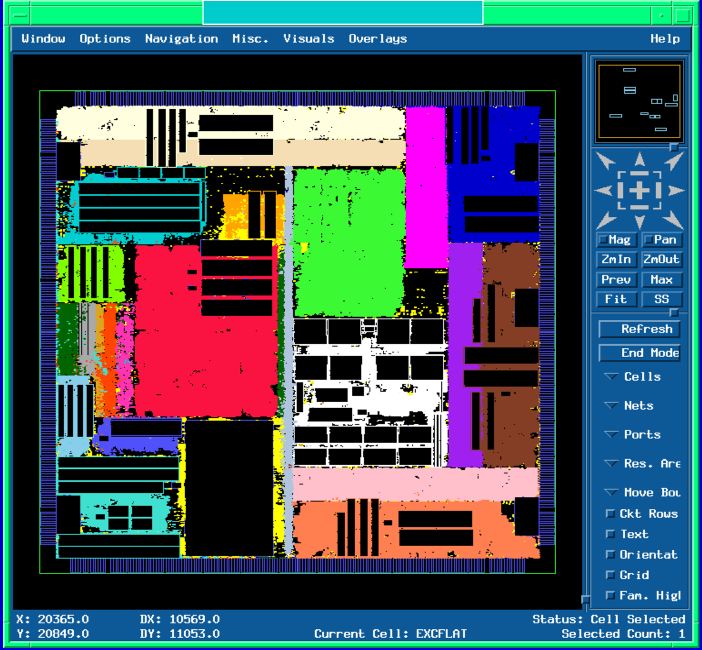
- Hierarchical Design Constraints
  - pin location
  - power rail
  - reserved layers
- Flat Design w/Floor Plan constraints
- Fixed circuits
- IO connections

Unconstrained Placement



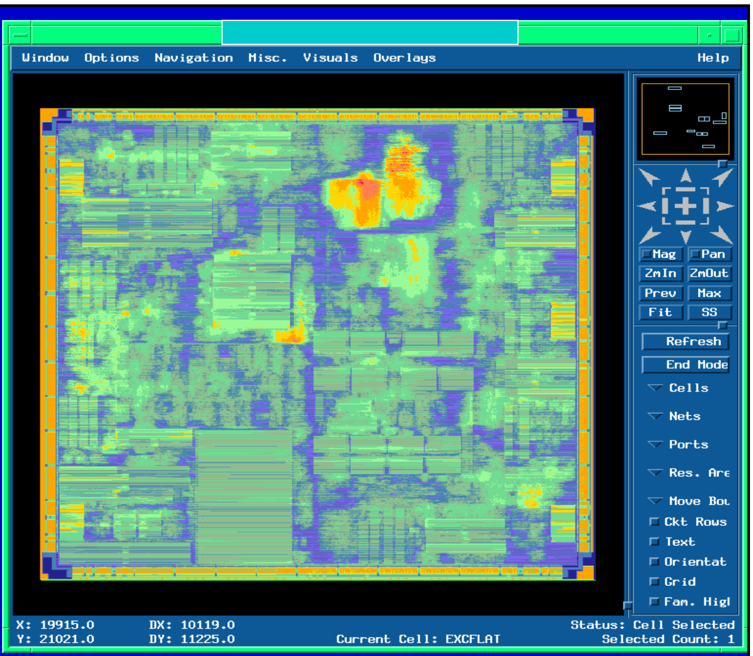
17182.0 DX: 8856.0
20849.0 DY: 10067.0 Current Cell: EXCFLAT
Status: Cell Selected
Selected Count: 1

Floor planned Placement



```
X: 20365.0      DX: 10569.0                               Status: Cell Selected
Y: 20849.0      DY: 11053.0        Current Cell: EXCFLAT     Selected Count: 1
```

Congestion MAP



```
X: 19915.0      DX: 10119.0                               Status: Cell Selected
Y: 21021.0      DY: 11225.0        Current Cell: EXCFLAT     Selected Count: 1
```

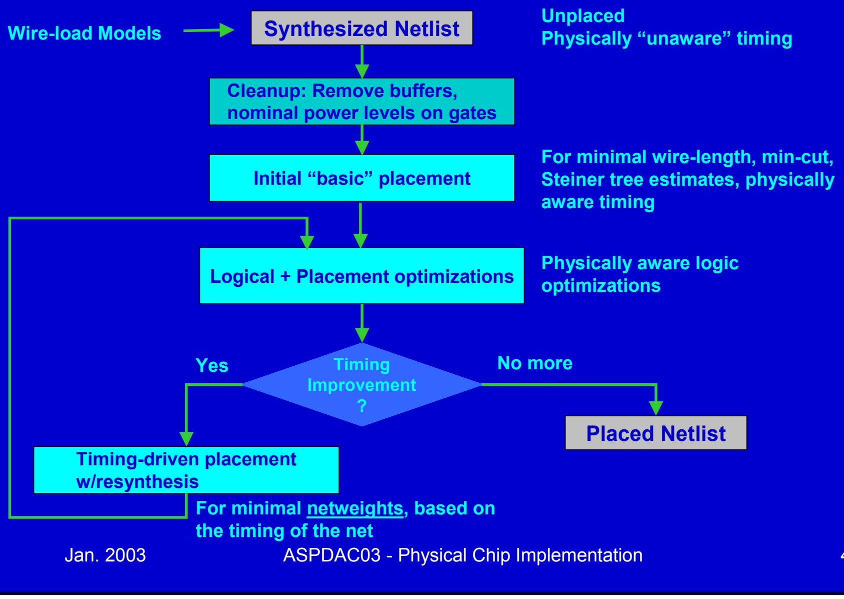
Advantages of Hierarchy

- Design is carved into smaller pieces that can be worked on in parallel (improved throughput)
- A known floor plan provides the logic design team with a large degree of placement control.
- A known floor plan provided early knowledge of long wires
- Timing closure problems can be addressed by tools, logic design, and hierarchy manipulation
- Late design changes can be done with minimal turmoil to the entire design

Disadvantages of Hierarchy

- Results depend on the quality of the hierarchy. The logic hierarchy must be designed with PD taken into account.
- Additional methodology requirements must be met to enable hierarchy. Ex. Pin assignment, Macro Abstract management, area budgeting, floor planning, timing budgets, etc
- Late design changes may affect multiple components.
- Hierarchy allows divergent methodologies
- Hierarchy hinders DA algorithms. They can no longer perform global optimizations.

Physical Synthesis Flow



Wire-load Models

Synthesized Netlist

Unplaced
Physically "unaware" timing

Cleanup: Remove buffers, nominal power levels on gates

Initial "basic" placement

For minimal wire-length, min-cut, Steiner tree estimates, physically aware timing

Logical + Placement optimizations

Physically aware logic optimizations

Timing Improvement ?

Yes

No more

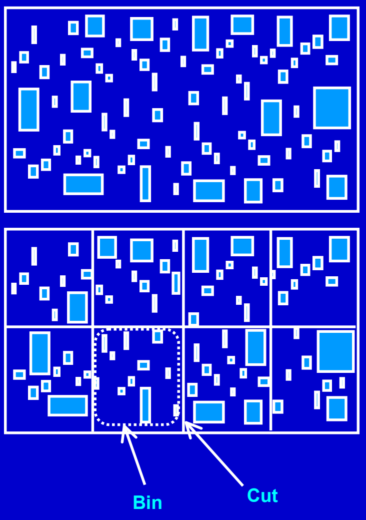
Placed Netlist

Timing-driven placement w/resynthesis

For minimal netweights, based on the timing of the net

Example of Logical + Placement Optimizations

- Start with a placed or unplaced netlist
- Do recursive partitioning
- During and following each partition action, apply logic optimizations such as
  - timing corrections
  - rebuffering
  - repowering
  - cloning
  - pin swapping
  - move boxes
  - … etc

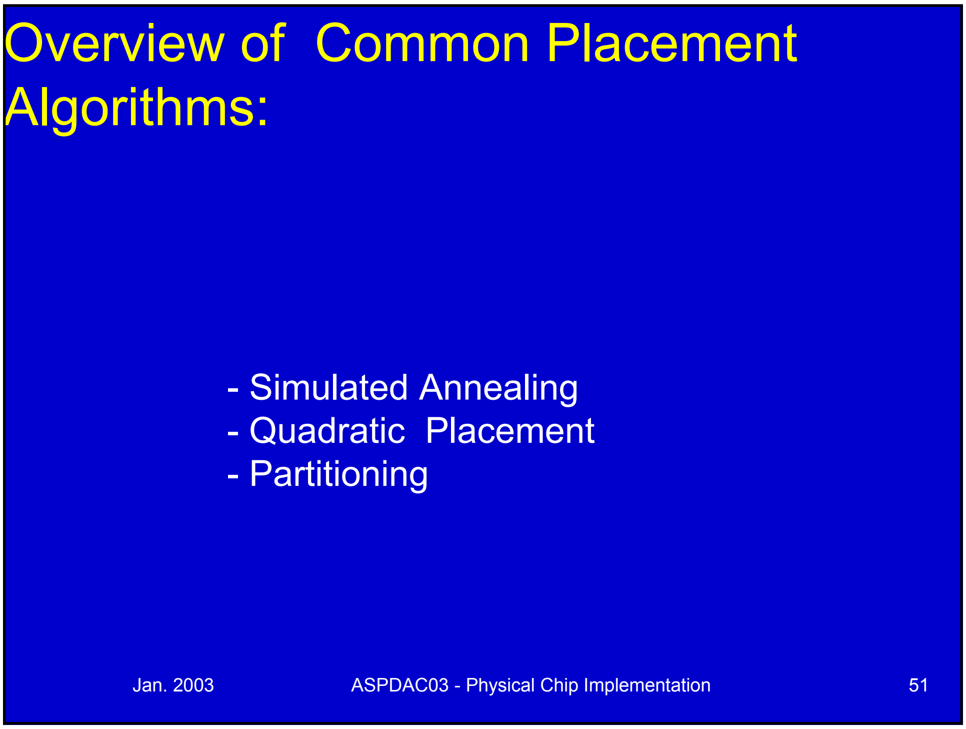


Summary of Placement Methods

- Simulated annealing
  - (+) High-quality, arbitrary objectives and constraints, parallelizable, easy to implement
  - (-) Doesn’t scale
- Quadratic (or, “analytic”)
  - (+) Mathematically clean, fast (ConjGrad) solvers
  - (-) Solving “the wrong problem”, highly illegal solutions must be legalized, fixed “anchors” needed
  - Example: Alpert, Nam, Villarubia QUAD+ACG placer (ICCAD-02)
- Partitioning-based
  - (+) Fastest, scales well if multilevel used, good quality
  - (-) Must be heavily tuned (hMetis, MLPart), difficult to constrain, unstable results (same quality but different structure) (?)
  - Example: Capo (http://gigascale.org/bookshelf/)

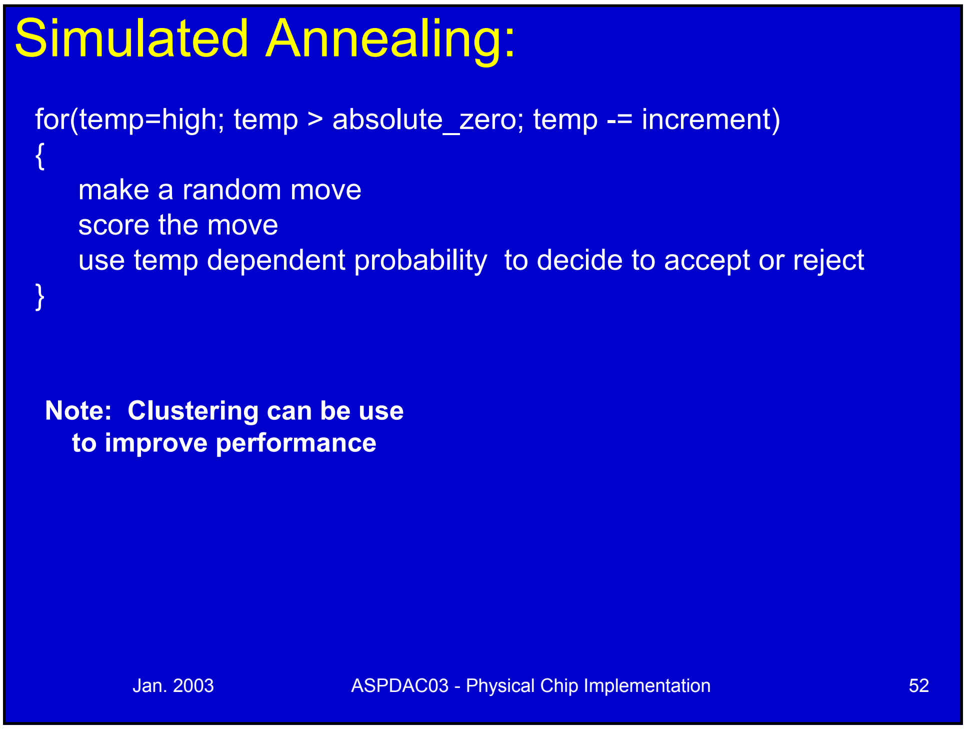
Section Outline

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing optimization



Overview of Common Placement Algorithms:

- Simulated Annealing
- Quadratic Placement
- Partitioning



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 use to improve performance

Annealing:

Pros:

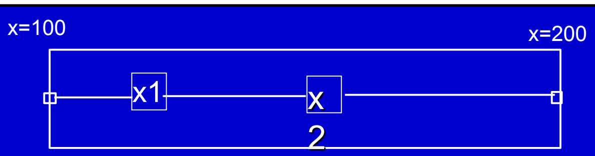
- ease of implementation, dumb moves / smart scoring
- can easily accommodate new constraints - just add them to the scoring function
- great quality
- can be made to run on parallel processors

Cons:

- very long run time

Quadratic Placement

Review:



$$Cost = (x_1 - 100)^2 + (x_1 - x_2)^2 + (x_2 - 200)^2$$

$$\frac{\partial}{\partial x_1} Cost = 2(x_1 - 100) + 2(x_1 - x_2)$$

$$\frac{\partial}{\partial x_2} Cost = -2(x_1 - x_2) + 2(x_2 - 200)$$

setting the partial derivatives = 0 we solve for the minimum Cost:

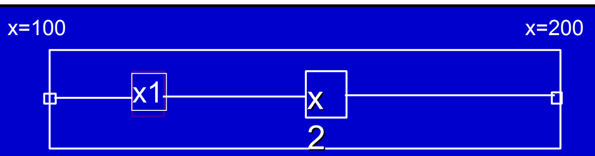
Ax + B = 0

$$\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -200 \\ -400 \end{bmatrix} = 0$$

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -100 \\ -200 \end{bmatrix} = 0$$

x1=400/3   x2=500/3

Review:



setting the partial derivatives = 0 we solve for the minimum Cost:

Ax + B = 0

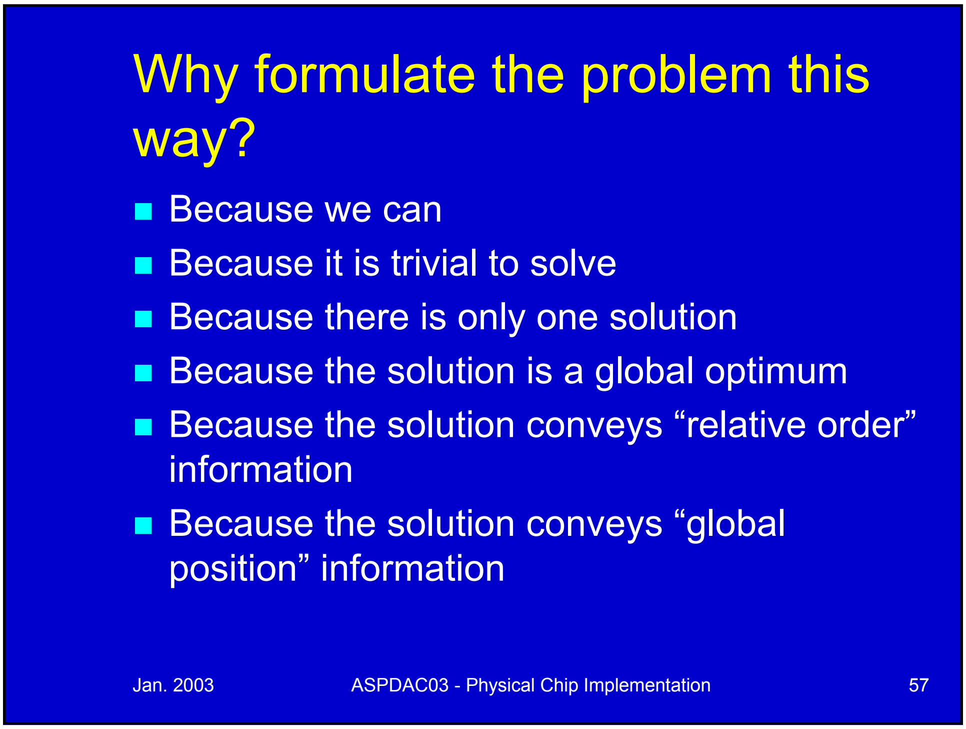
$$\begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -200 \\ -400 \end{bmatrix} = 0$$

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -100 \\ -200 \end{bmatrix} = 0$$

x1=400/3   x2=500/3

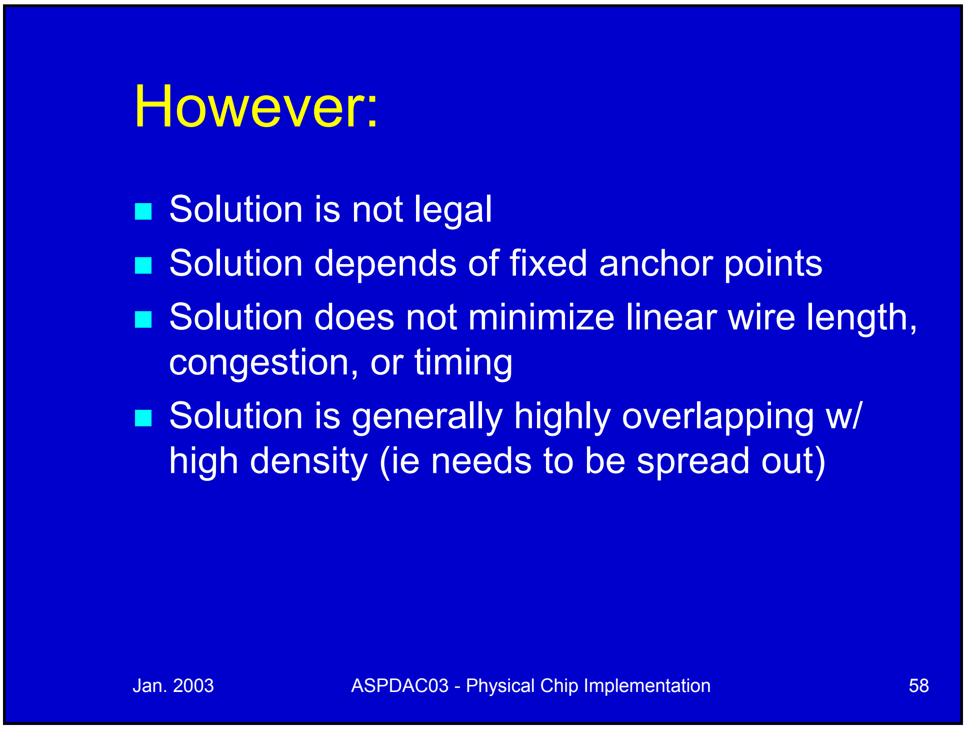
Interpretation of matrices A and B:

The diagonal values A[i,i] correspond to the number of connections to xi
The off diagonal values A[i,j] are 1 if object i is connected to object j, 0 otherwise
The values B[i] correspond to the sum of the locations of fixed objects connected to object i



Why formulate the problem this way?

- Because we can
- Because it is trivial to solve
- Because there is only one solution
- Because the solution is a global optimum
- Because the solution conveys “relative order” information
- Because the solution conveys “global position” information



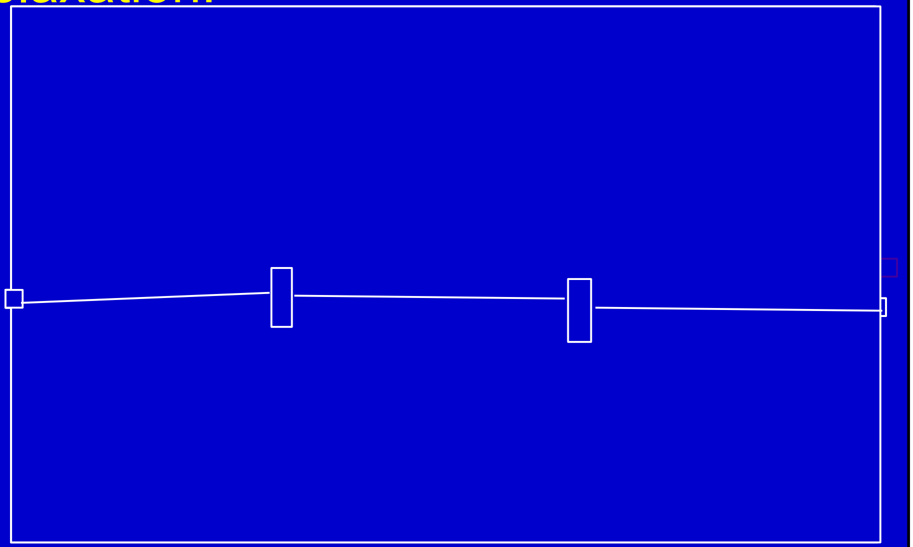
However:

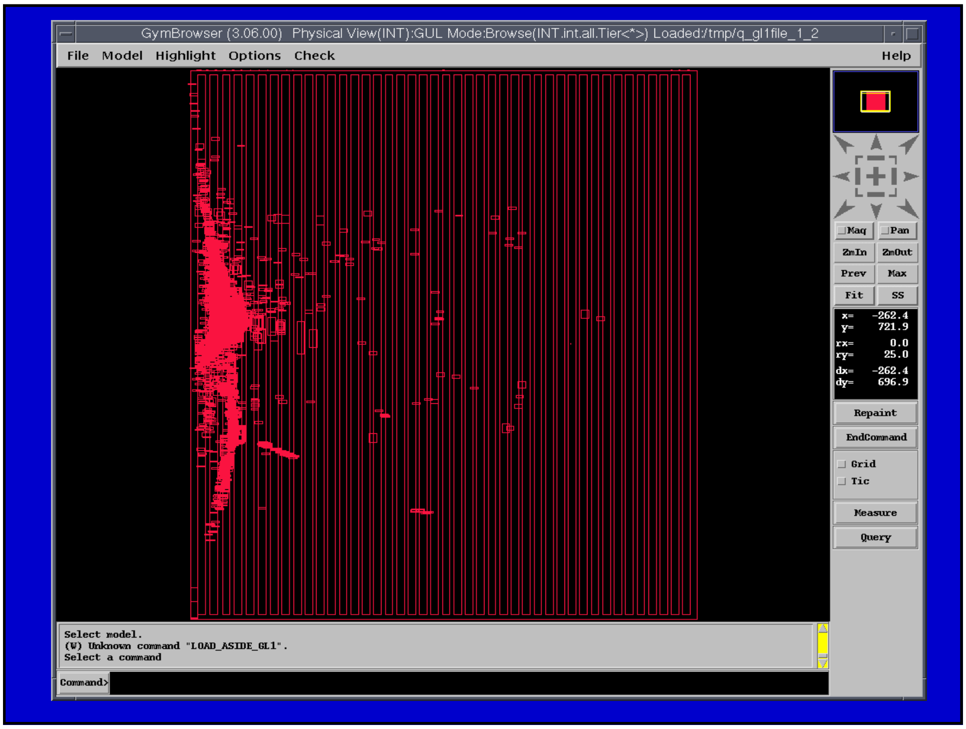
- Solution is not legal
- Solution depends of fixed anchor points
- Solution does not minimize linear wire length, congestion, or timing
- Solution is generally highly overlapping w/ high density (ie needs to be spread out)

What does the solution look like?

- To get an intuitive feel for the solution, examine the relaxation method for solving Ax + B = 0
- Actual program implementation may use other solution methods (that are generally less intuitive).

Solution of Quadratic using Relaxation:

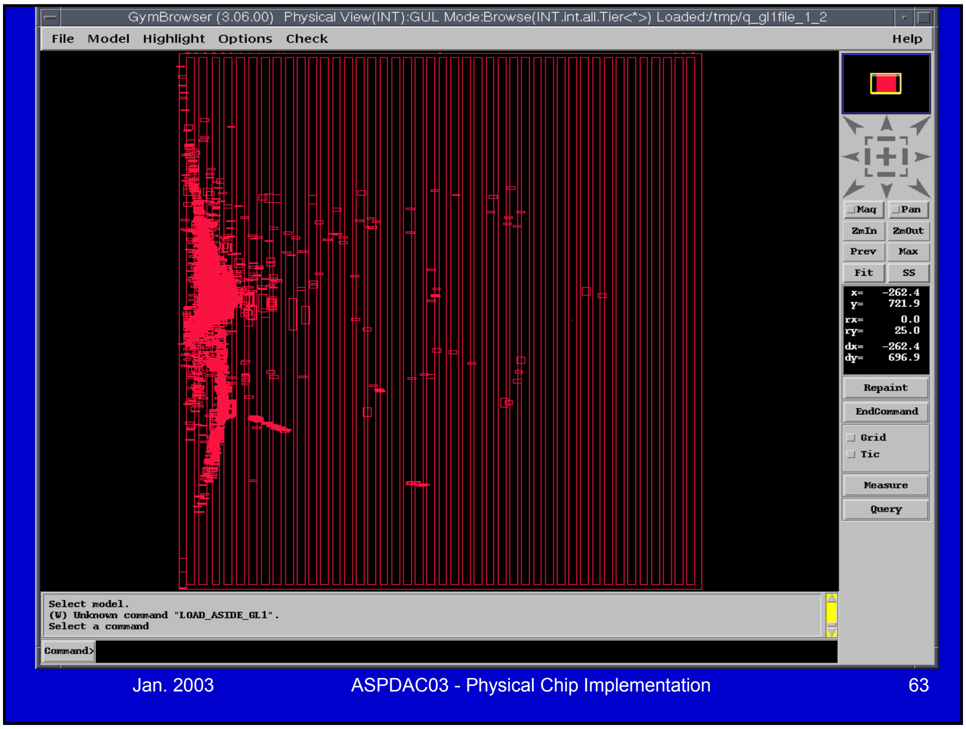




```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
Command>
```

Constrained Solutions:

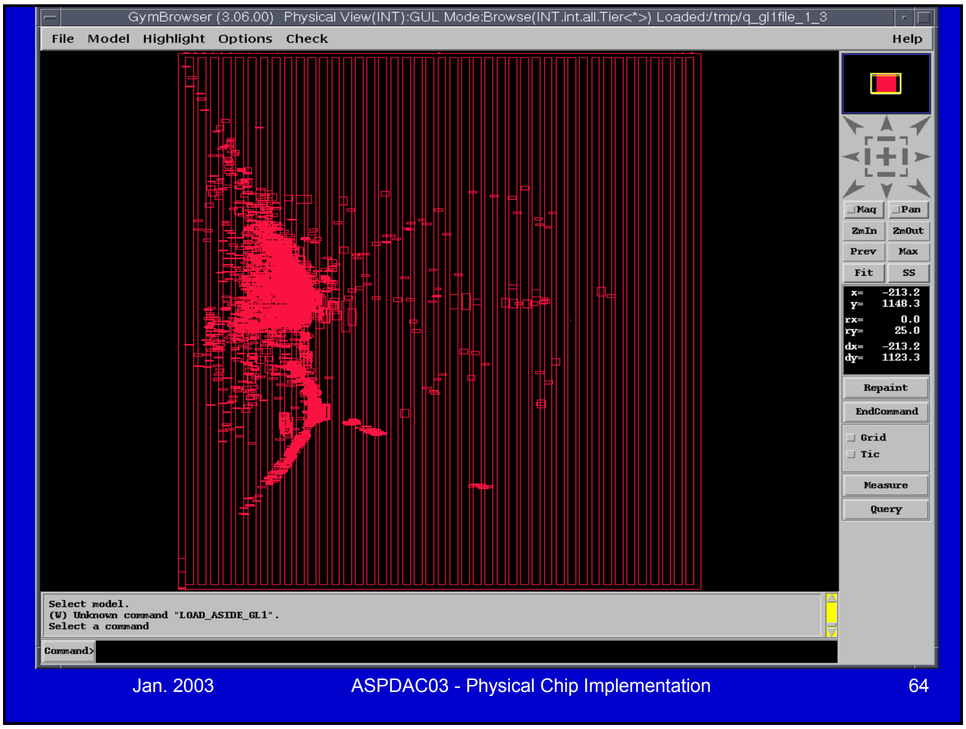
- Sometimes we want to solve for the minimum wire
- length subject to a constraint
- Example: Using quadratic for partitioning, we may want the quadratic placement to be "centered"


GymBrowser (3.06.00) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/tmp/q_gl1file_1_2
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
□ Grid
□ Tic
Measure
Query

```
x=   -262.4
y=    721.9
rx=     0.0
ry=    25.0
dx=  -262.4
dy=   696.9
```

```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
```

```
Command>
```


GymBrowser (3.06.00) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/tmp/q_gl1file_1_3
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
□ Grid
□ Tic
Measure
Query

```
x=   -213.2
y=   1148.3
rx=     0.0
ry=    25.0
dx=  -213.2
dy=  1123.3
```

```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
```

```
Command>
```

Constrained Solutions

To minimize Cost = f(x) subject to a constraint g(x) = 0 we can use langrangian multipliers to modify the Cost function as follows:

$$Cost = f(x) + \lambda g(x)$$

$$\frac{\partial}{\partial x} Cost = \frac{\partial}{\partial x} f(x) + \lambda \frac{\partial}{\partial x} g(x)$$

Using CG as a constraint:

$CG = \sum_{i=1}^{n} s_i x_i / \sum_{i=1}^{n} s_i$ where: s is the size of object_i
n is the number of objects

$g(x) = (\sum_{i=1}^{n} s_i x_i / \sum_{i=1}^{n} s_i) - CG$

$\frac{\partial}{\partial x} g(x) = \frac{s_i}{N}$ where we use N to represent the constant $\sum_{i=1}^{n} s_i$

We have already shown that

$\frac{\partial}{\partial x} f(x) = 0$ leads to the system of equations --- Ax + B = 0

Therefore solving the constrained porblem $\frac{\partial}{\partial x} Cost = \frac{\partial}{\partial x} f(x) + \lambda \frac{\partial}{\partial x} g(x) = 0$

leads to: $Ax + B + \lambda \left[\frac{s_i}{N} \right] = 0$

Constrained Solutions (cont):

To solve Ax + B + $\lambda \left[\frac{s_i}{N} \right] = 0$ we could use a packaged solver and add the additional unknown λ and equation $CG = \sum_{i=1}^{n} s_i x_i / N$ to our matricies and solve.

Here is an alternative way to solve the system:

by substitution we let $x = x_u + \lambda x_l$
where x_u is the unconstrained solution (ie the solution to Ax + B = 0)

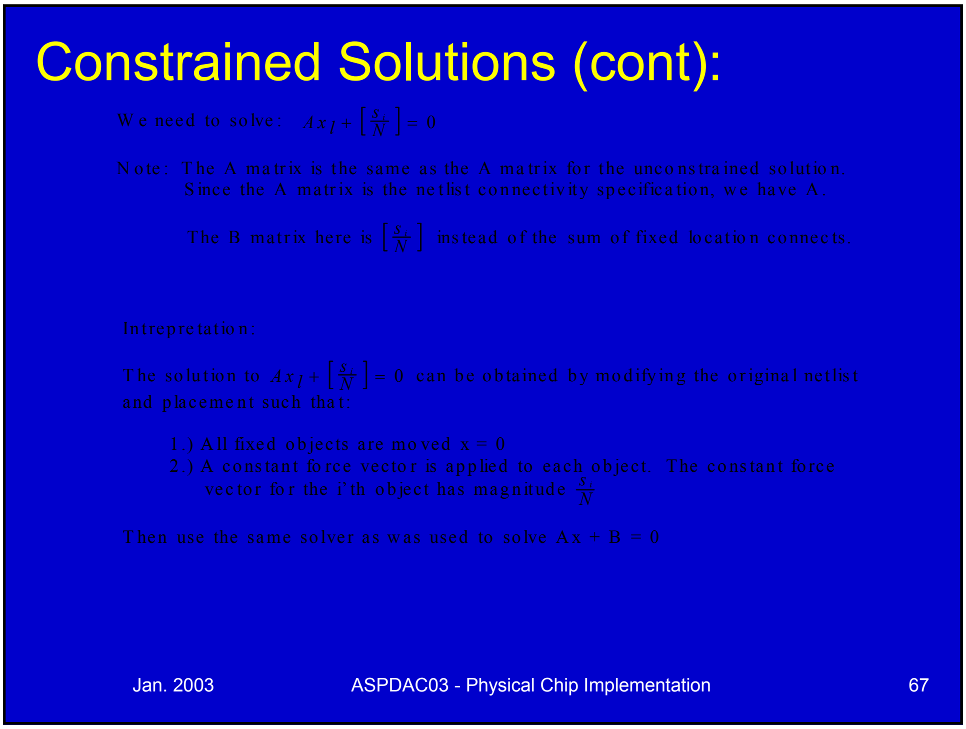
Assuming we can solve the unconstrained problem, x_u is known.

By substitution we get:

$$A(x_u + \lambda x_l) + B + \lambda \left[\frac{s_i}{N} \right] = 0$$

which becomes:

$A \lambda x_l + \lambda \left[\frac{s_i}{N} \right] = 0$ or $A x_l + \left[\frac{s_i}{N} \right] = 0$



Constrained Solutions (cont):

We need to solve: $Ax_l + \left[\frac{s_i}{N}\right] = 0$

Note: The A matrix is the same as the A matrix for the unconstrained solution.
Since the A matrix is the netlist connectivity specification, we have A.

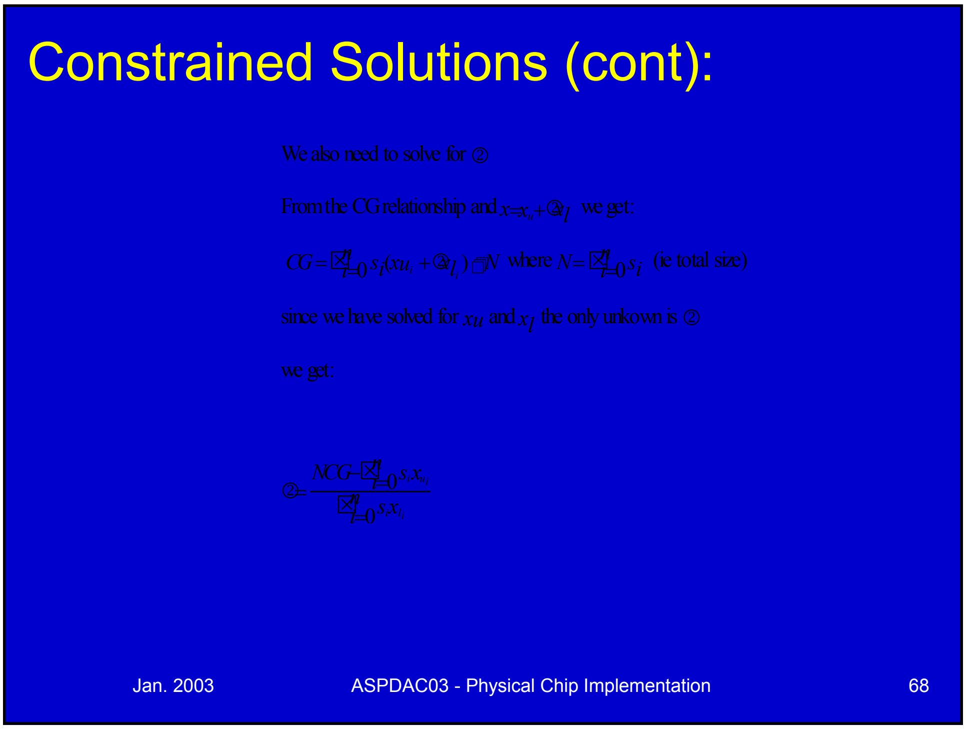
The B matrix here is $\left[\frac{s_i}{N}\right]$ instead of the sum of fixed location connects.

Intrepretation:

The solution to $Ax_l + \left[\frac{s_i}{N}\right] = 0$ can be obtained by modifying the original netlist and placement such that:

1.) All fixed objects are moved x = 0.
2.) A constant force vector is applied to each object. The constant force vector for the i'th object has magnitude $\frac{s_i}{N}$

Then use the same solver as was used to solve Ax + B = 0



Constrained Solutions (cont):

We also need to solve for ②

From the CG relationship and $x = x_u + ② x_l$ we get:

$CG = \boxtimes_{i=0}^{n} s_i(xu_i + ② x_{l_i}) □ N$ where $N = \boxtimes_{i=0}^{n} s_i$ (ie total size)

since we have solved for xu and x_l the only unkown is ②

we get:

$$② = \frac{NCG - \boxtimes_{i=0}^{n} s_i x_{u_i}}{\boxtimes_{i=0}^{n} s_i x_{l_i}}$$

Constrained Solutions (summary):

To minimize f(x) (the wl squared cost function) subject to a CG constraint we do the following:

1.) Solve for x_u by solving $Ax_u + B = 0$ using relaxation or some other method

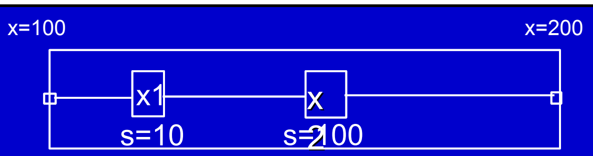
2.) Solve for x_l as follows: $Ax_l + \left[\frac{s_i}{N} \right] = 0$

- Move all fixed objects to location=0
- Add a constant force vector to each object. The constant force vector for the i'th object has magnitude $\frac{s_i}{N}$
- Using relaxation or some other method, solve for x_l

3.) Solve for ② using $② = \frac{NCG - ⊠_{i=0}^{n} s_i x_{u_i}}{⊠_{i=0}^{n} s_i x_{l_i}}$

4.) Compute the final placement using $x = x_u + ② x_l$

Review:



Force CG to 150

From the previous example we know that the solution to:

$Ax_u + B = 0$

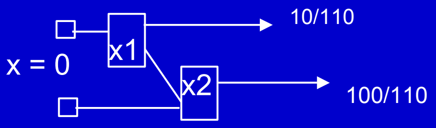
$x = \begin{bmatrix} 133.33 \\ 166.67 \end{bmatrix}$

with this solution the CG is at $\frac{(10⋊133.33)+(100⋊166.67)}{110}$ = 163.64 (ie not 150)

Now we need to solve:

$Ax_l + \frac{s_i}{N} = 0$ which is the same as solving -> $Ax_l + \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{s_i}{N} = 0$

Recall that the B matrix represents the position of fixed objects. So, this equation represents the solution to:



Constrained Solutions (summary):

Advantages of this approach:

1.) The Solver data structure is the netlist only
ie. no additional memory requirements

2.) Sometimes the unconstrained solution is by itself sufficient, therefore we can avoid the additional overhead of producing the constrained solution

3.) The numerical iterations in this method are NOT dependent on the CG. We can solve for xu and xl, then try many different CG points at very low cost.

Quadratic Techniques:

Pros:
- mathematically well behaved
- efficient solution techniques find global optimum
- great quality

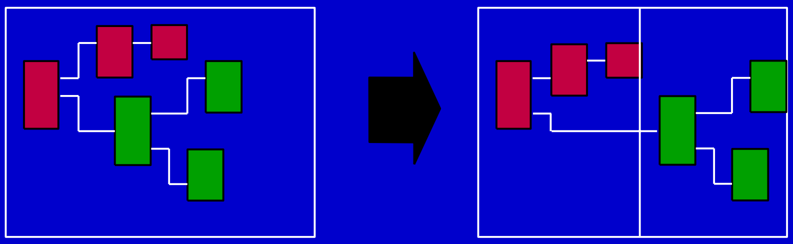
Cons:
- solution of Ax + B = 0 is not a legal placement, so generally some additional partitioning techniques are required.
- solution of Ax + B = 0 is that of the "mapped" problem, ie nets are represented as cliques, and the solution minimizes wire length squared, not linear wire length unless additional methods are deployed
- fixed IOs are required for these techniques to work well

Partitioning

Partitioning:

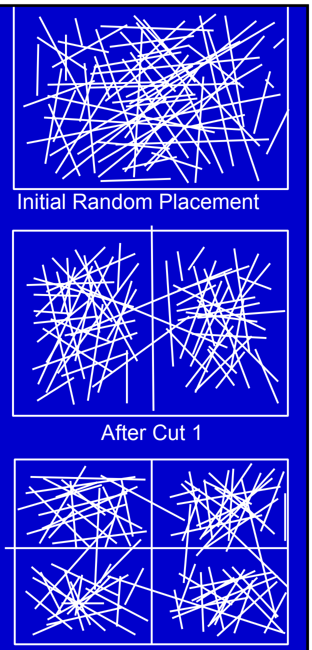
Objective:

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



FM Partitioning:

```
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

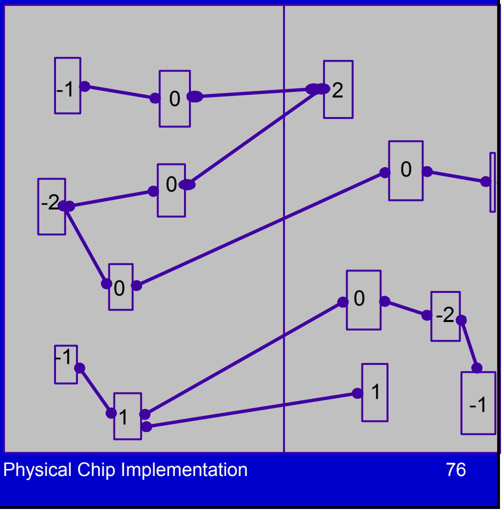
After Cut 2

FM Partitioning:

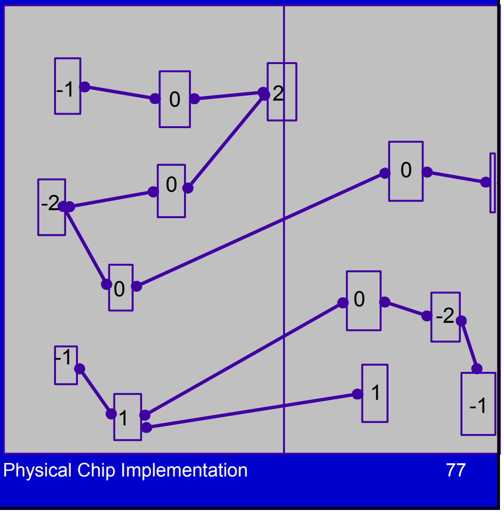
Moves are made based on object gain

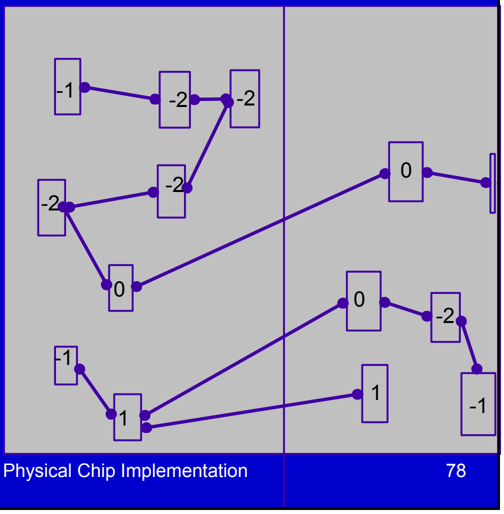
Object Gain: The amount of change in cut crossings that will occur if an object is moved from its current partition into the other partition

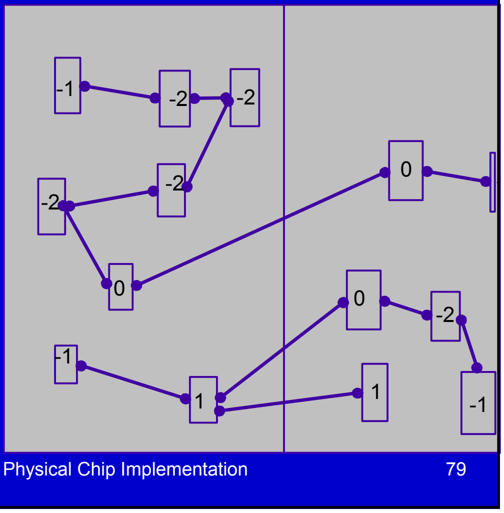
- each object is assigned a gain
- objects are put into a sorted gain list
- the object with the highest gain from the smaller of the two sides is selected and moved.
- the moved object is "locked"
- gains of "touched" objects are recomputed
- gain lists are resorted

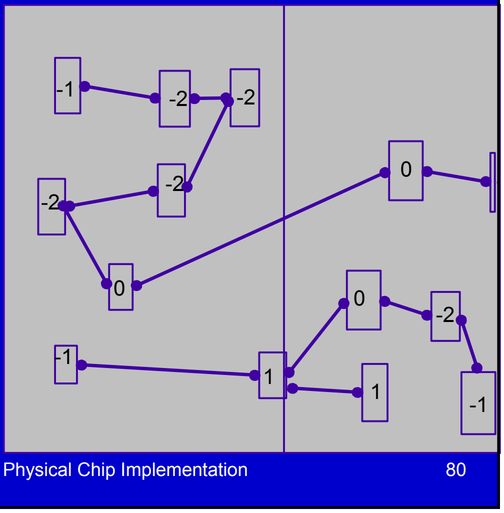


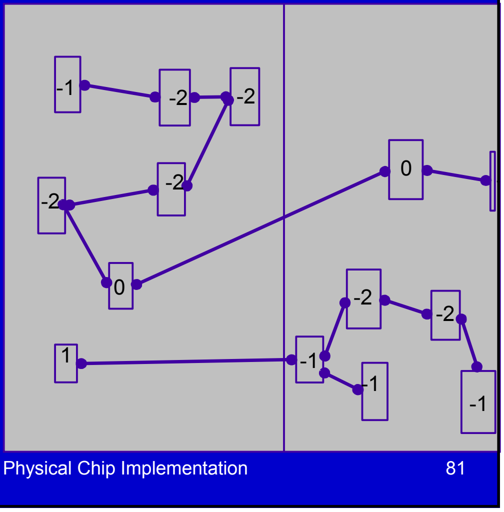
FM Partitioning:

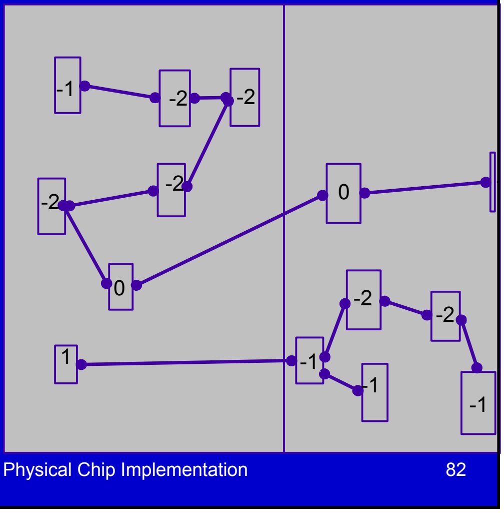


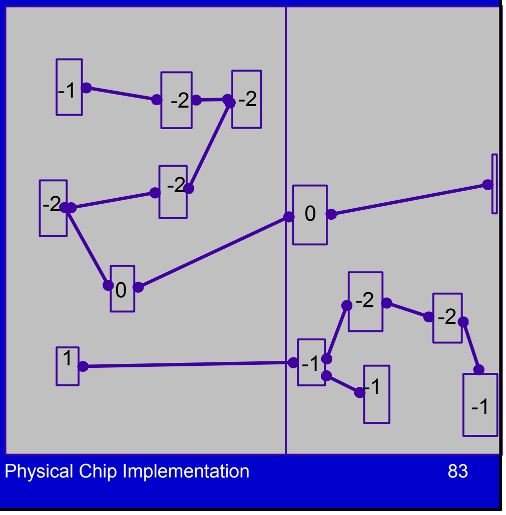


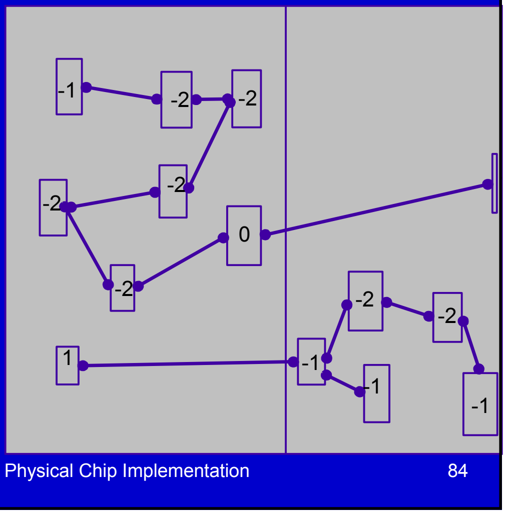

-1
-2
-2
-2
-2
0
0
0
-2
-1
1
1
-1

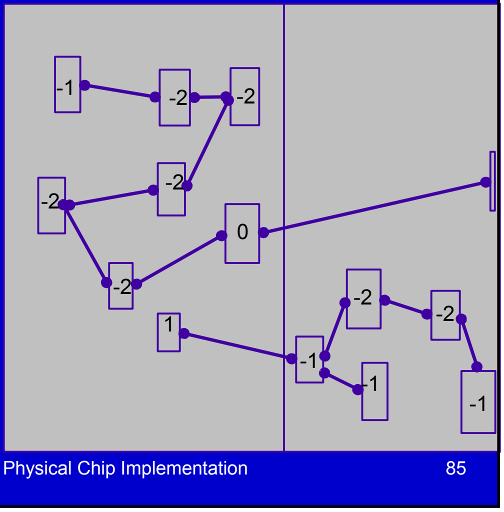

-1
-2
-2
-2
-2
0
0
0
-2
-1
1
1
-1


-1
-2
-2
0
-2
-2
0
-2
-2
1
-1
-1
-1

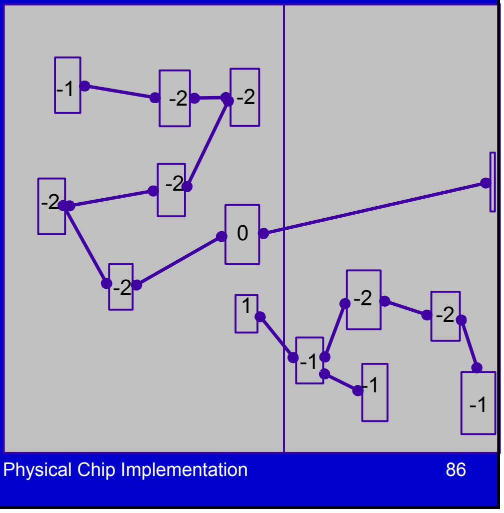

-1
-2
-2
-2
-2
0
0
-2
-2
1
-1
-1
-1


-1
-2
-2
-2
-2
0
0
-2
-2
1
-1
-1
-1

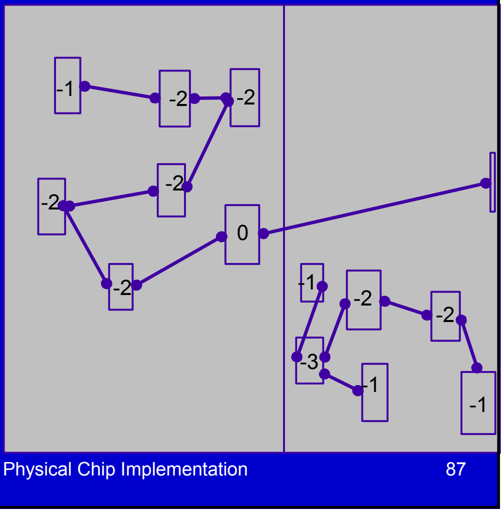

-1
-2
-2
-2
-2
0
-2
-2
-2
1
-1
-1
-1

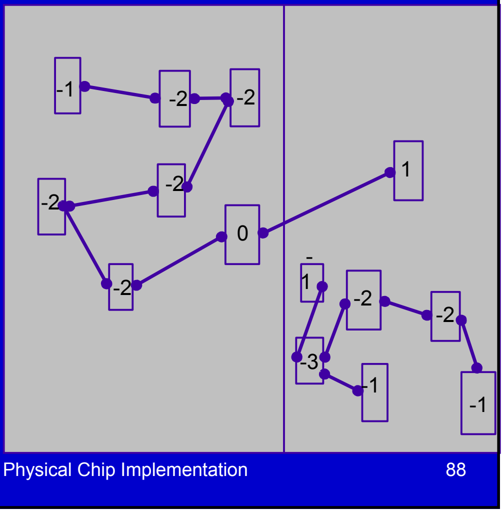


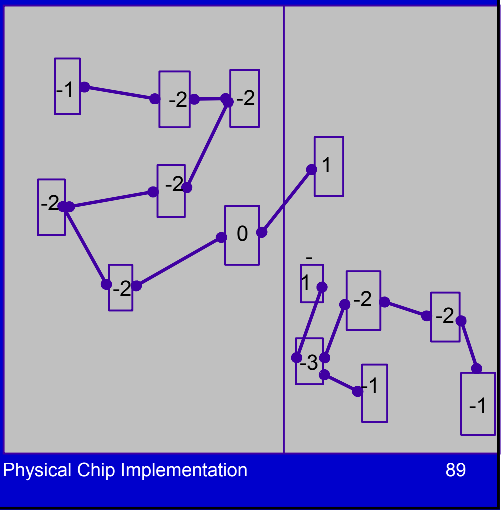
-1
-2
-2
-2
-2
0
-2
1
-2
-2
-1
-1
-1

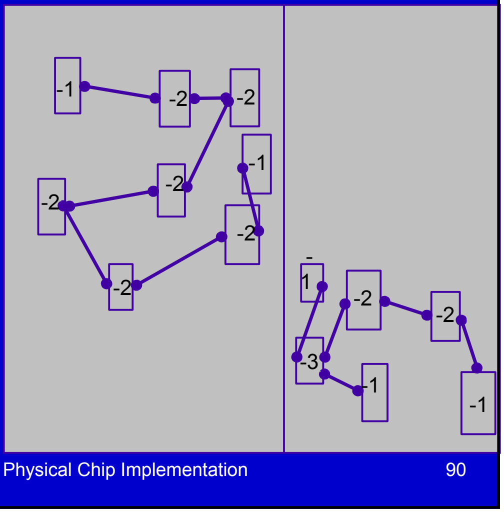


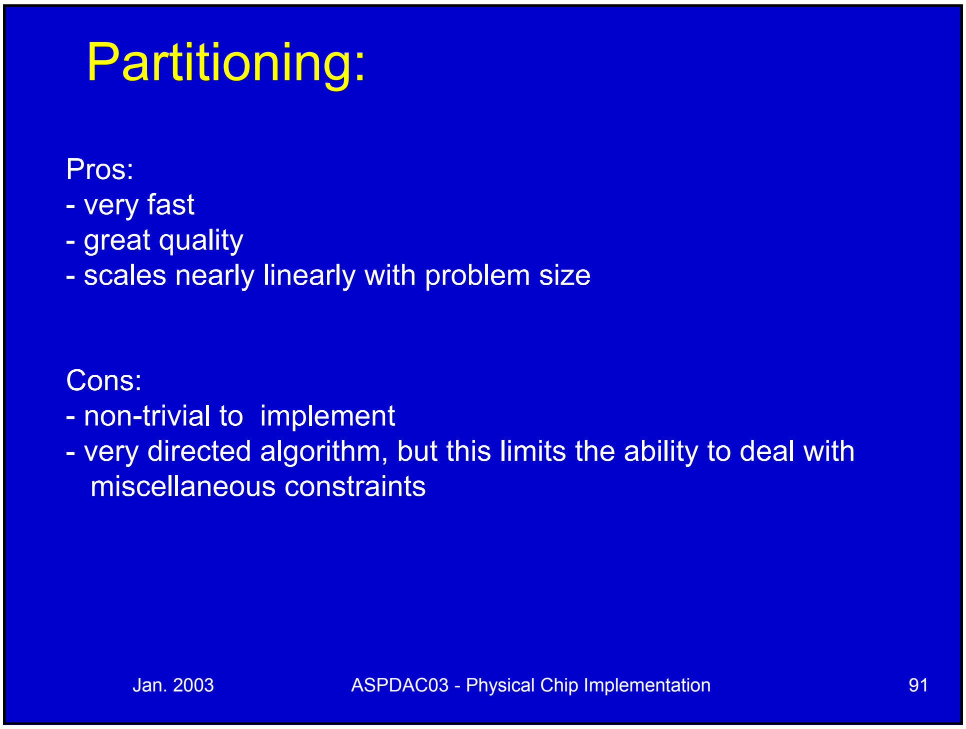
-1
-2
-2
-2
-2
0
-2
1
-2
-2
-1
-1
-1


-1
-2
-2
-2
-2
-2
0
-1
-2
-2
-3
-1
-1


-1
-2
-2
-2
-2
-2
0
1
-
1
-2
-2
-3
-1
-1


-1
-2
-2
1
-2
-2
0
-2
-
1
-2
-2
-3
-1
-1

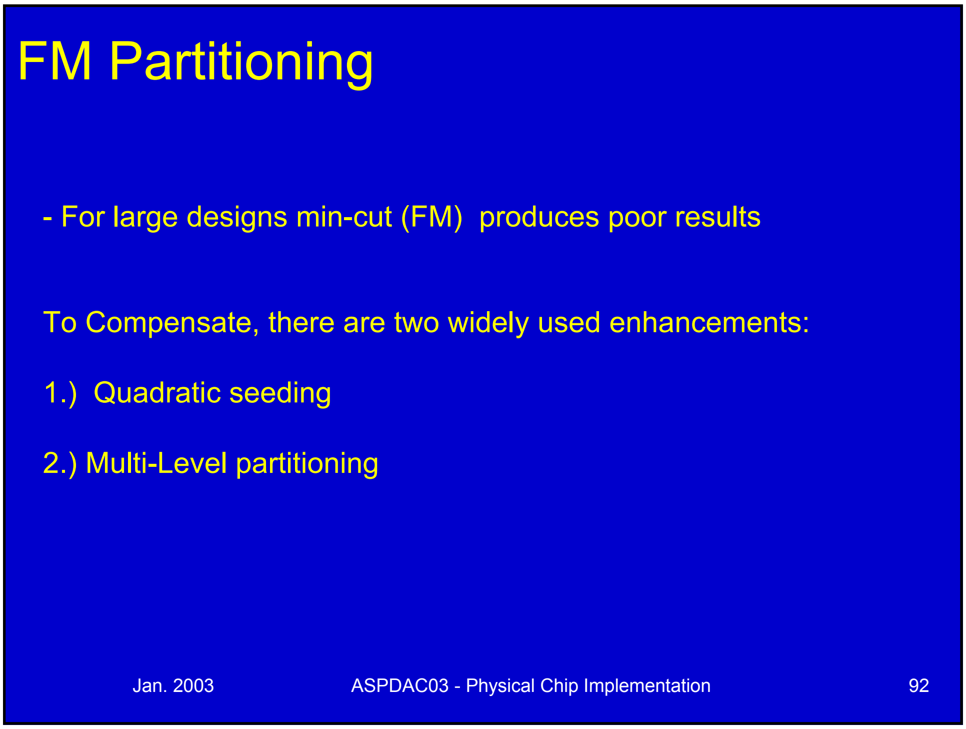

-1
-2
-2
-1
-2
-2
-2
-2
-
1
-2
-2
-3
-1
-1



Partitioning:

Pros:
- very fast
- great quality
- scales nearly linearly with problem size

Cons:
- non-trivial to implement
- very directed algorithm, but this limits the ability to deal with miscellaneous constraints



FM Partitioning

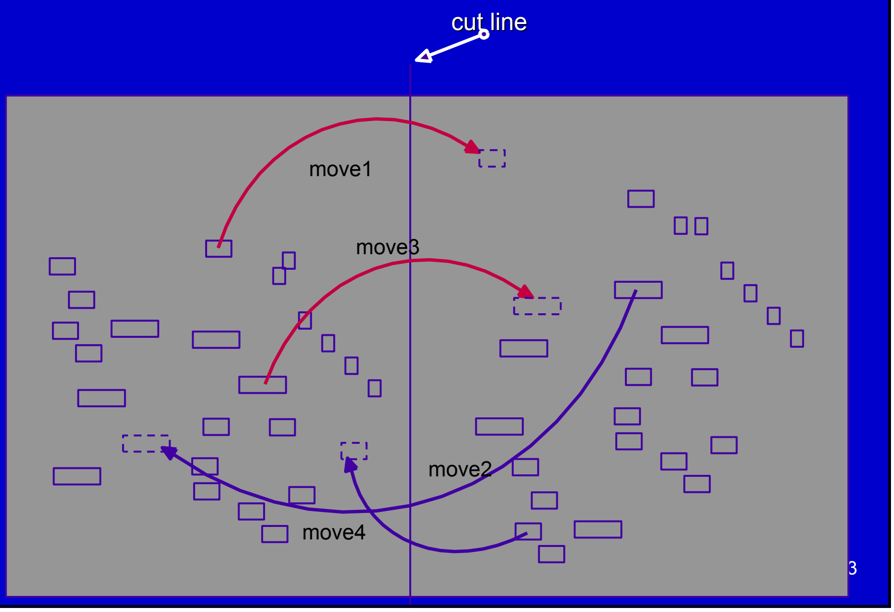
- For large designs min-cut (FM) produces poor results

To Compensate, there are two widely used enhancements:

1.) Quadratic seeding

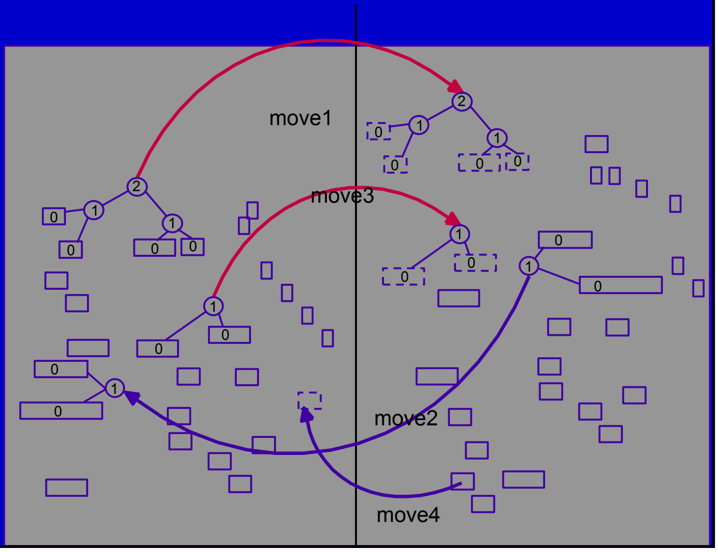
2.) Multi-Level partitioning

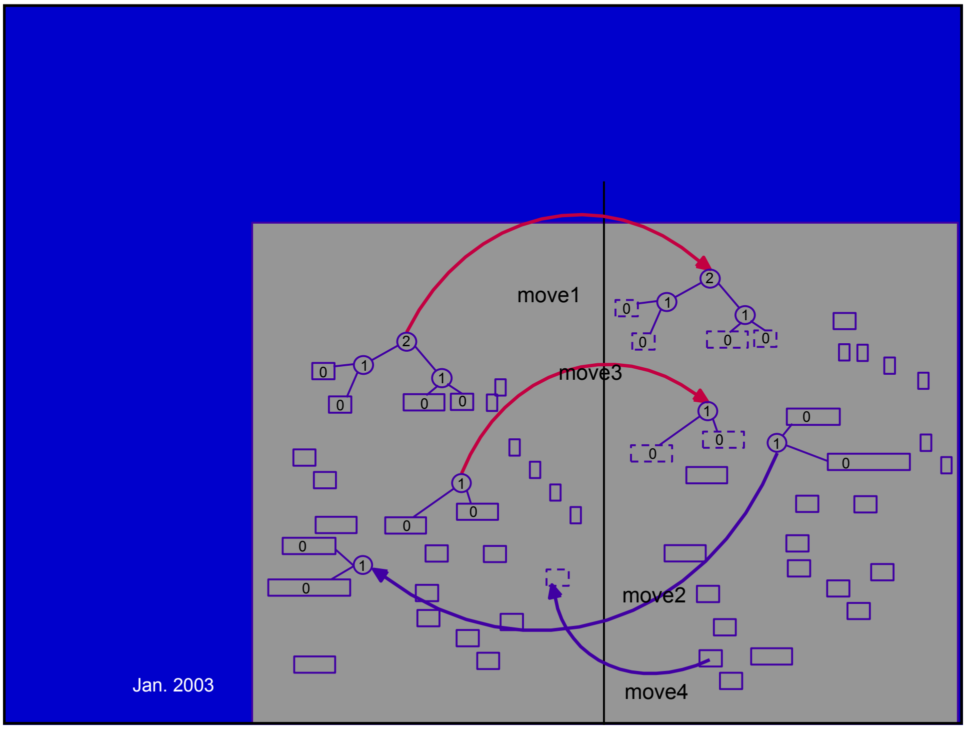
Partitioning:



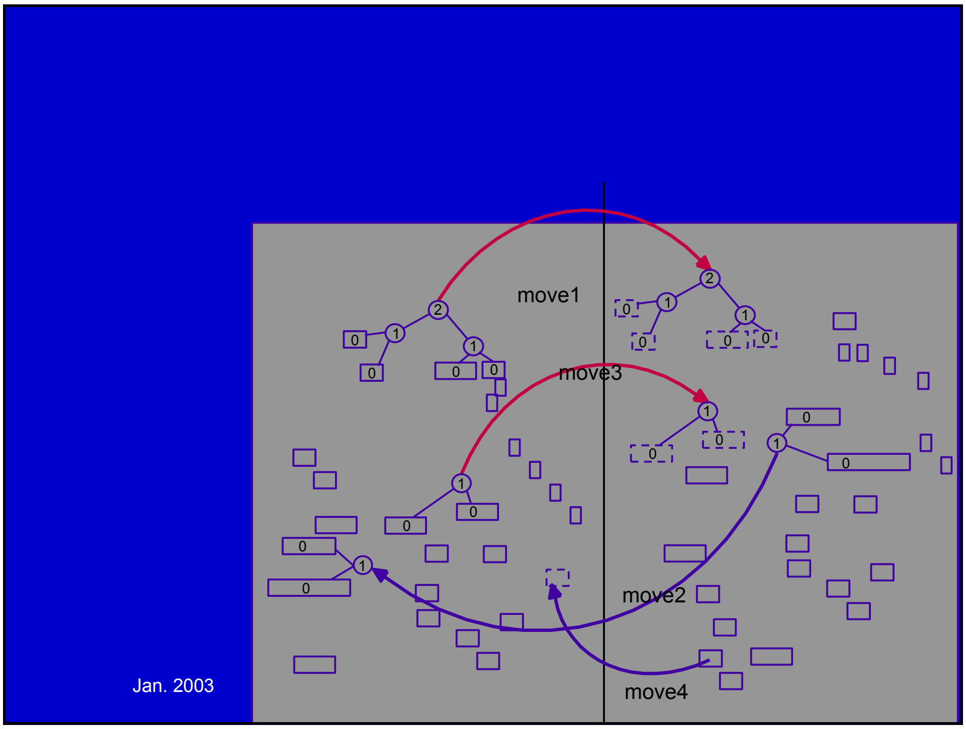
Global Placement - Multi-Level Partitioning:

```
generate clusters:
while(there are clusters)
 {
    partition_it;
    remove 1 cluster layer;
 }
partition_it;
```

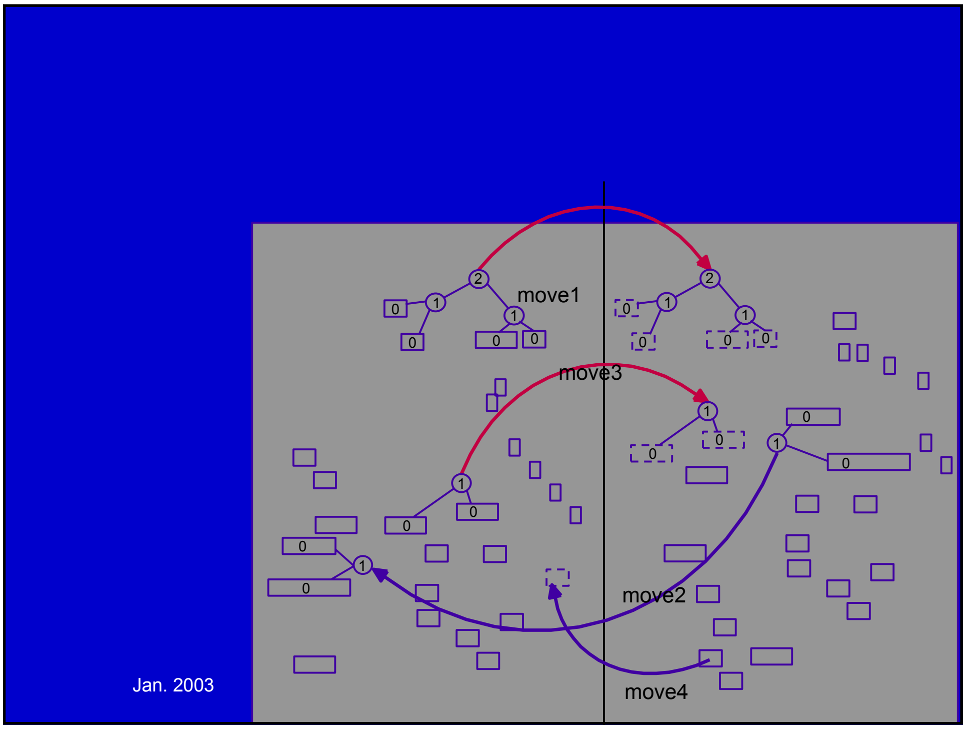




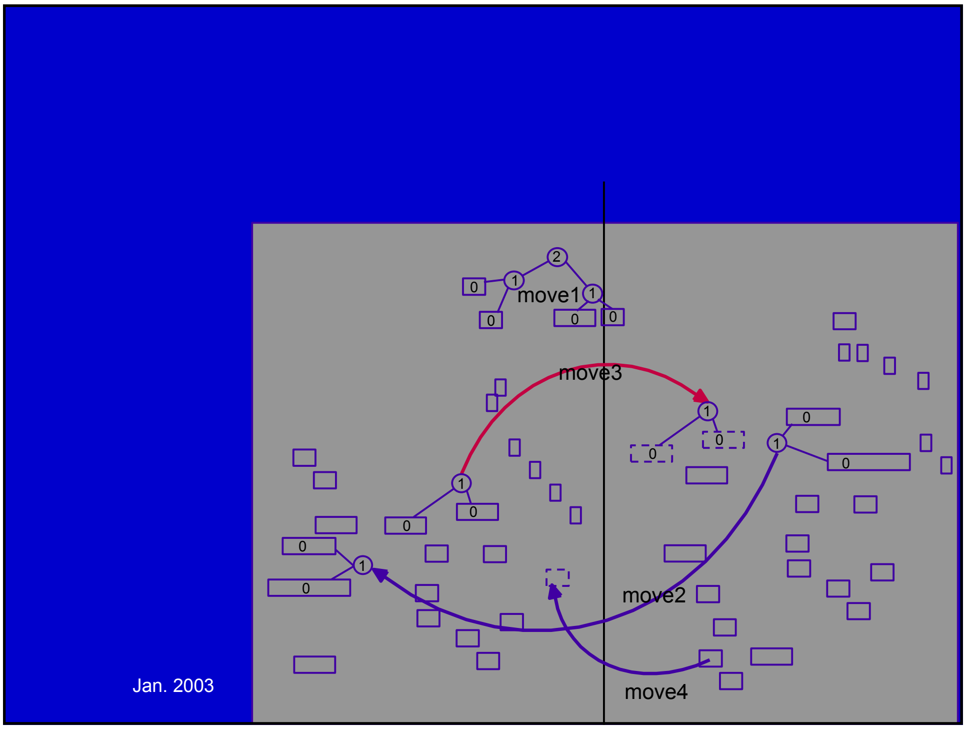
Jan. 2003



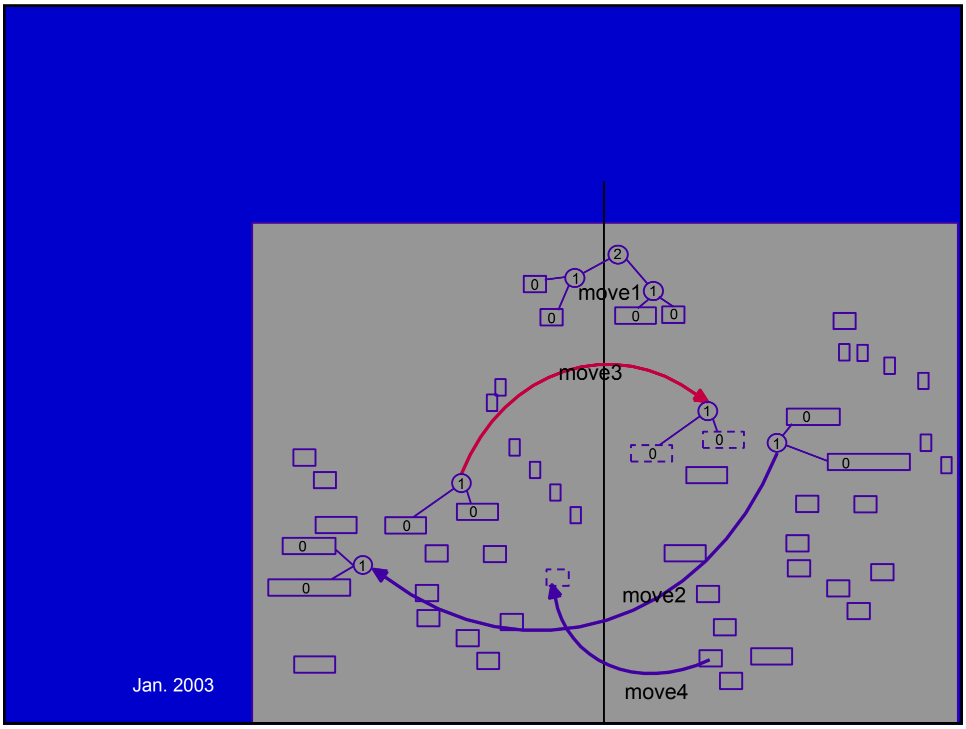
Jan. 2003



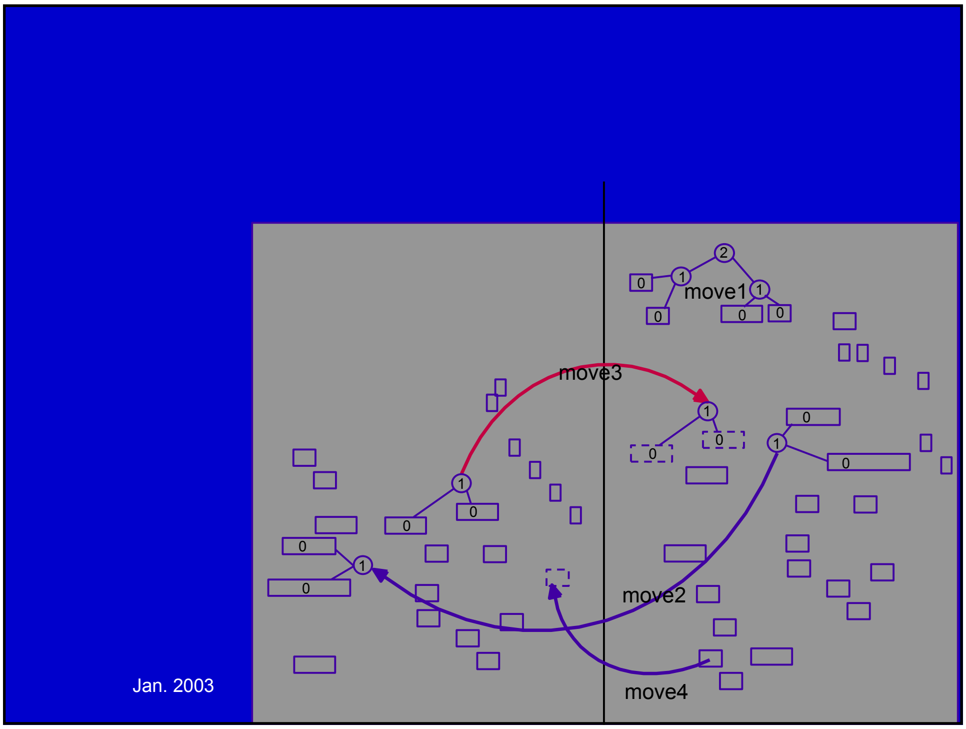
Jan. 2003



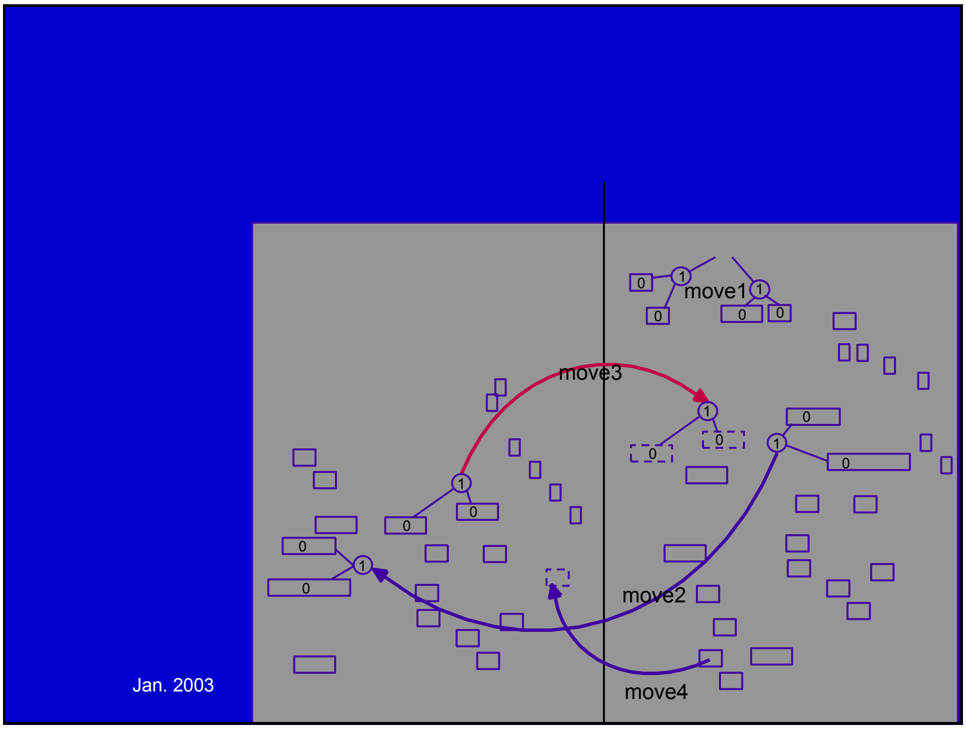
Jan. 2003



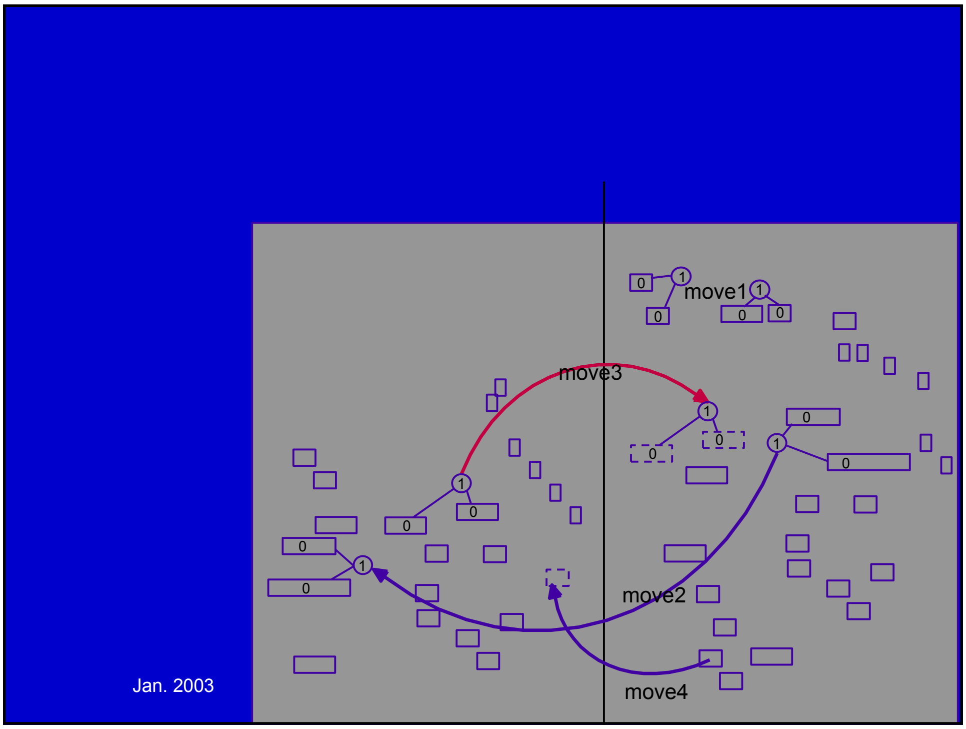
Jan. 2003



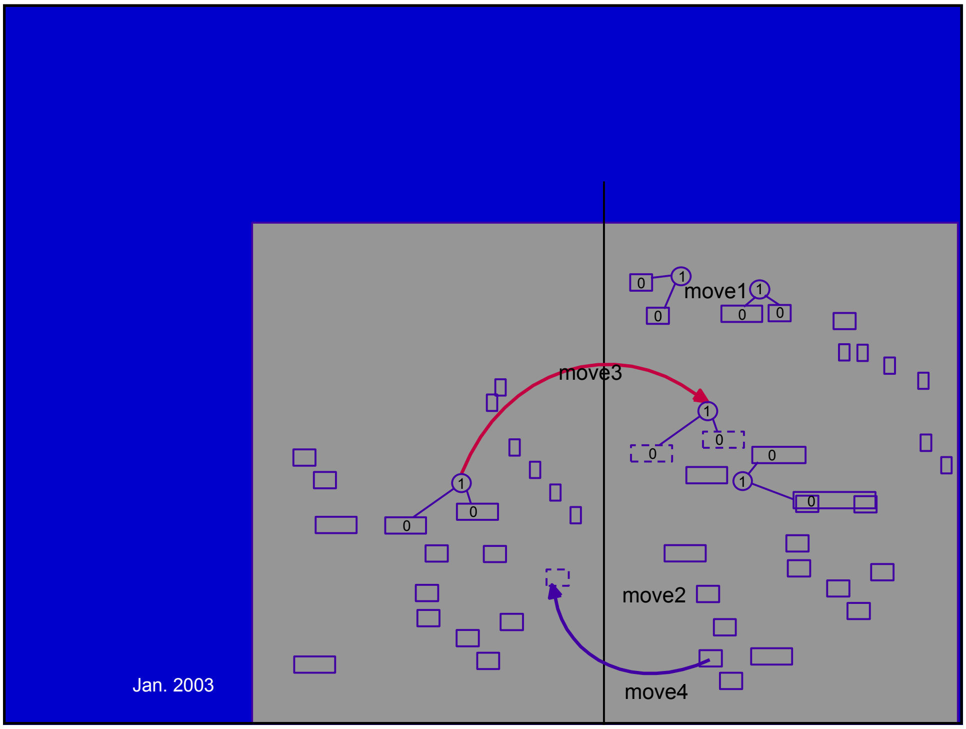
Jan. 2003



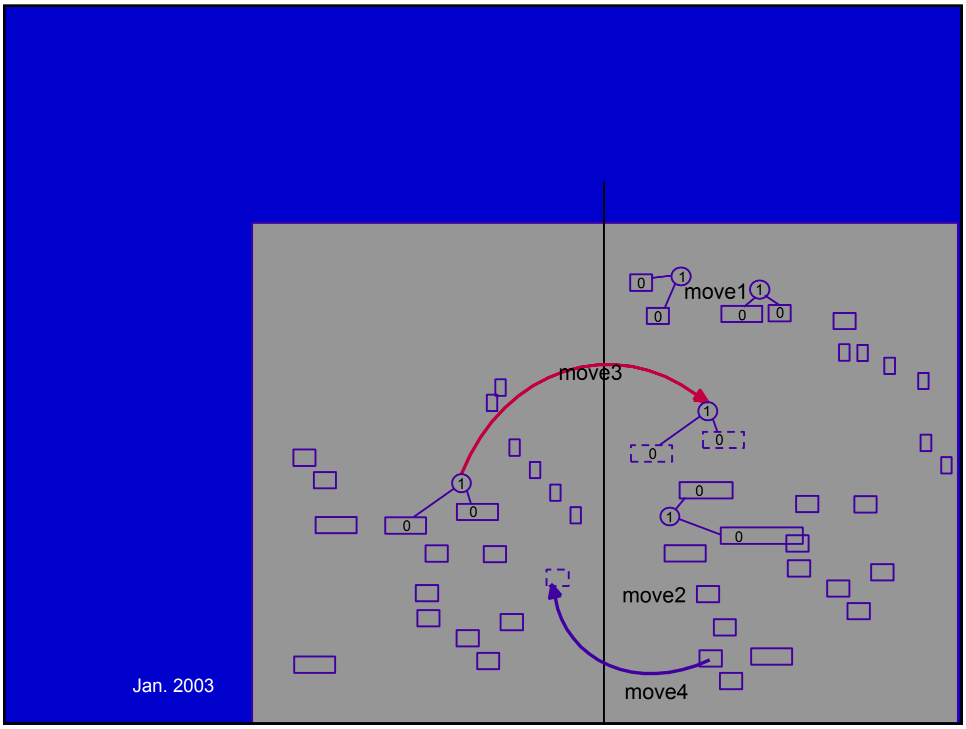
Jan. 2003



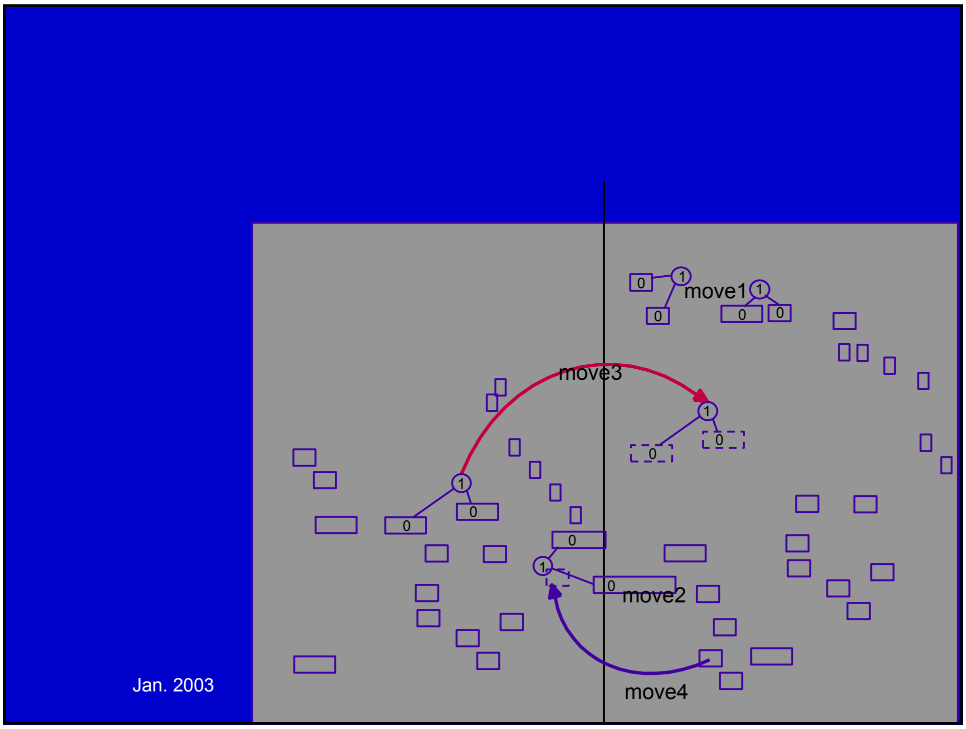
Jan. 2003



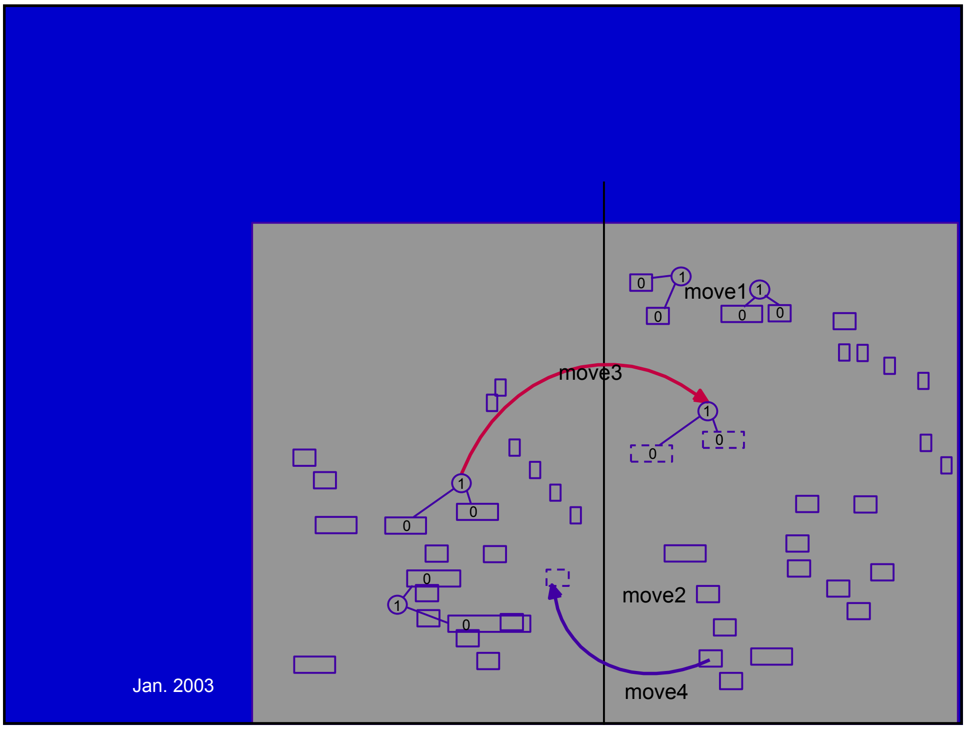
Jan. 2003



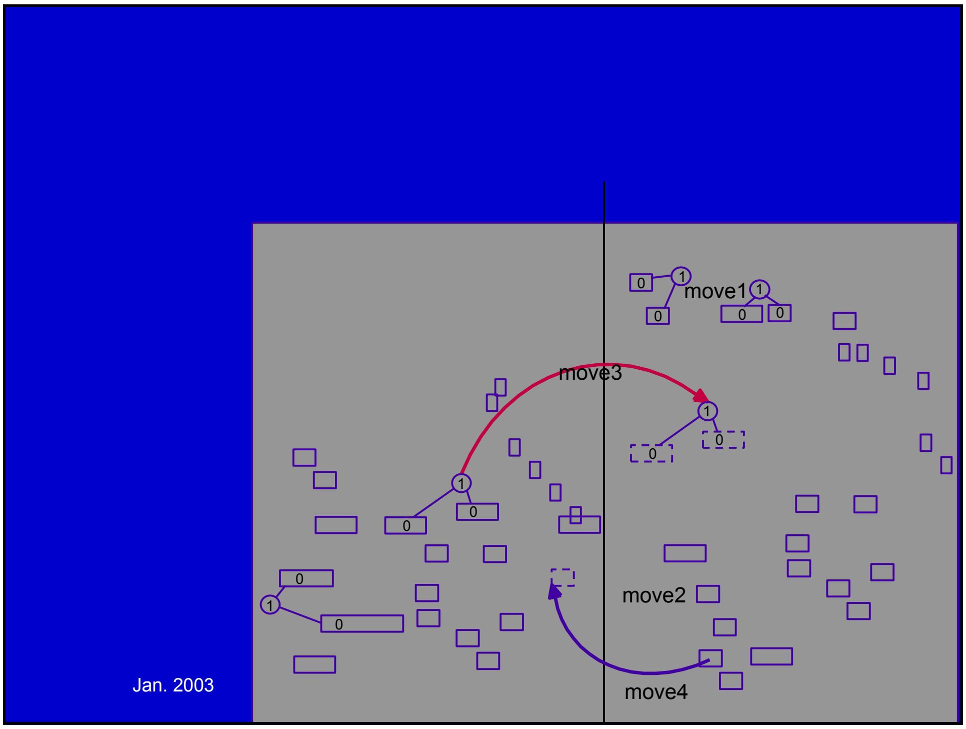
Jan. 2003



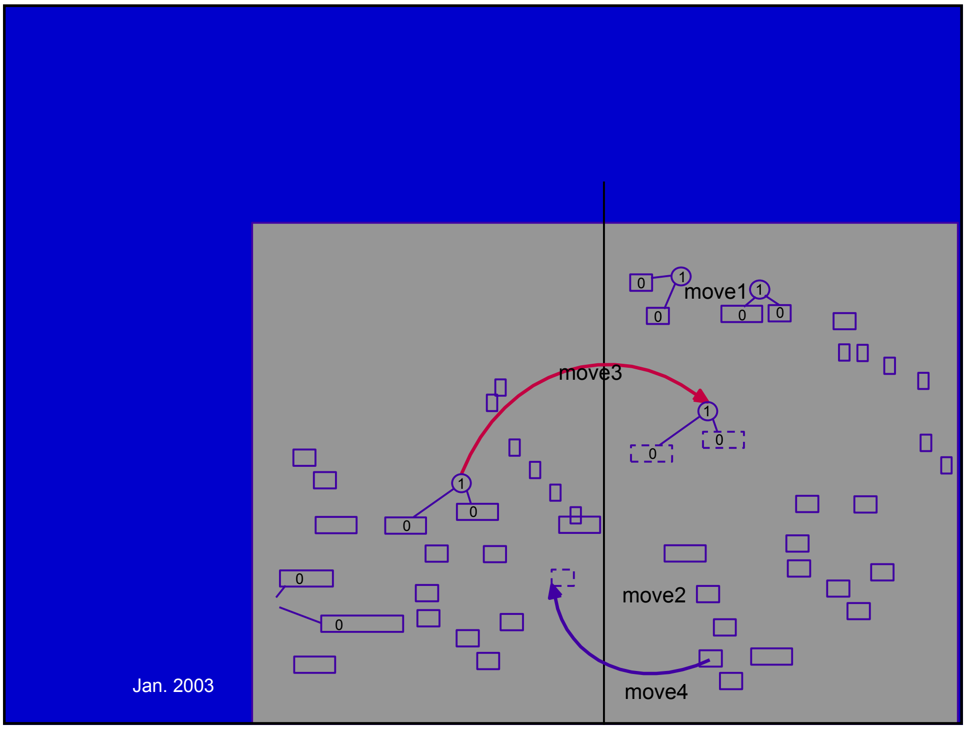
Jan. 2003



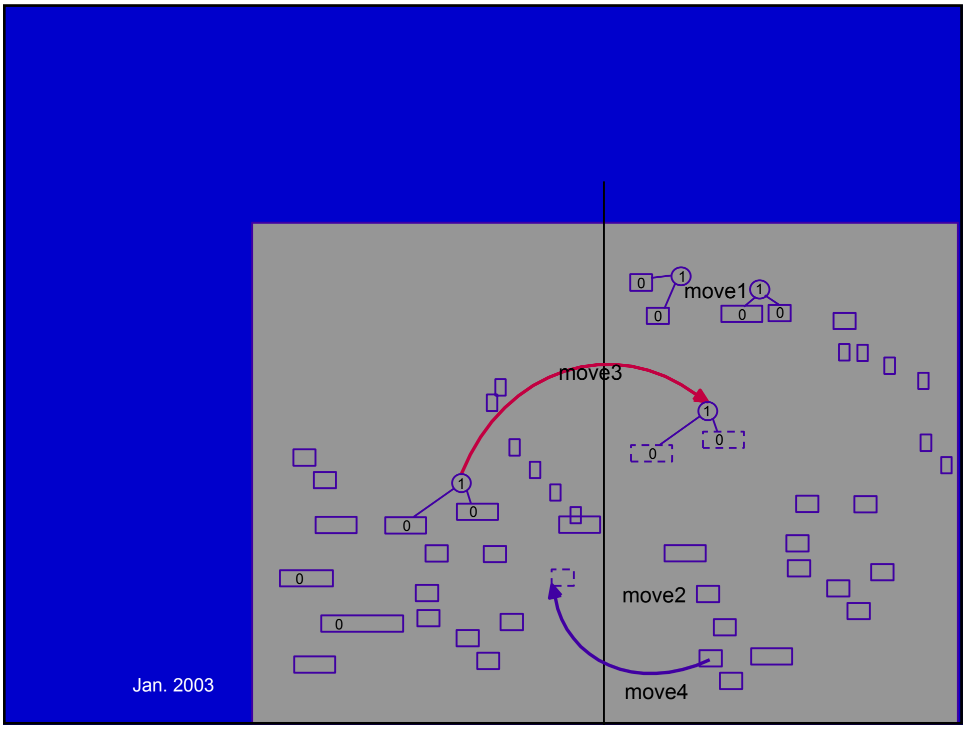
Jan. 2003



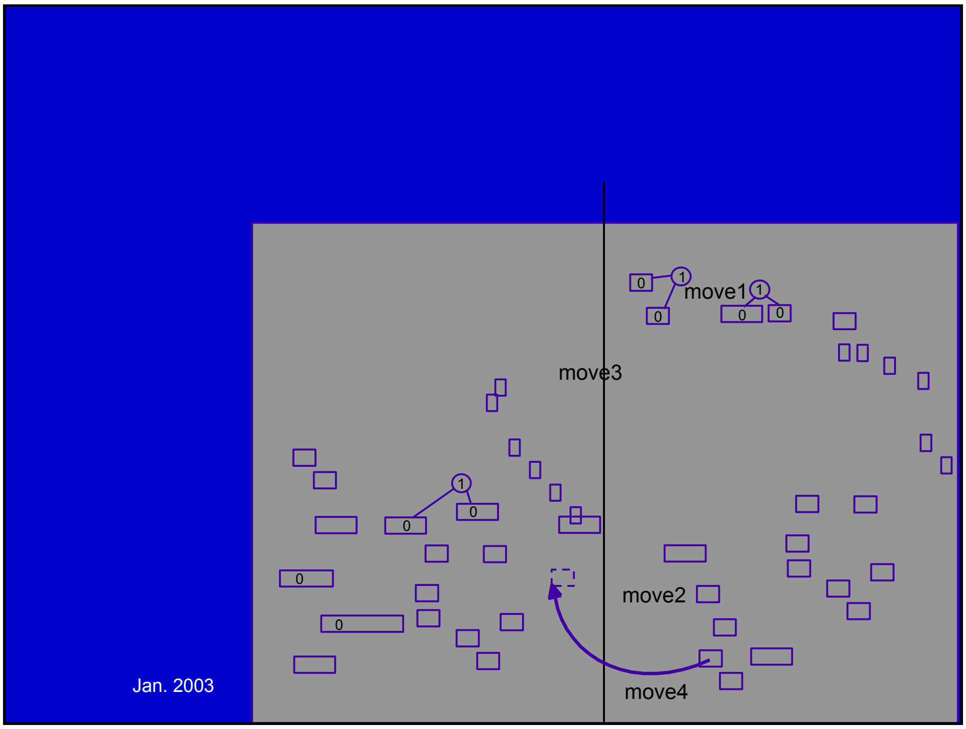
Jan. 2003



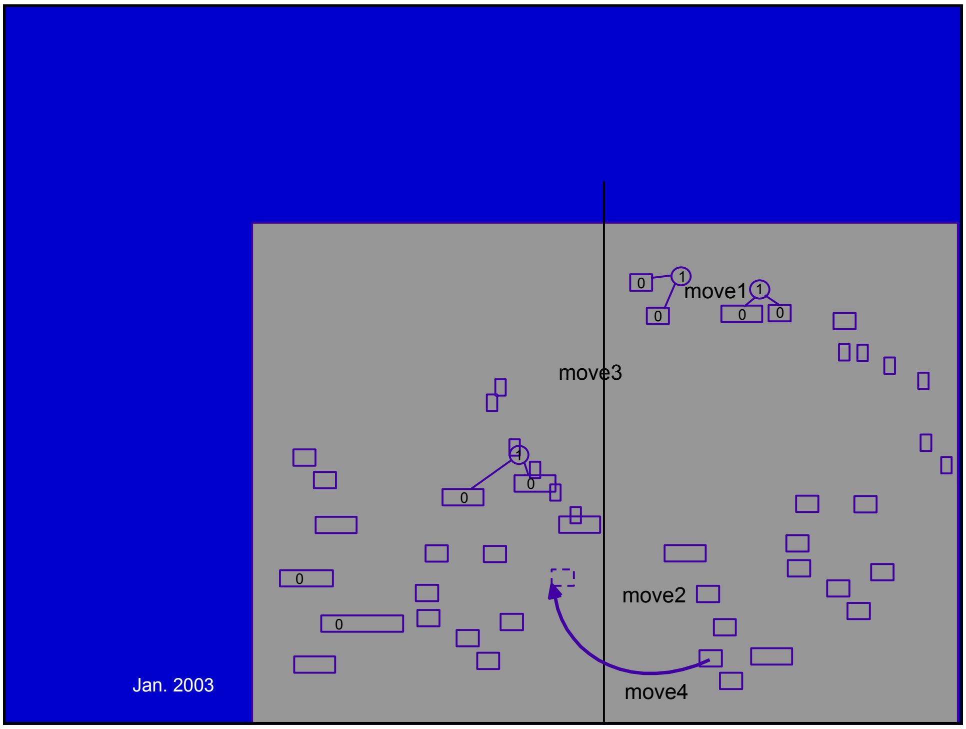
Jan. 2003

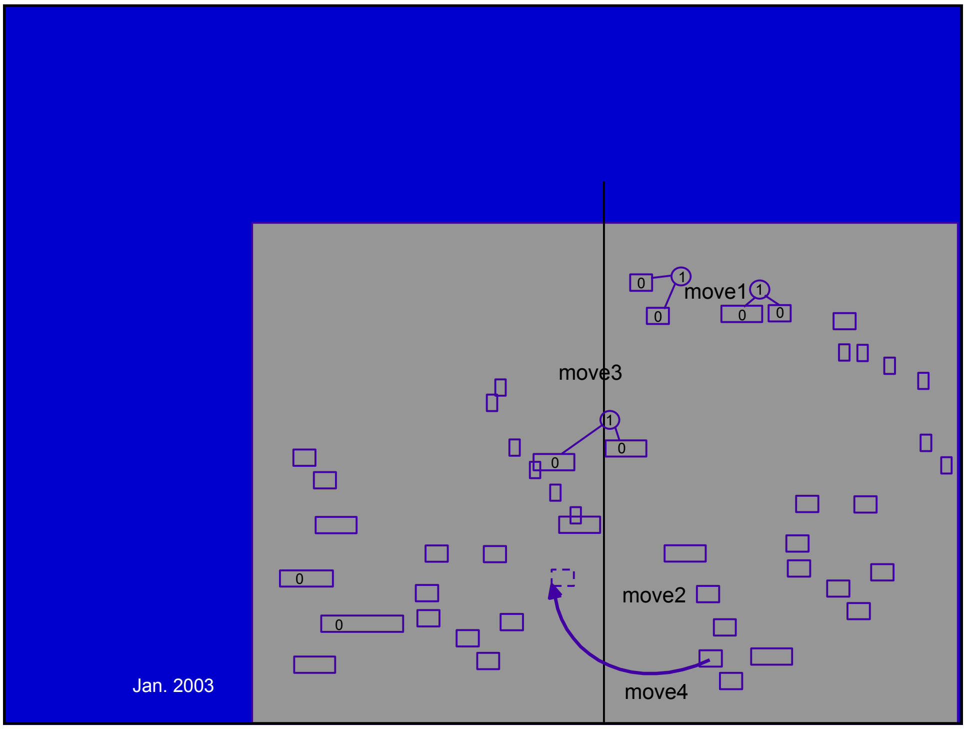


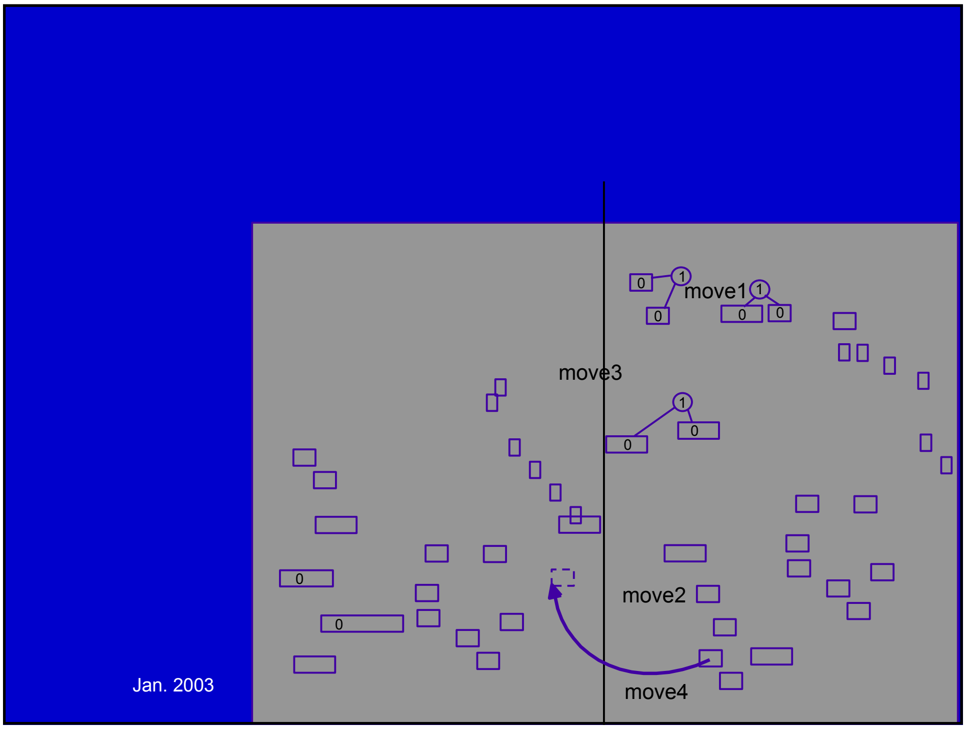
Jan. 2003

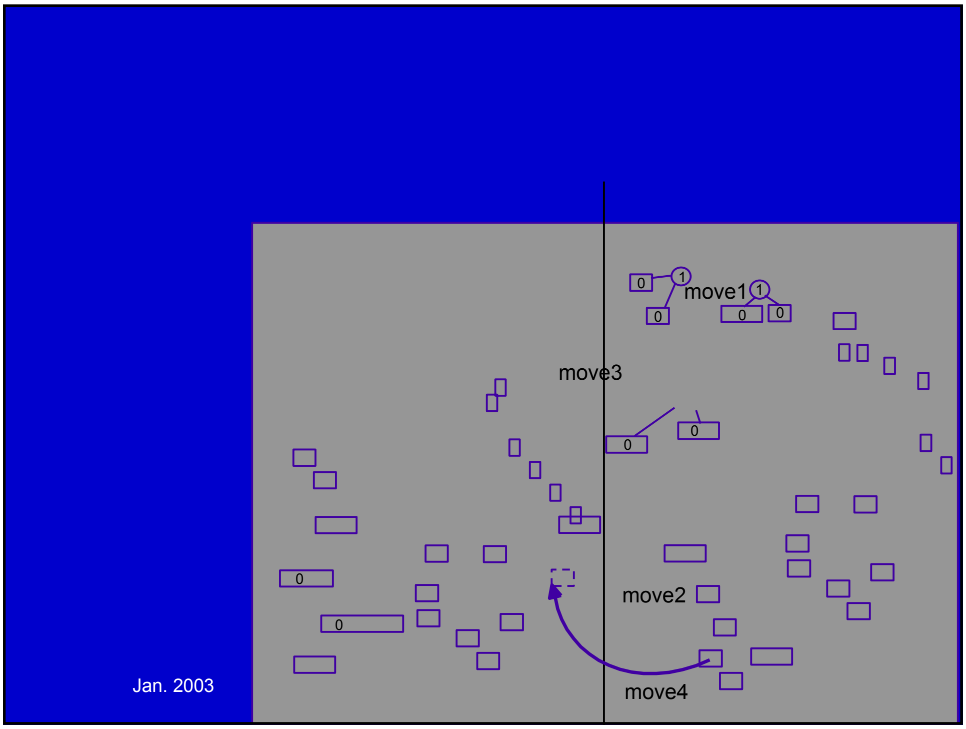


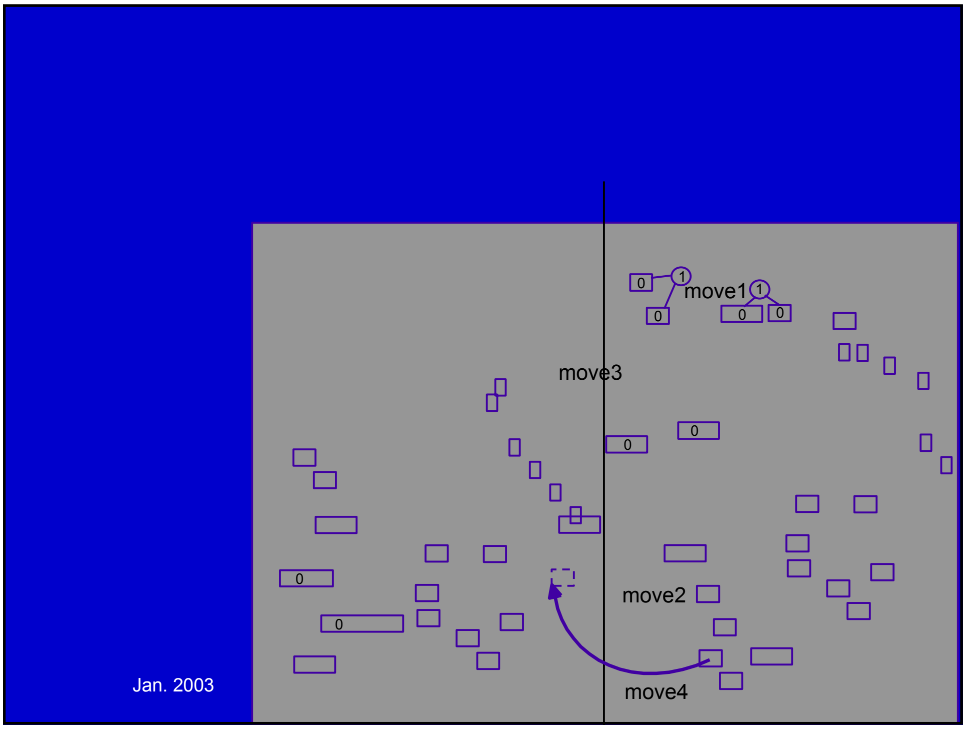
Jan. 2003


0
1
move1
1
0
0
0
move3
1
0
0
0
0
move2
move4
Jan. 2003

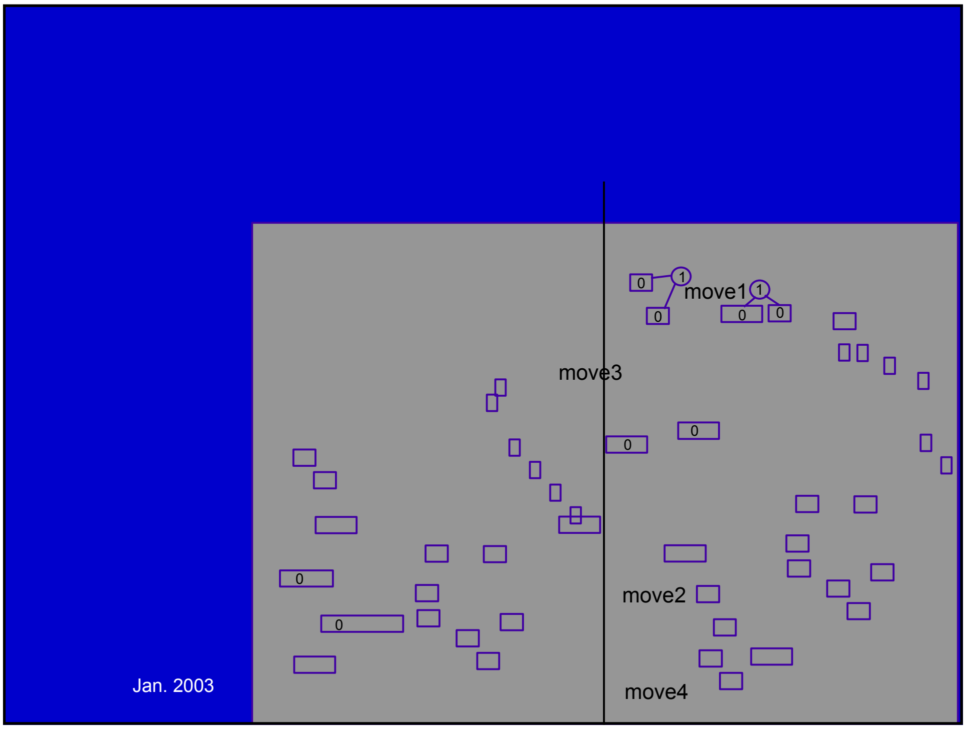

0
1
move1
1
0
0
0
move3
1
0
0
0
0
move2
move4
Jan. 2003


0
1
move1
1
0
0
0
move3
1
0
0
0
0
move2
move4
Jan. 2003

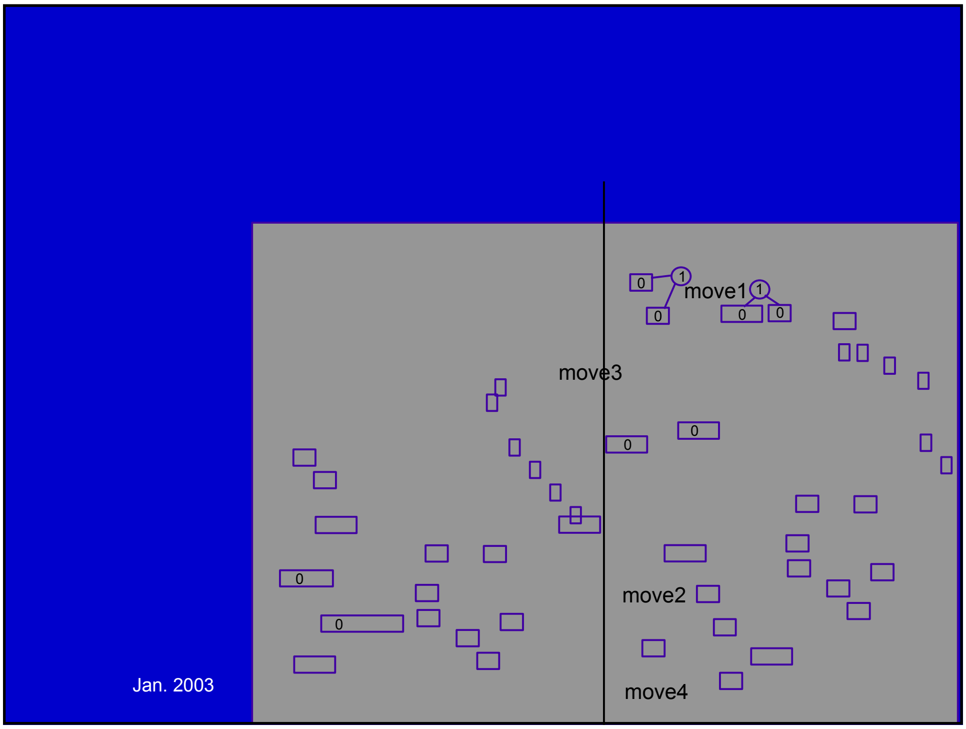

0
1
move1
1
0
0
0
move3
0
0
0
0
move2
move4
Jan. 2003

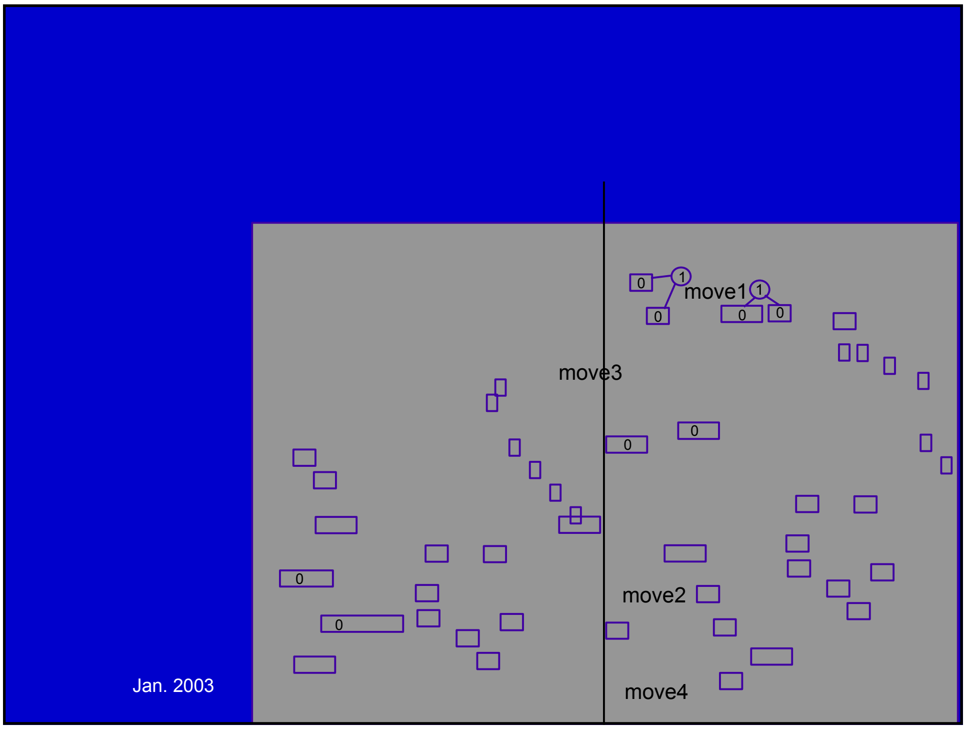


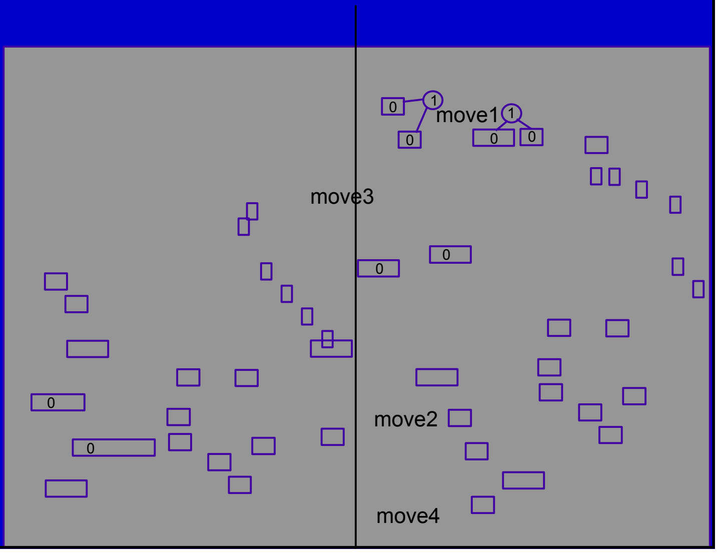
Jan. 2003



Jan. 2003


0
1
move1
1
0
0
0
move3
0
0
0
0
move2
move4
Jan. 2003


0
1
move1
1
0
0
0
move3
0
0
0
0
move2
move4
Jan. 2003

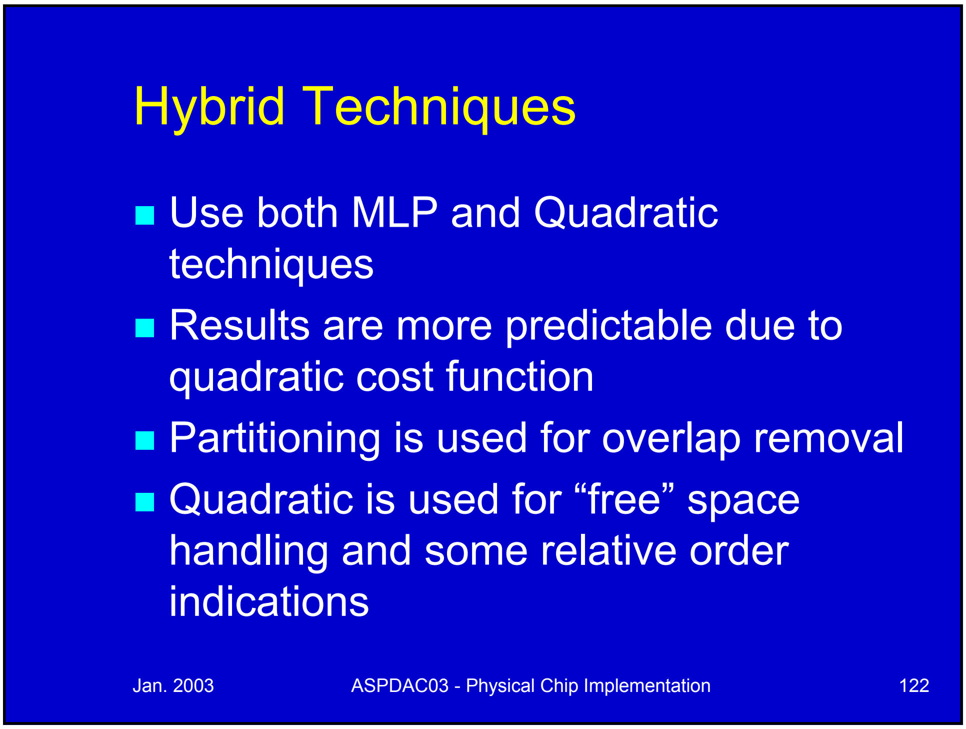


MLP/FM Partitioning Cons:

- Does not know how to handle “free” space
- Results tend to be erratic, ie results from run to run have significant variation

MLP/FM Partitioning Pros:

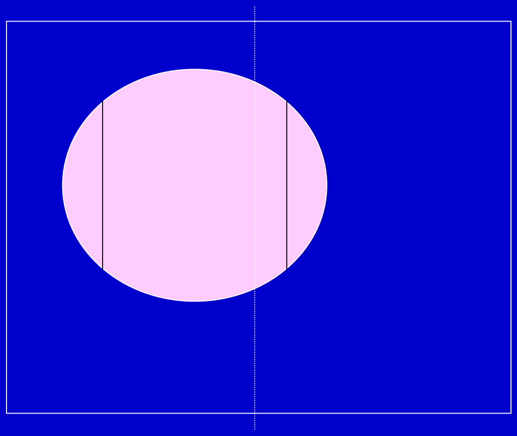
- Handles designs that have no fixed connection points
- Very fast - can handle large designs

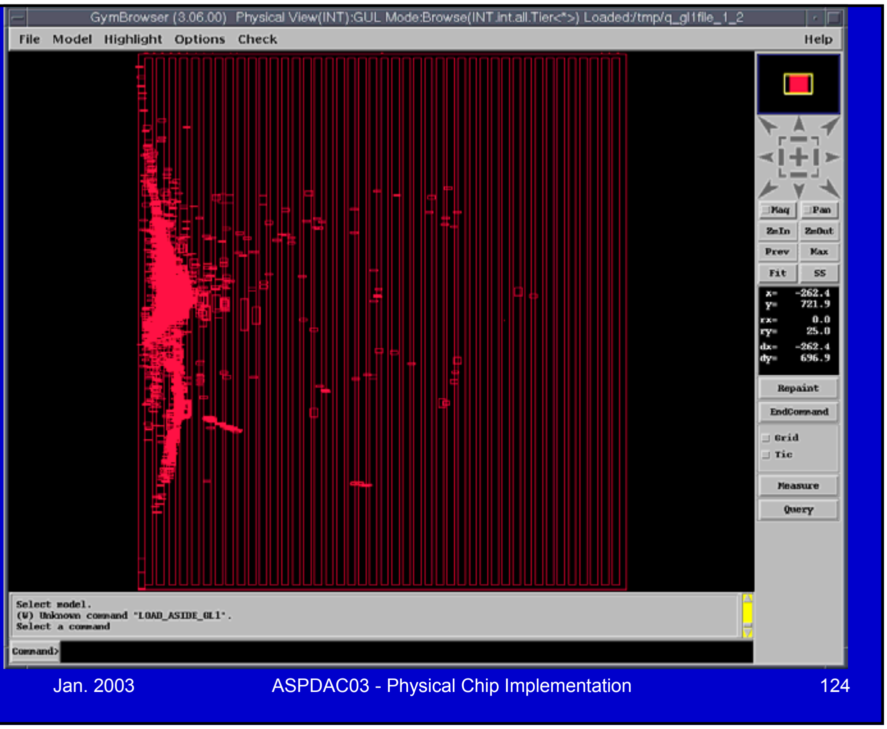


Hybrid Techniques

- Use both MLP and Quadratic techniques
- Results are more predictable due to quadratic cost function
- Partitioning is used for overlap removal
- Quadratic is used for “free” space handling and some relative order indications

Quadratic Partitioning

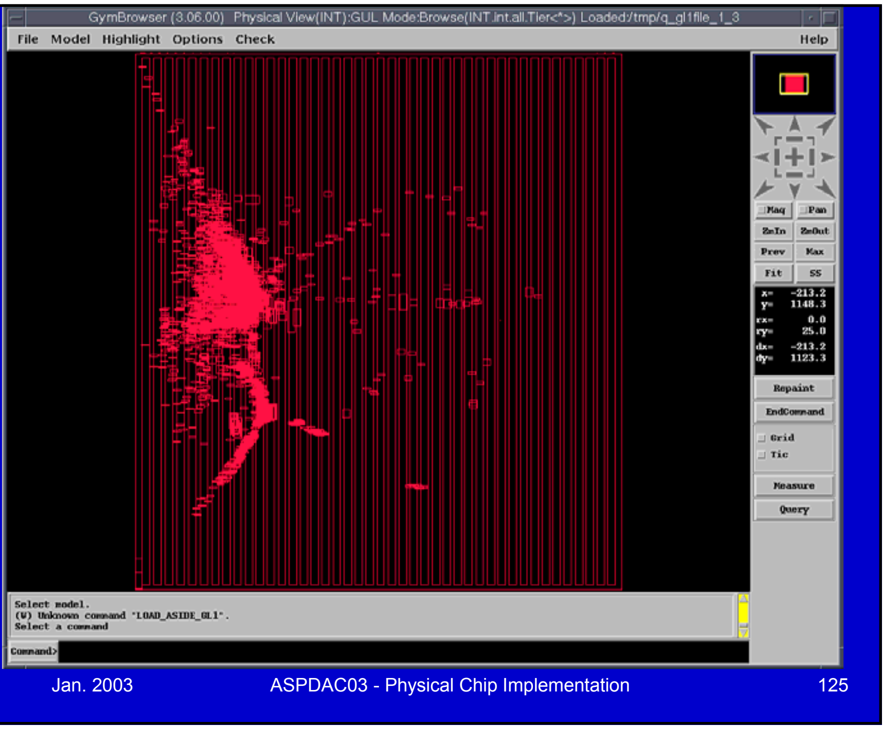




GymBrowser (3.06.00) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/tmp/q_gl1file_1_2

```
x=    -262.4
y=     721.9
rx=      0.0
ry=     25.0
dx=   -262.4
dy=    696.9
```

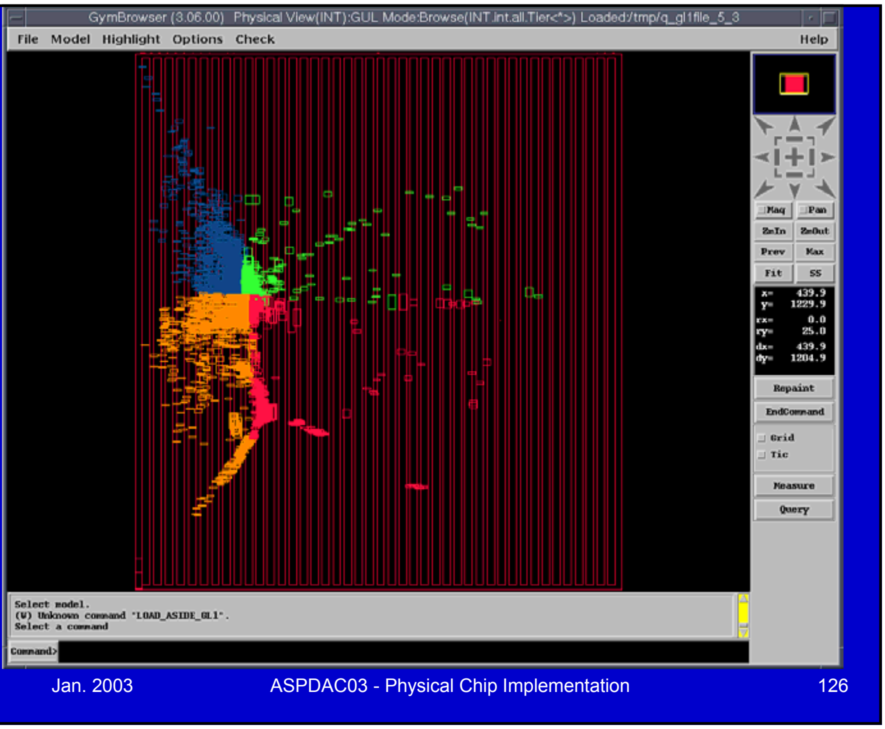
```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
Command>
```



```
GymBrowser (3.06.00)  Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/tmp/q_gl1file_1_3
```

```
x=   -213.2
y=   1148.3
rx=     0.0
ry=    25.0
dx=  -213.2
dy=  1123.3
```

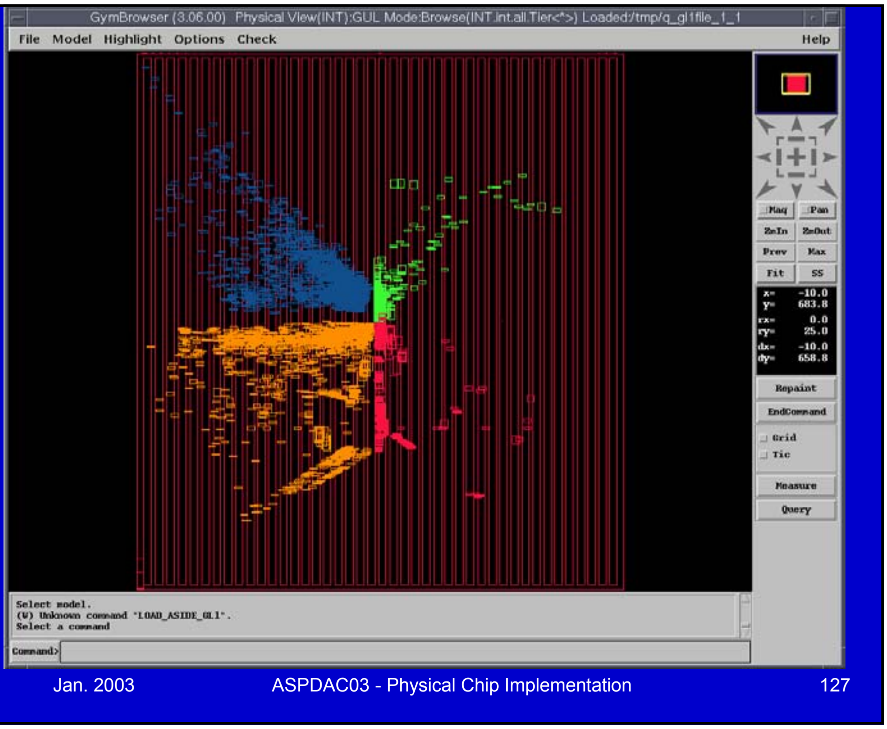
```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
Command>
```



```
GymBrowser (3.06.00)  Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/tmp/q_gl1file_5_3
```

```
x=    439.9
y=   1229.9
rx=     0.0
ry=    25.0
dx=   439.9
dy=  1204.9
```

```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
Command>
```



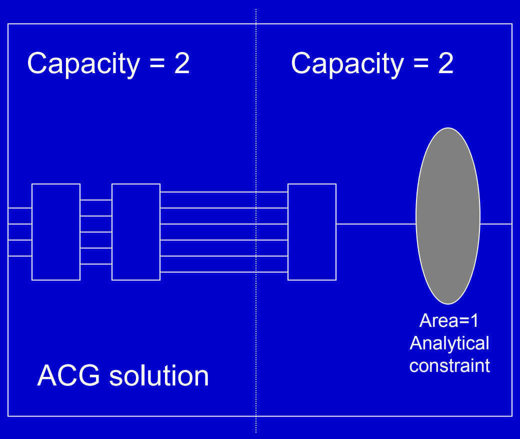
x= -10.0
y= 683.8
rx= 0.0
ry= 25.0
dx= -10.0
dy= 658.8

```
Select model.
(W) Unknown command "LOAD_ASIDE_GL1".
Select a command
```

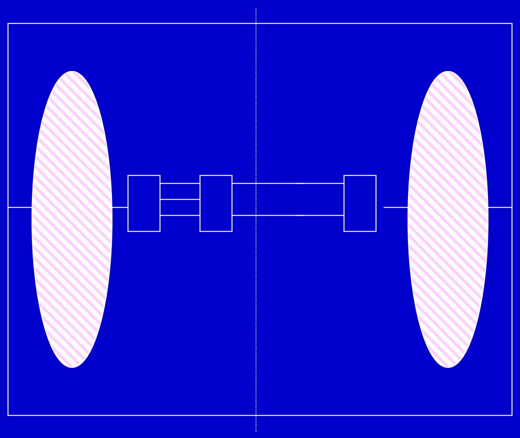
Analytical Constraint Generation

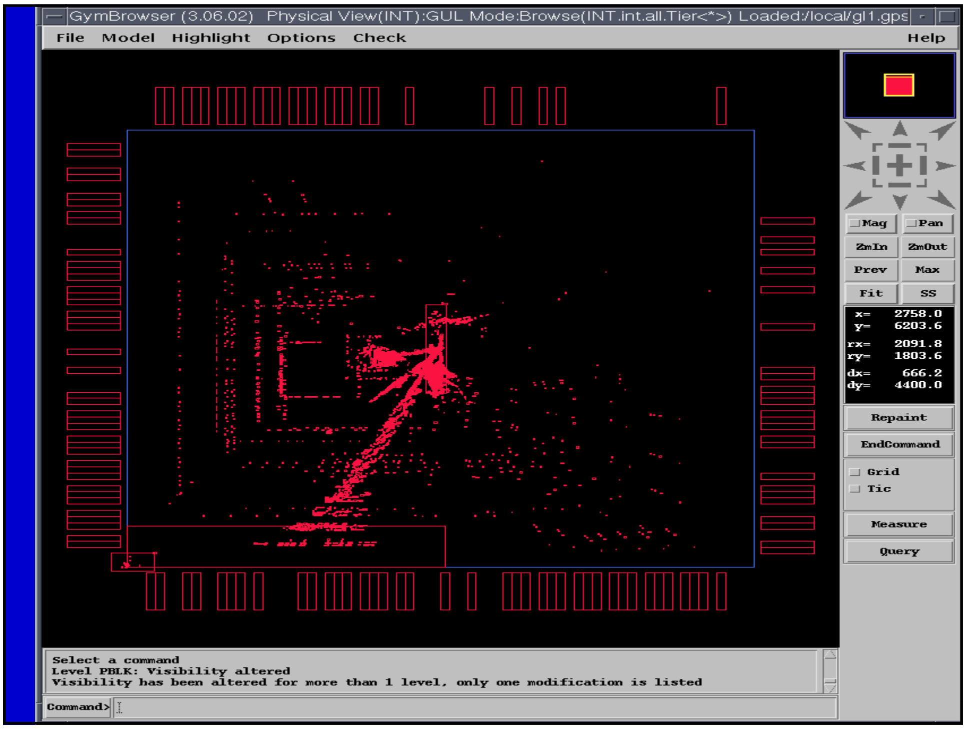
- Combine Quadratic techniques with MLP
- Use Quadratic solution to determine global position (ie balance)
- Use MLP to determine relative ordering of cells

Analytical Constraint Generation



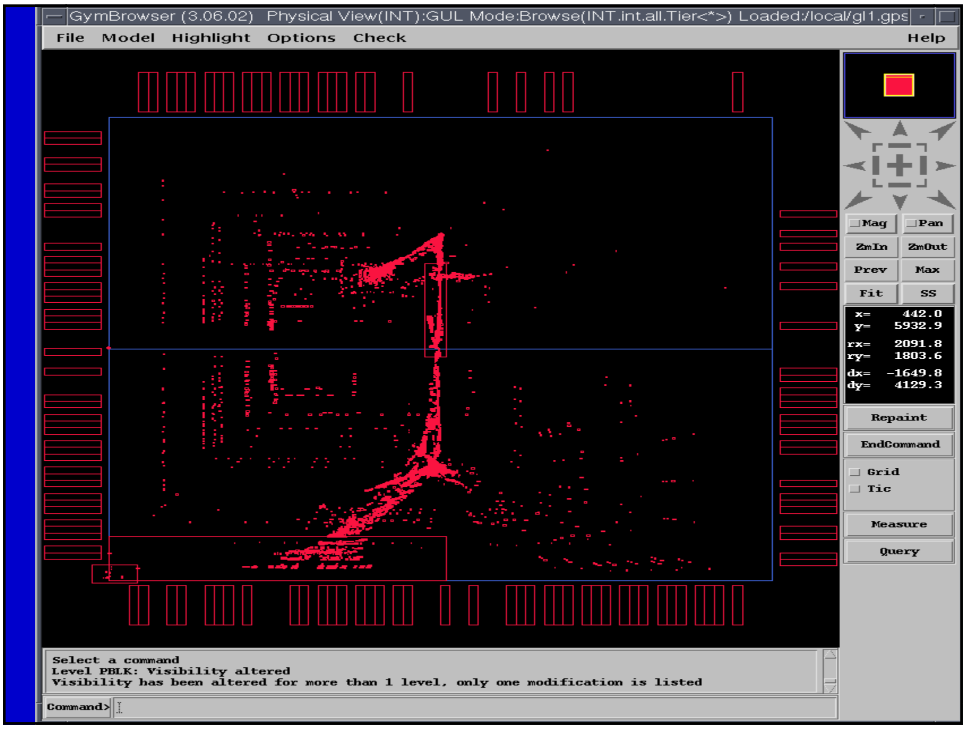
Analytical Constraint Generation




GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/local/gl1.gps
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

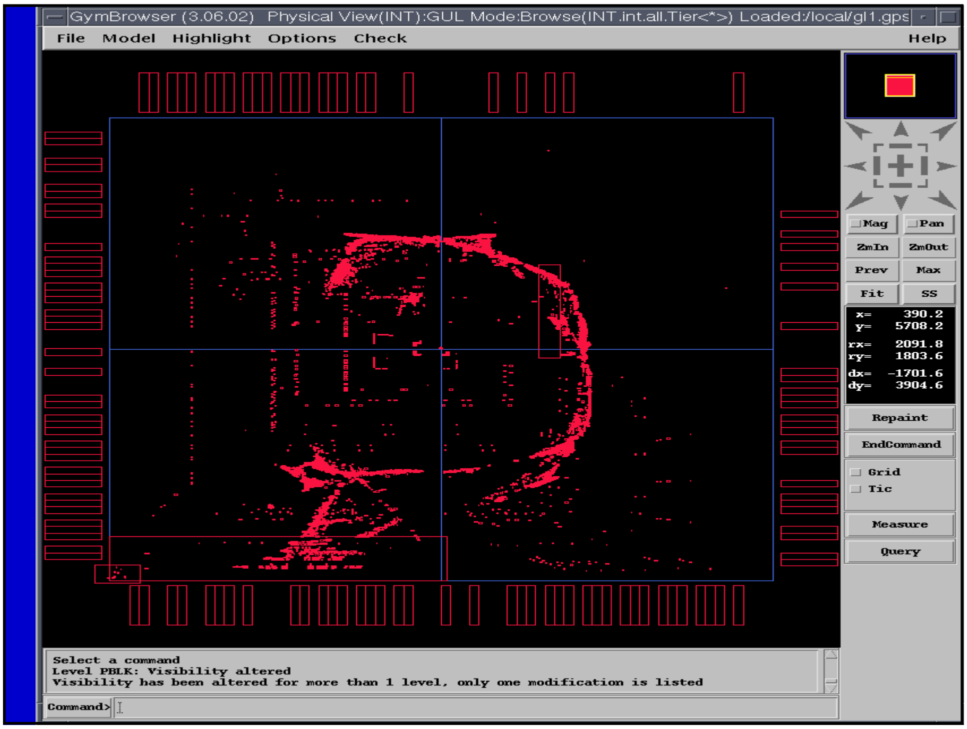
```
x=    2758.0
y=    6203.6
rx=   2091.8
ry=   1803.6
dx=    666.2
dy=   4400.0
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/local/gl1.gps
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

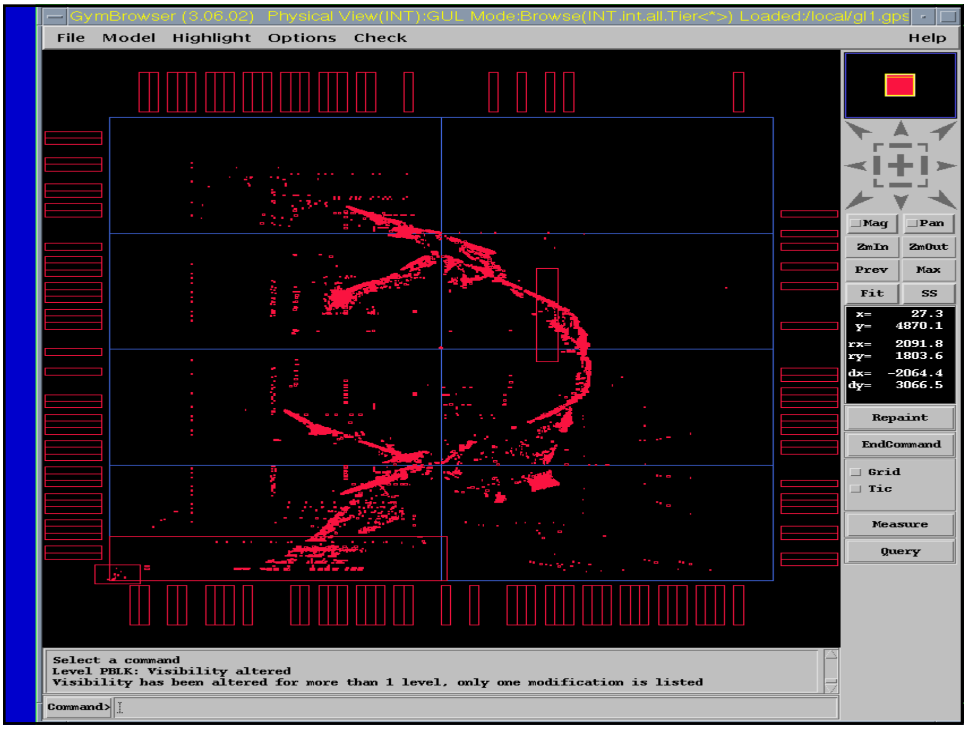
```
x=     442.0
y=    5932.9
rx=   2091.8
ry=   1803.6
dx=  -1649.8
dy=   4129.3
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/local/gl1.gps
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

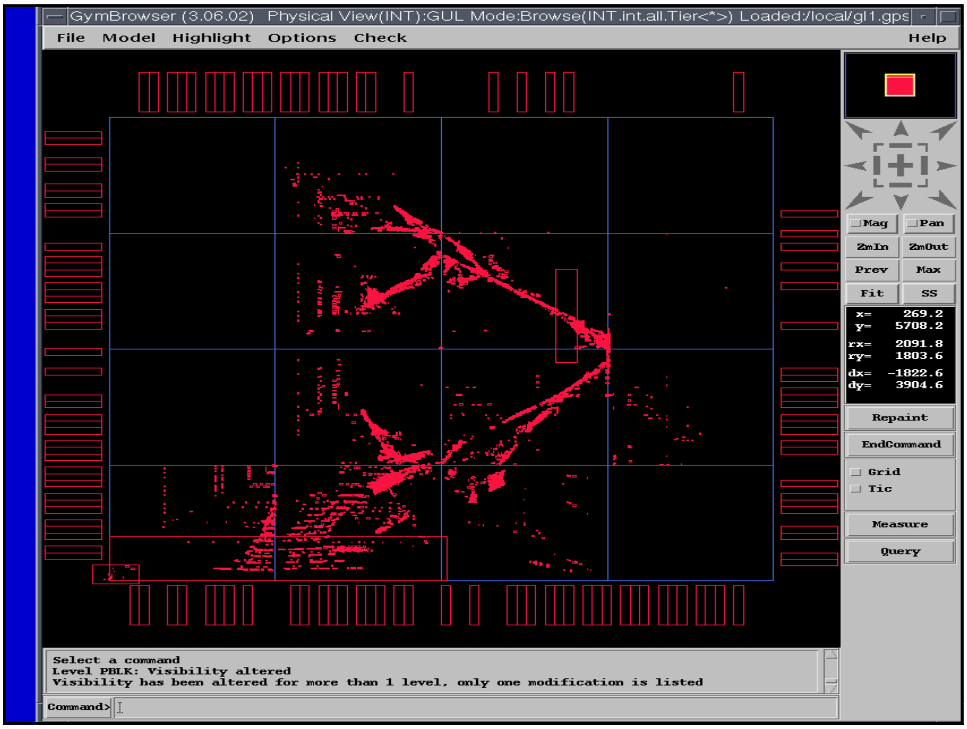
```
x=    390.2
y=   5708.2
rx=  2091.8
ry=  1803.6
dx= -1701.6
dy=  3904.6
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/local/gl1.gps
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

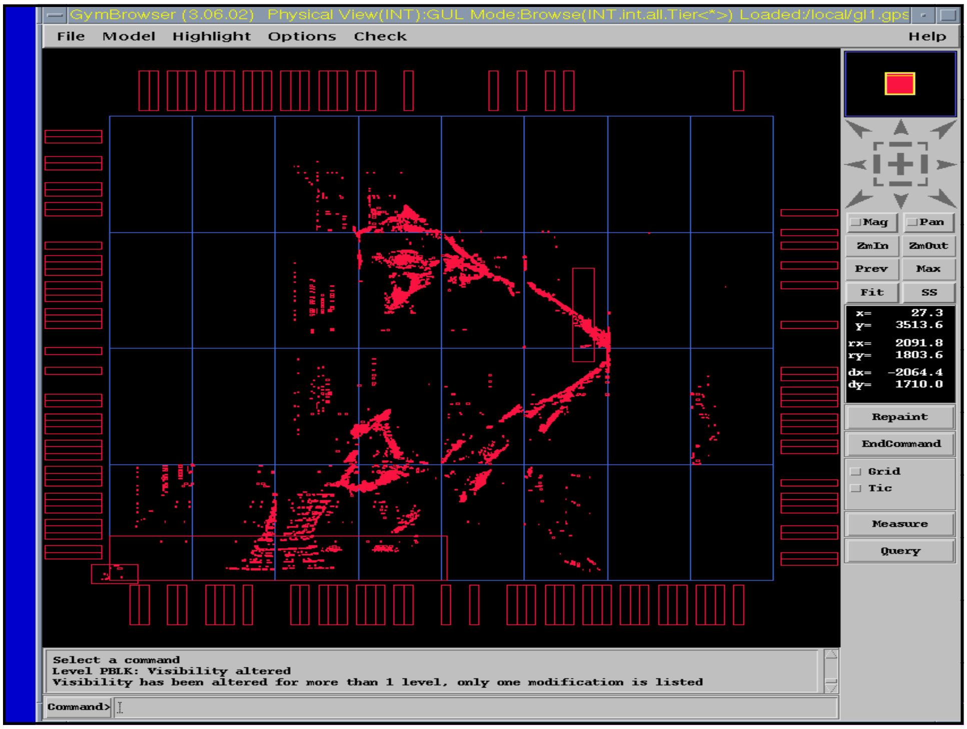
```
x=     27.3
y=   4870.1
rx=  2091.8
ry=  1803.6
dx= -2064.4
dy=  3066.5
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```



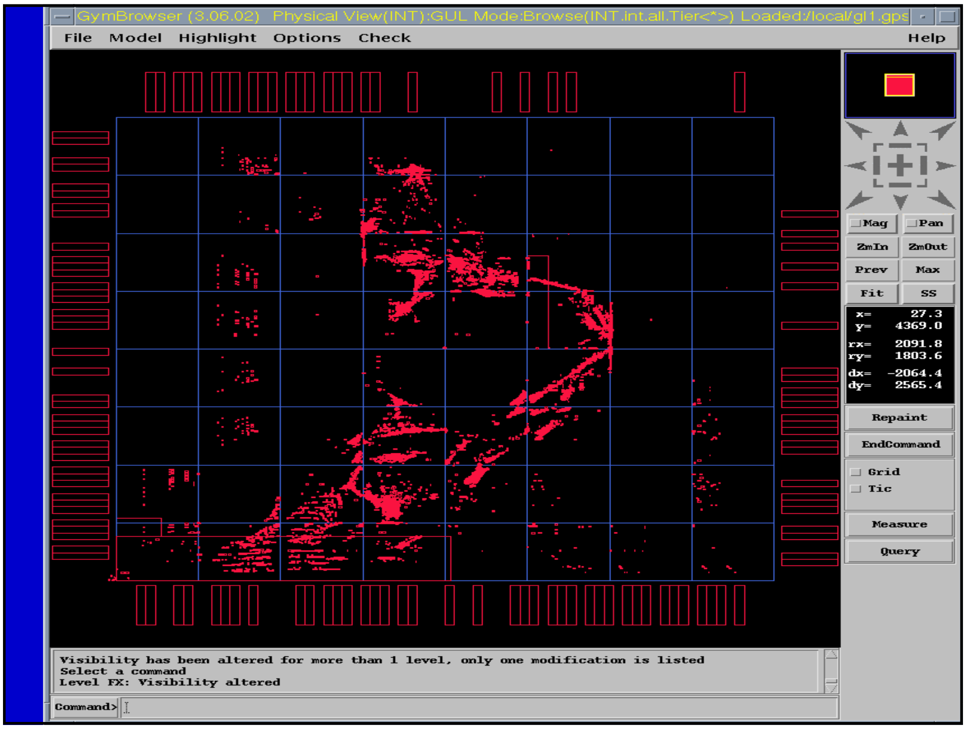
```
x=     269.2
y=    5708.2
rx=   2091.8
ry=   1803.6
dx=  -1822.6
dy=   3904.6
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```



```
x=      27.3
y=    3513.6
rx=   2091.8
ry=   1803.6
dx=  -2064.4
dy=   1710.0
```

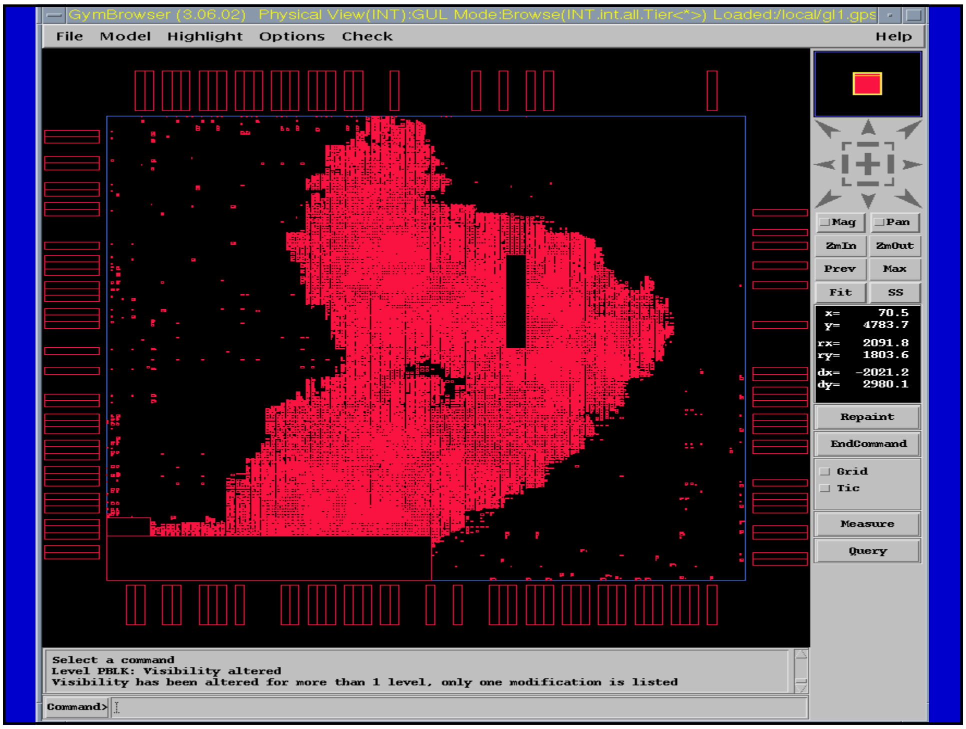
```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```



```
x=      27.3
y=    4369.0
rx=   2091.8
ry=   1803.6
dx=  -2064.4
dy=   2565.4
```

```
Visibility has been altered for more than 1 level, only one modification is listed
Select a command
Level FX: Visibility altered
```

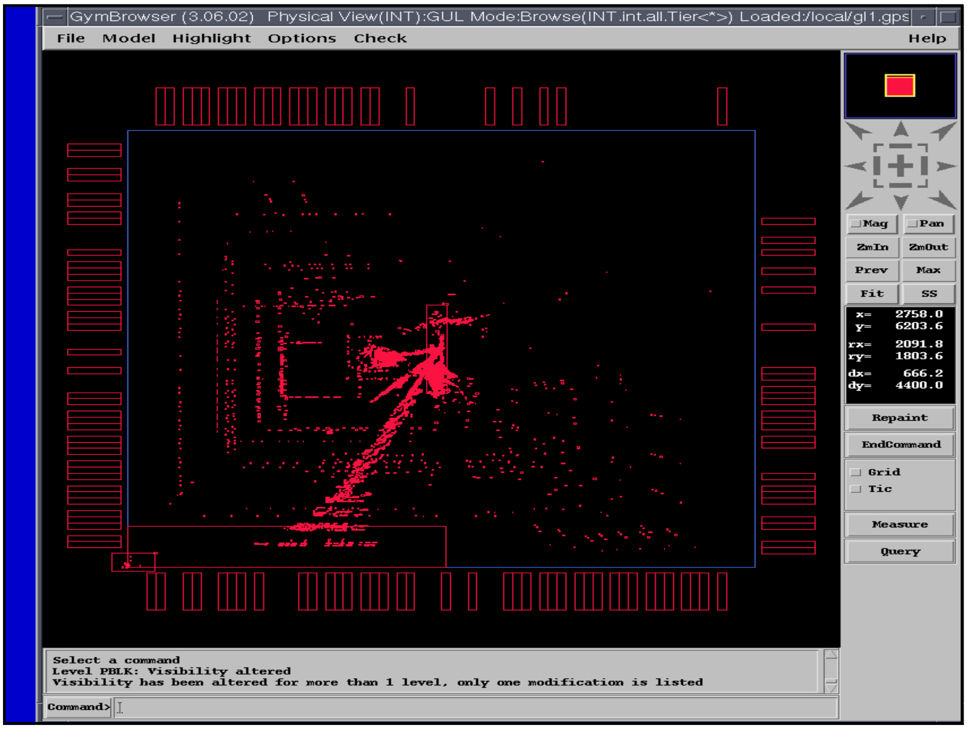
```
Command>
```



```
x=      70.5
y=    4783.7
rx=   2091.8
ry=   1803.6
dx=  -2021.2
dy=   2980.1
```

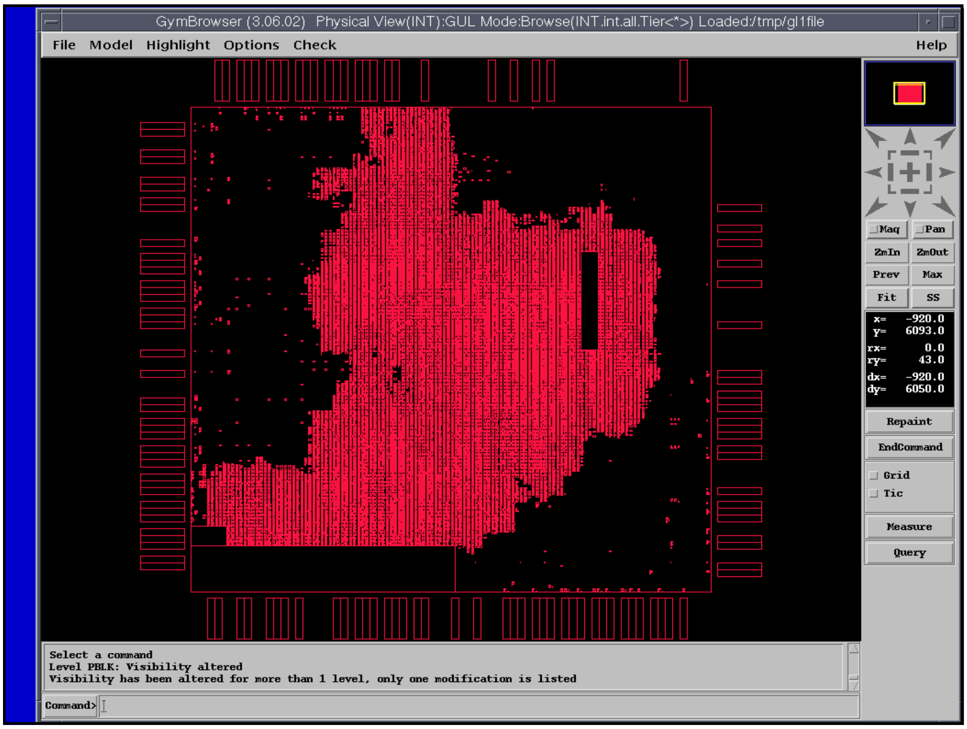
```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```

```
Command>
```


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/local/gl1.gps
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

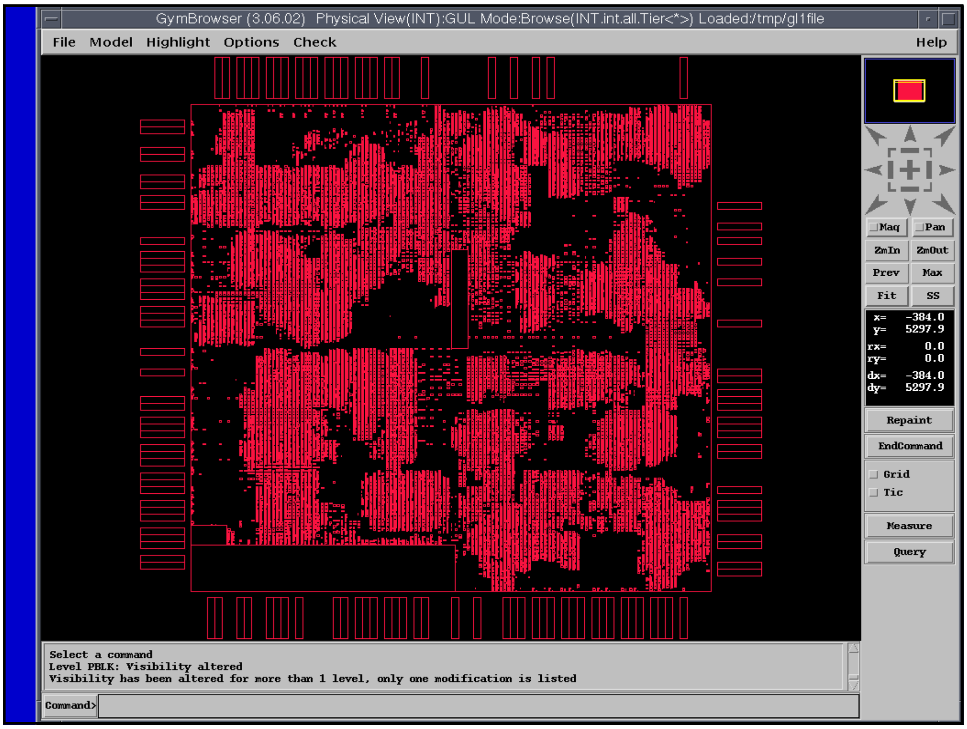
```
x=    2758.0
y=    6203.6
rx=   2091.8
ry=   1803.6
dx=    666.2
dy=   4400.0
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```



```
x=    -920.0
y=    6093.0
rx=      0.0
ry=     43.0
dx=   -920.0
dy=   6050.0
```

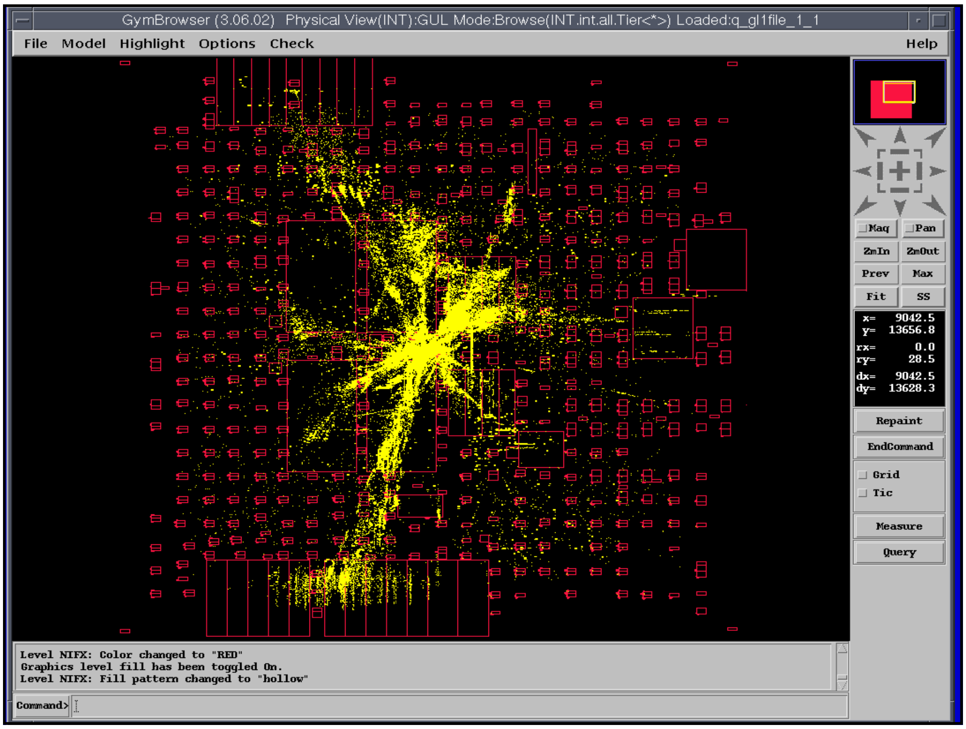
```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Command>
```


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:/tmp/gl1file
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

```
x=   -384.0
y=   5297.9
rx=     0.0
ry=     0.0
dx=  -384.0
dy=  5297.9
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```

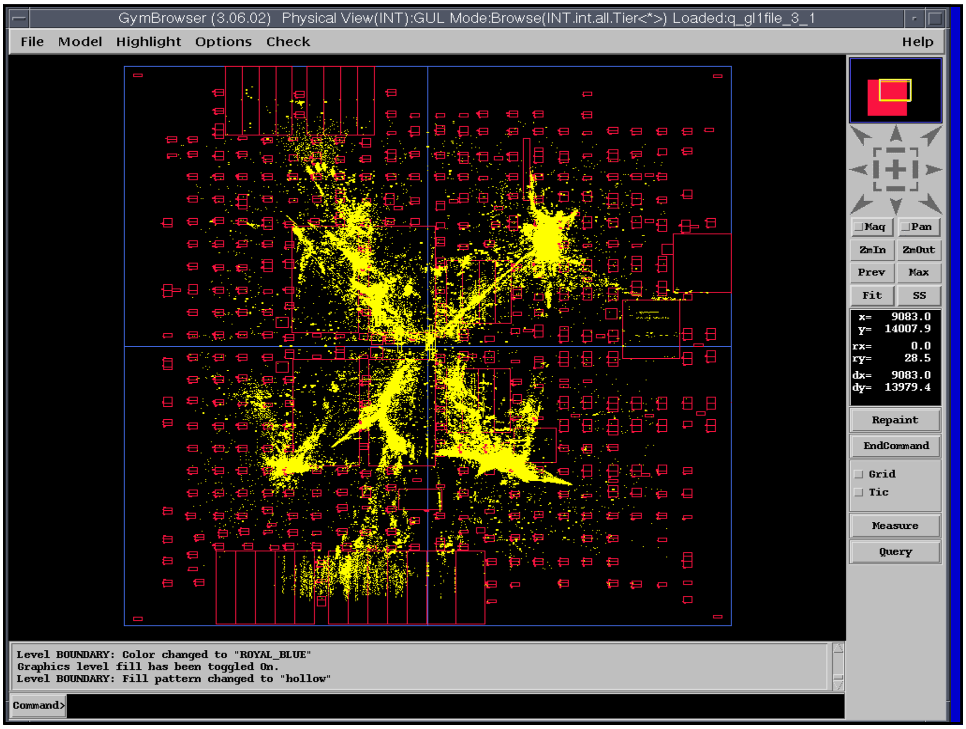
Command>


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:q_gl1file_1_1
File
Model
Highlight
Options
Check
Help
Mag
Pan
ZmIn
ZmOut
Prev
Max
Fit
SS
Repaint
EndCommand
Grid
Tic
Measure
Query

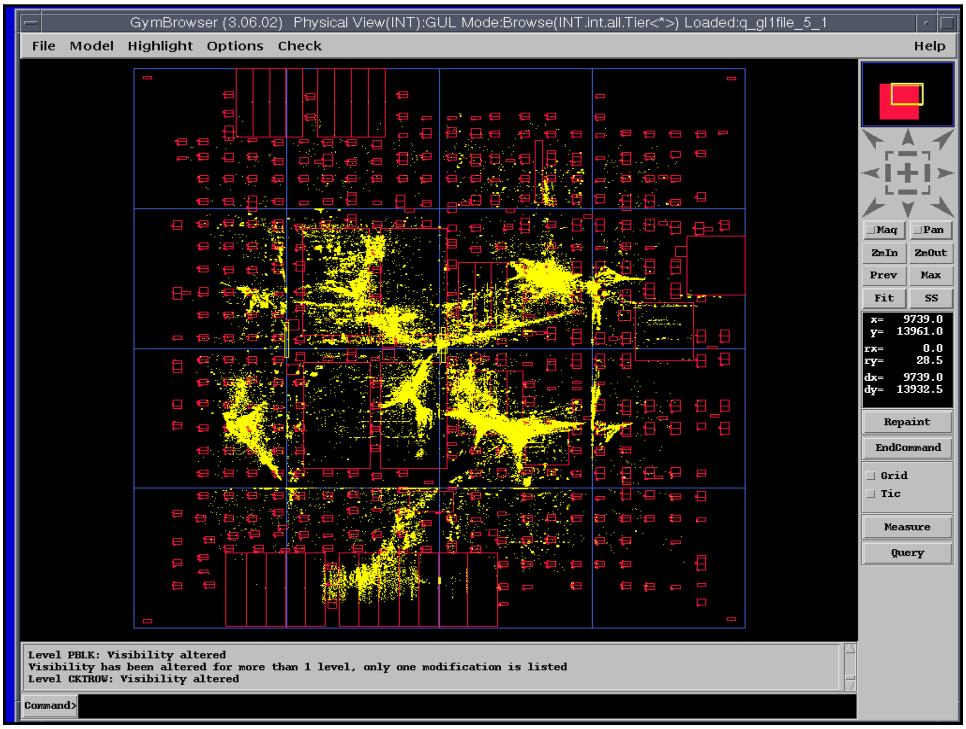
```
x=   9042.5
y=  13656.8
rx=     0.0
ry=    28.5
dx=  9042.5
dy= 13628.3
```

```
Level NIFX: Color changed to "RED"
Graphics level fill has been toggled On.
Level NIFX: Fill pattern changed to "hollow"
```

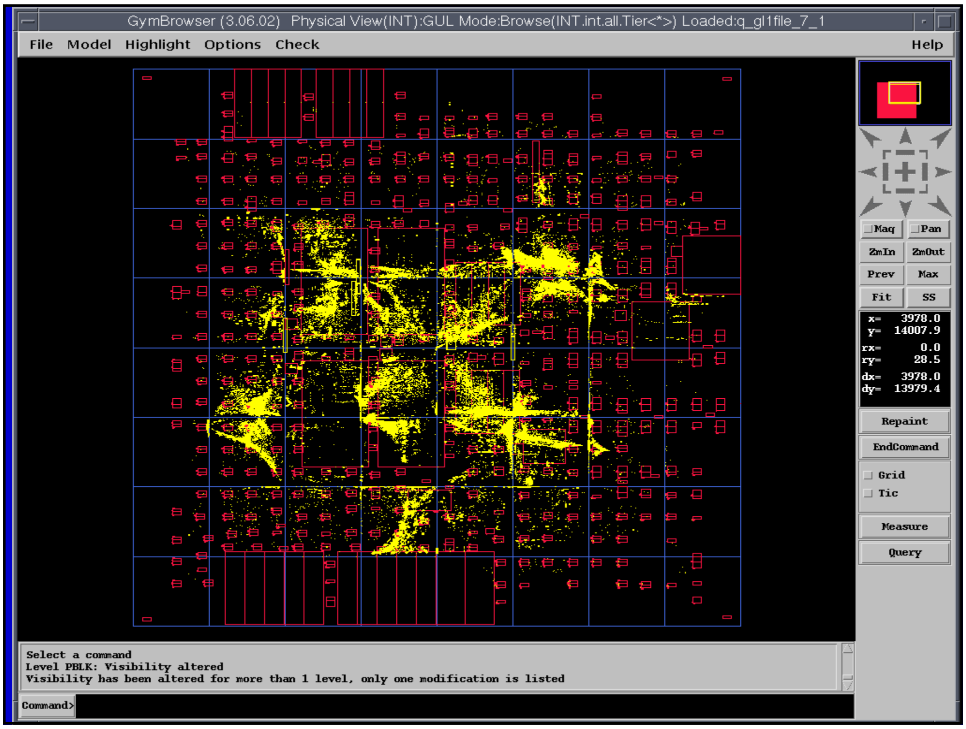
Command>



```
Level BOUNDARY: Color changed to "ROYAL_BLUE"
Graphics level fill has been toggled On.
Level BOUNDARY: Fill pattern changed to "hollow"
```

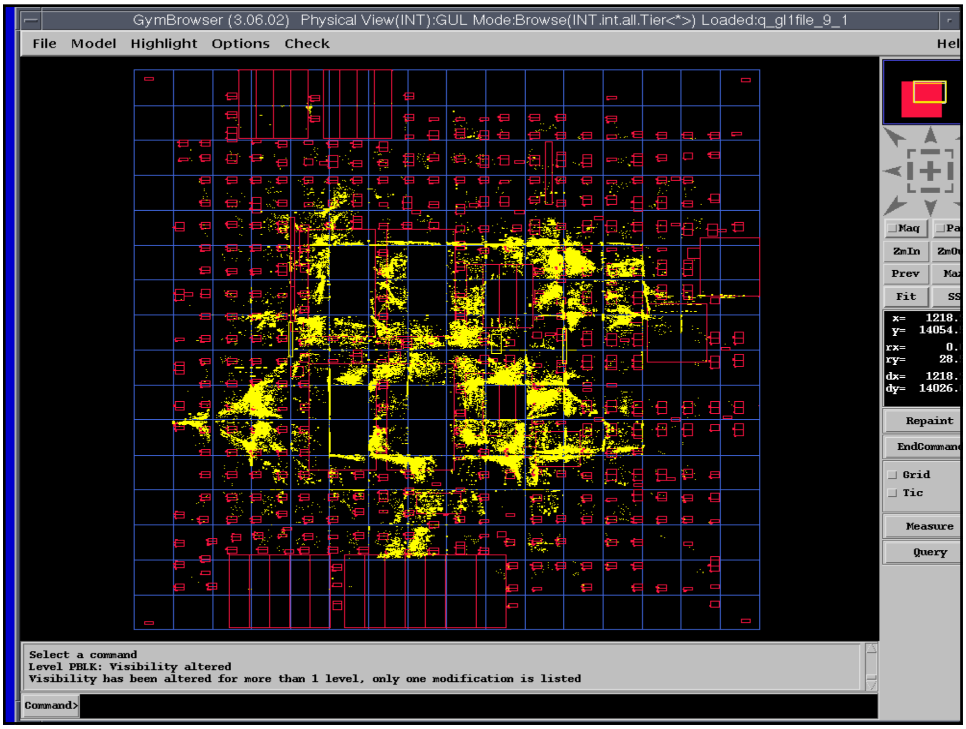


```
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Level CKTROW: Visibility altered
```



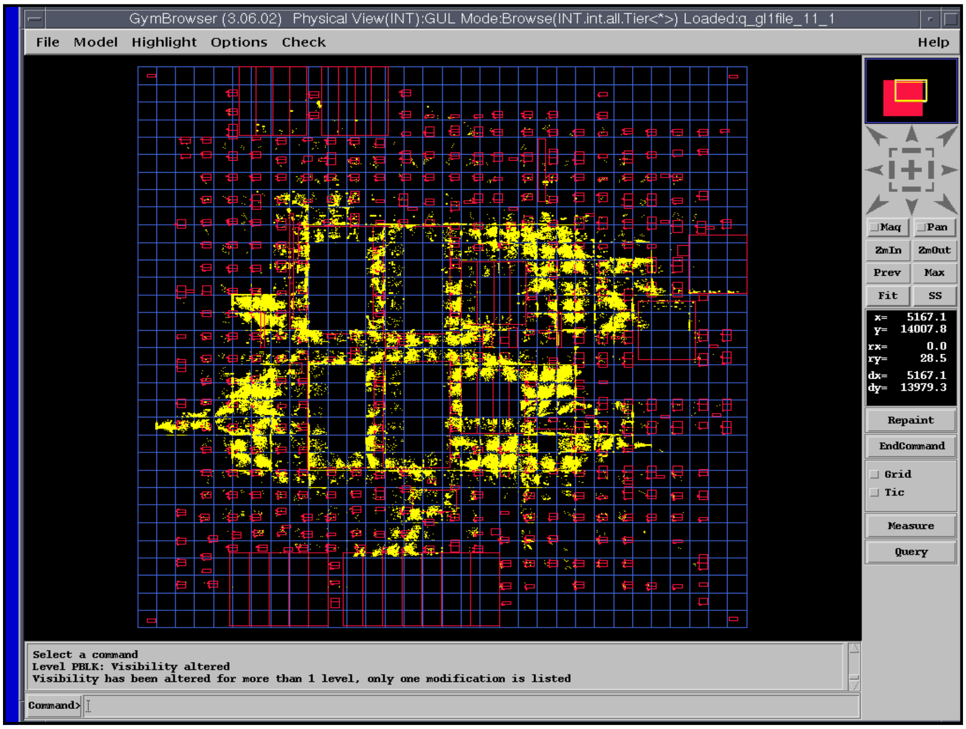
```
x=    3978.0
y=   14007.9
rx=      0.0
ry=     28.5
dx=   3978.0
dy=  13979.4
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```



```
x=    1218.
y=   14054.
rx=      0.
ry=     28.
dx=   1218.
dy=  14026.
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```

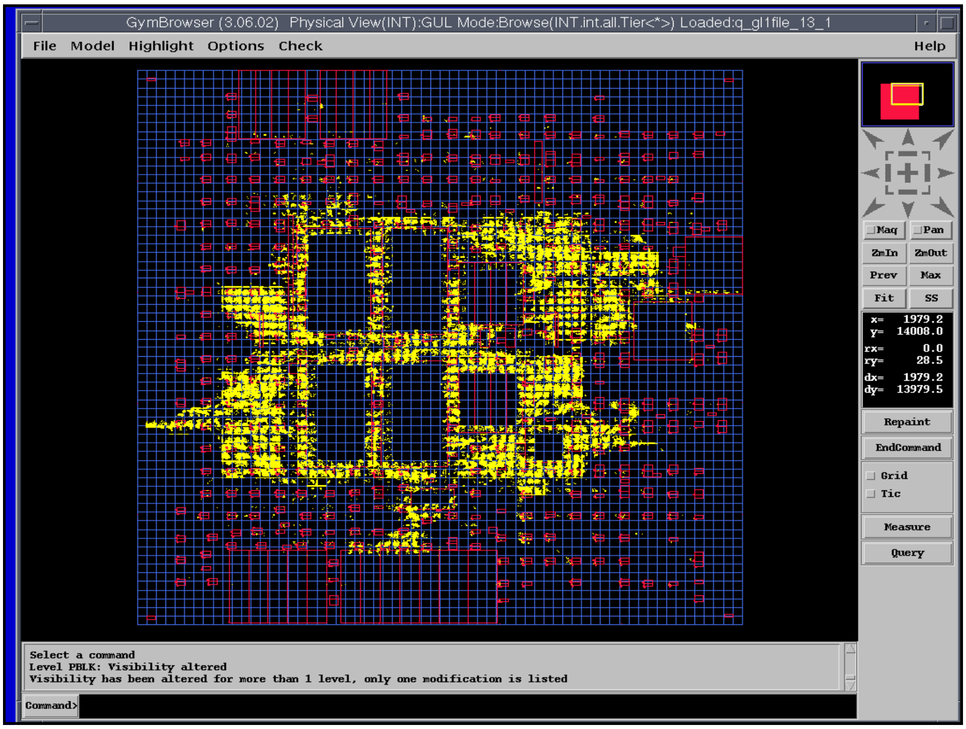


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:q_gl1file_11_1

```
x=   5167.1
y=  14007.8
rx=     0.0
ry=    28.5
dx=  5167.1
dy= 13979.3
```

```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```

Command>

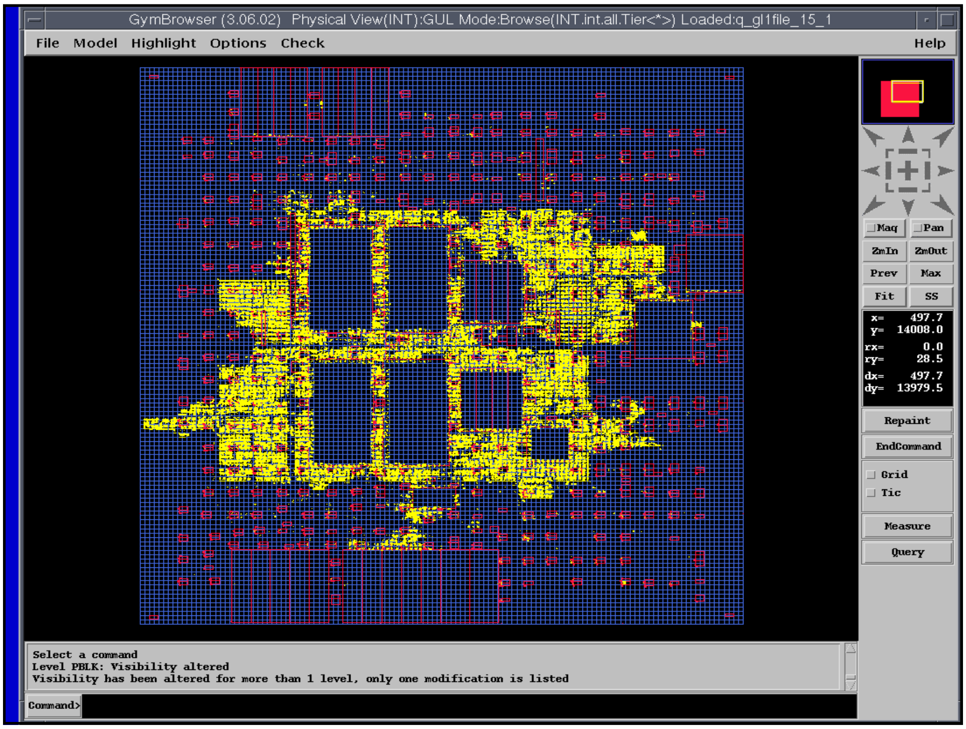


GymBrowser (3.06.02) Physical View(INT):GUL Mode:Browse(INT.int.all.Tier<*>) Loaded:q_gl1file_13_1

```
x=   1979.2
y=  14008.0
rx=     0.0
ry=    28.5
dx=  1979.2
dy= 13979.5
```

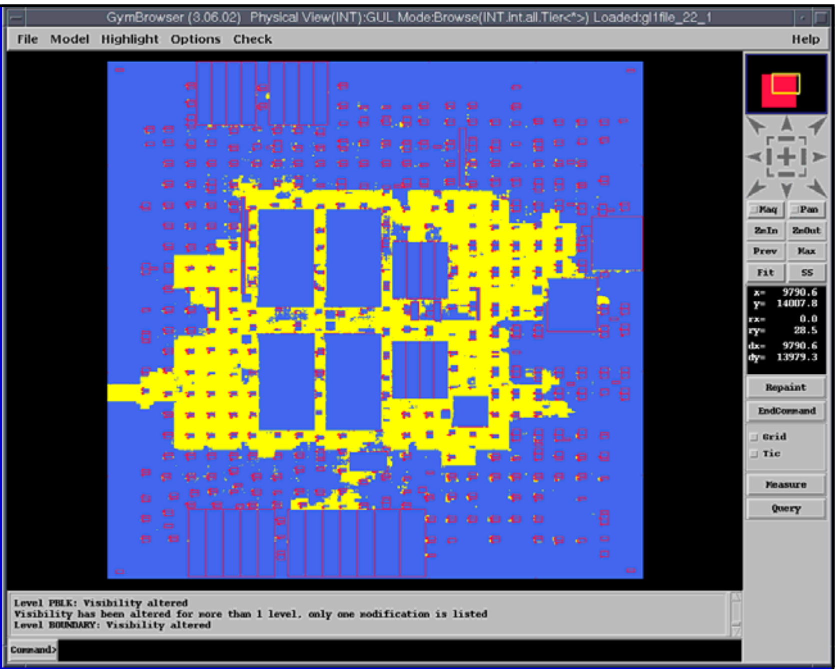
```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```

Command>



```
x=    497.7
y=  14008.0
rx=     0.0
ry=    28.5
dx=   497.7
dy=  13979.5
```

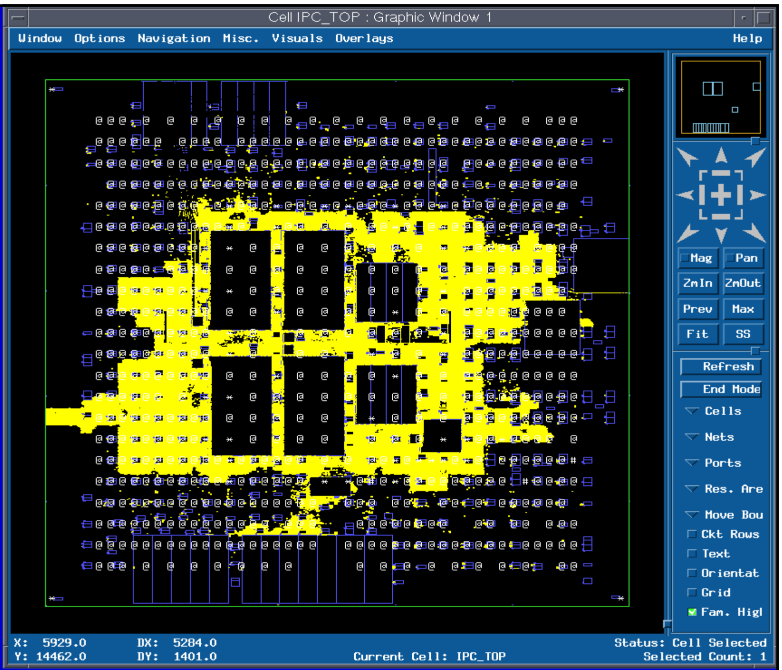
```
Select a command
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
```



```
x=   9790.6
y=  14007.8
rx=     0.0
ry=    28.5
dx=  9790.6
dy= 13979.3
```

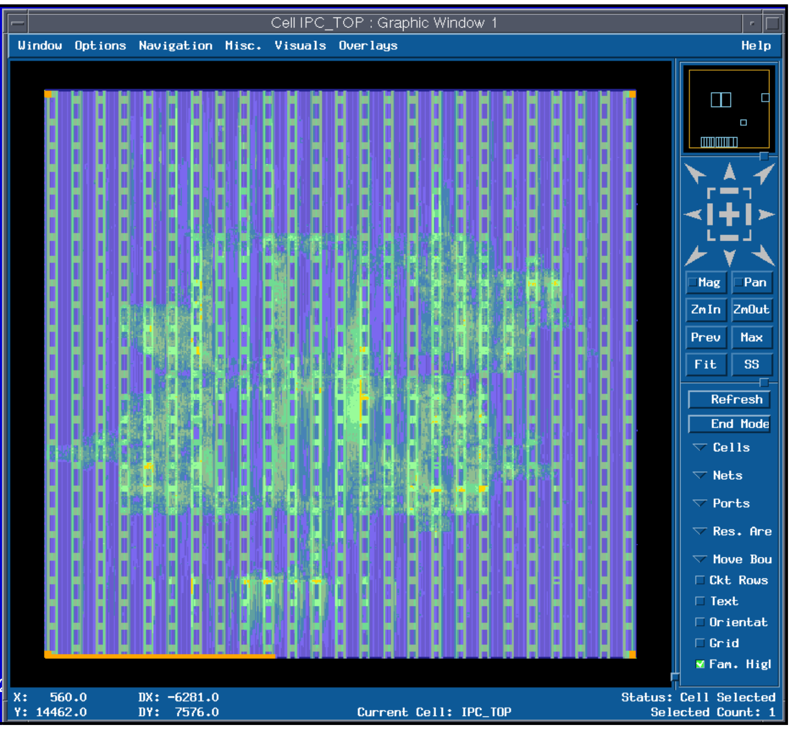
```
Level PBLK: Visibility altered
Visibility has been altered for more than 1 level, only one modification is listed
Level BOUNDARY: Visibility altered
```

MLP w/ACG



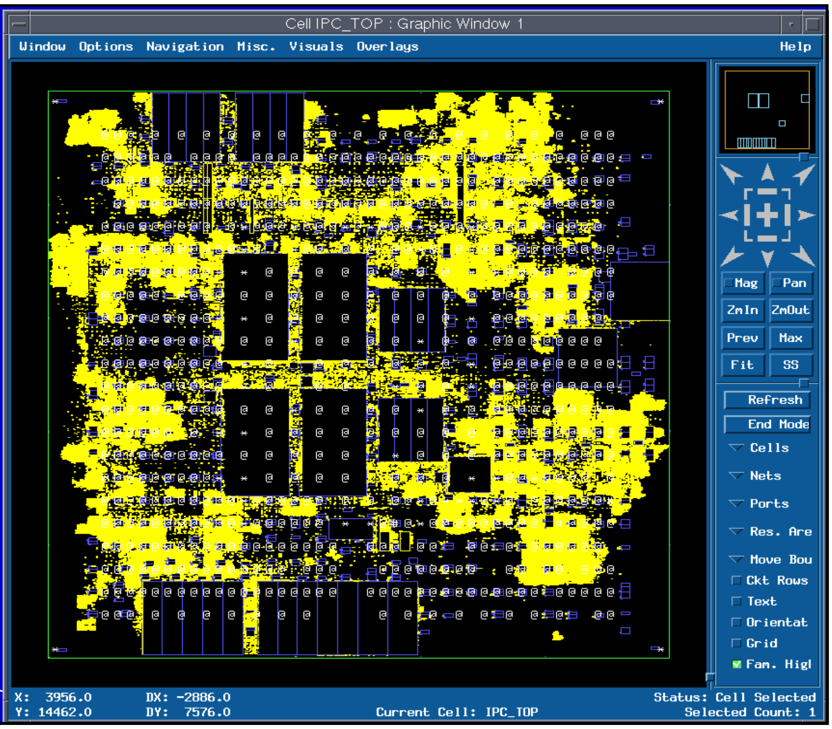
X: 5929.0 DX: 5284.0 Status: Cell Selected
Y: 14462.0 DY: 1401.0 Current Cell: IPC_TOP Selected Count: 1

Global Route Results:



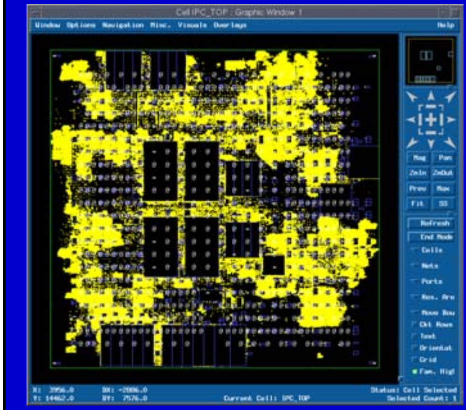
X: 560.0 DX: -6281.0 Status: Cell Selected
Y: 14462.0 DY: 7576.0 Current Cell: IPC_TOP Selected Count: 1

MLP
w/o
ACG

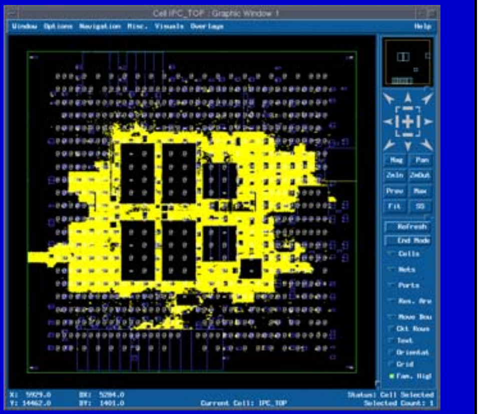


```
X:  3956.0      DX: -2886.0                                  Status: Cell Selected
Y: 14462.0      DY:  7576.0      Current Cell: IPC_TOP       Selected Count: 1
```

Side by Side Comparison:



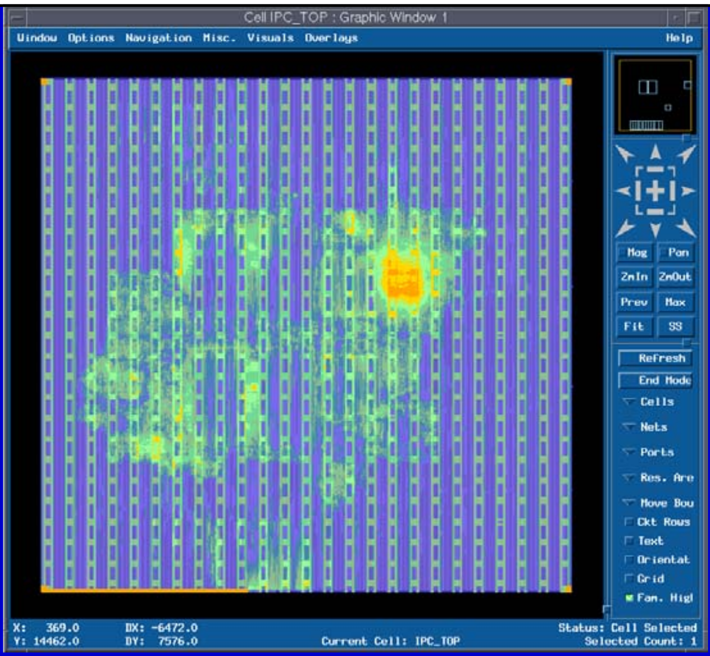
```
X:  3956.0      DX: -2886.0                                  Status: Cell Selected
Y: 14462.0      DY:  7576.0      Current Cell: IPC_TOP       Selected Count: 1
```



```
X:  5929.0      DX:  5204.0                                  Status: Cell Selected
Y: 14462.0      DY:  1401.0      Current Cell: IPC_TOP       Selected Count: 1
```

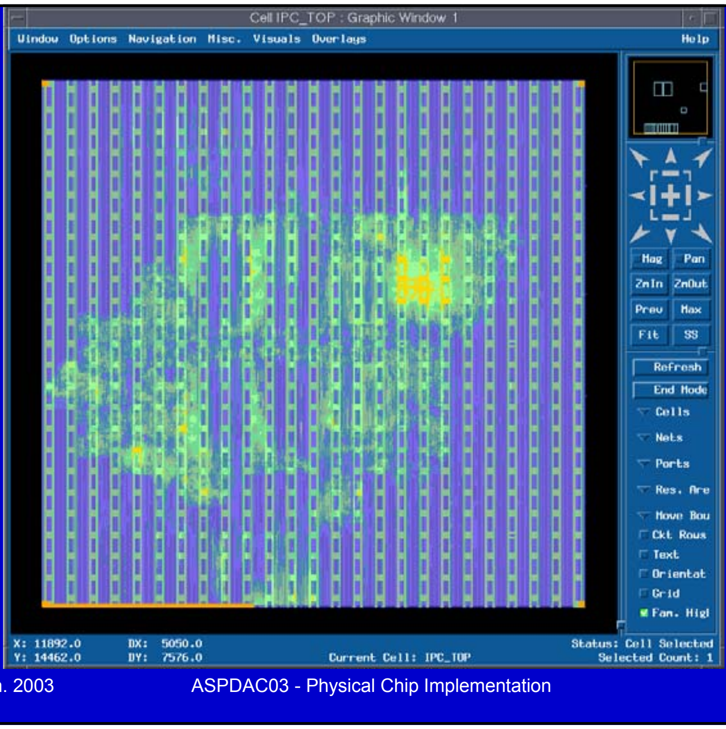
Original

ACG



X: 369.0 DX: -6472.0
Y: 14462.0 DY: 7576.0 Current Cell: IPC_TOP

Status: Cell Selected
Selected Count: 1



X: 11892.0 DX: 5050.0
Y: 14462.0 DY: 7576.0 Current Cell: IPC_TOP

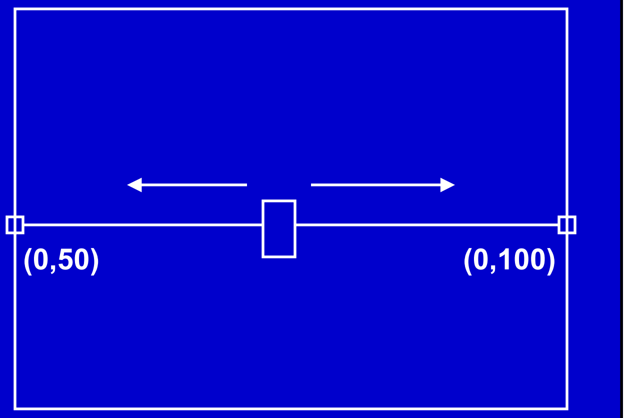
Status: Cell Selected
Selected Count: 1

Observations on Quadratic Placement

- placements are predictable and repeatable
- timing is inherently better
- wire length is not the best, but good
- run time: slower than MLP by 4x
- run time: faster than annealing by 4x
- excellent "free space" handling
- placements "feel" similar to those produced by annealing

Repeatability Example:

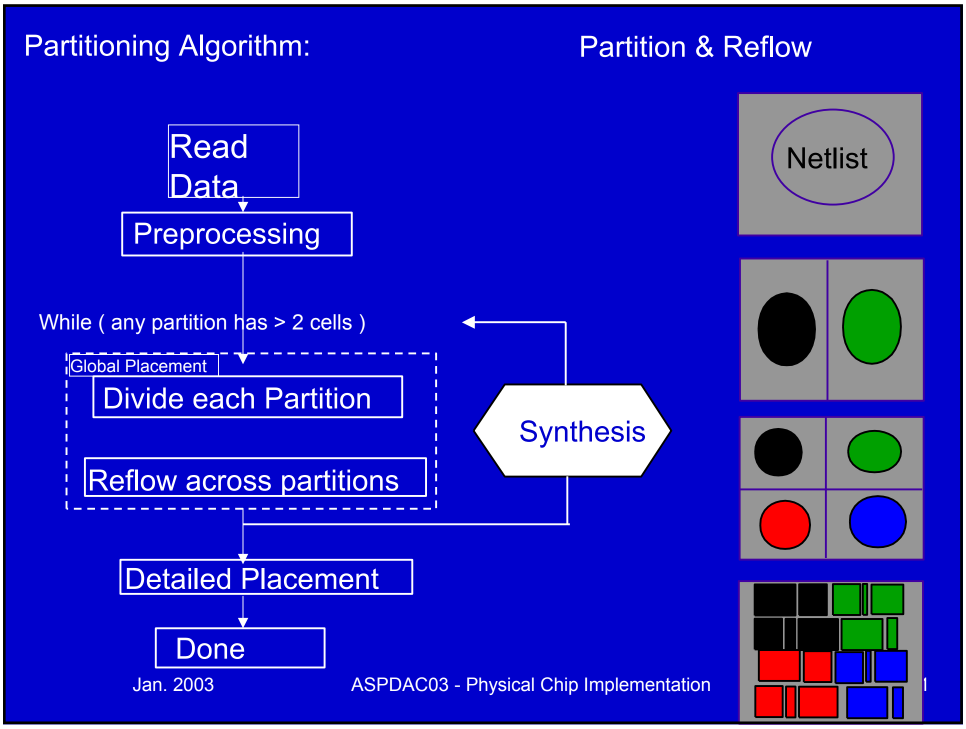
- One circuit
- Minimum linear length occurs for all solutions where y=50 0 < x < 100
- Minimum quadratic length occurs for y=50, x=50
- Quadratic solution IS both minimum linear and minimum quadratic length



Section Outline

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing optimization

Synthesis - Placement Interface



Partitioning Algorithm:

Partition & Reflow

Read Data

Preprocessing

While (any partition has > 2 cells)

Global Placement

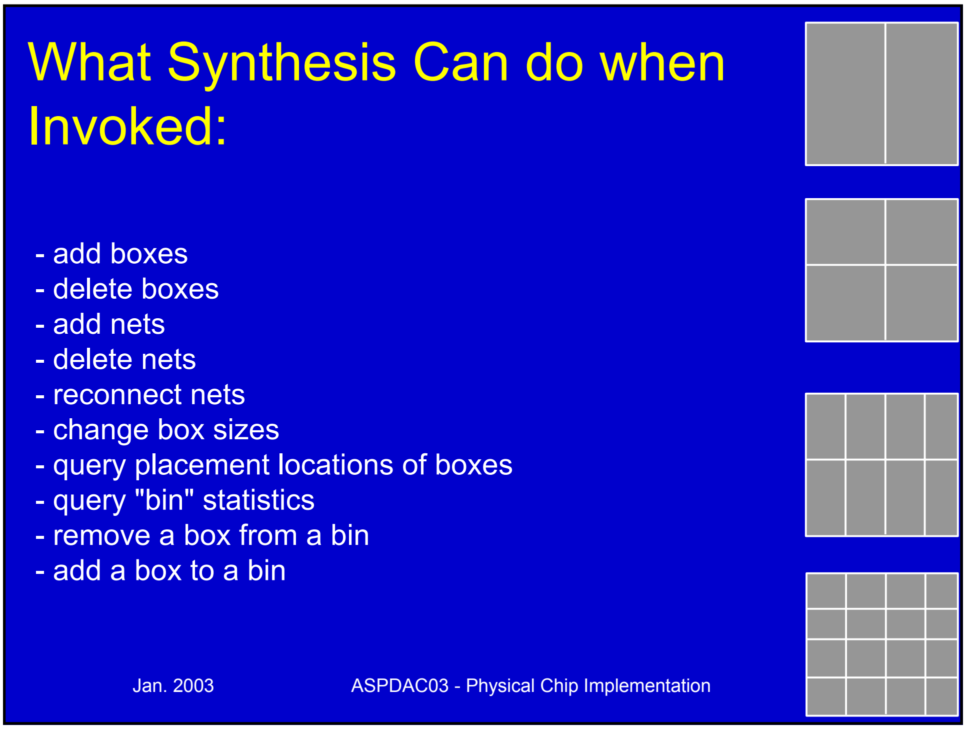
Divide each Partition

Reflow across partitions

Synthesis

Detailed Placement

Done



What Synthesis Can do when Invoked:

- add boxes
- delete boxes
- add nets
- delete nets
- reconnect nets
- change box sizes
- query placement locations of boxes
- query "bin" statistics
- remove a box from a bin
- add a box to a bin

Placement and Synthesis Integration

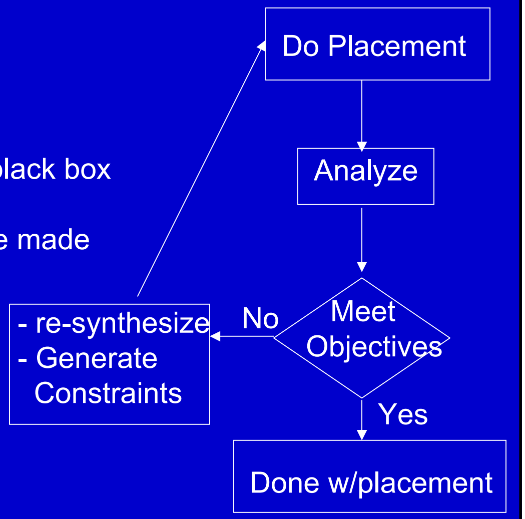
- Loosely coupled: (methodology coupling)
  - do some synthesis, then write out data
  - do some placement, then write out data
  - .. Repeat
- Interleaved: (placement & synthesis in same process)
  - do pre-pd synthesis
    - for each placement step redo synthesis
- Tightly coupled: (simultaneous P&S aware transforms)

Loosely Coupled Placement & Synthesis:

Characteristics:

- Placement is treated as a black box

- Multiple placement runs are made

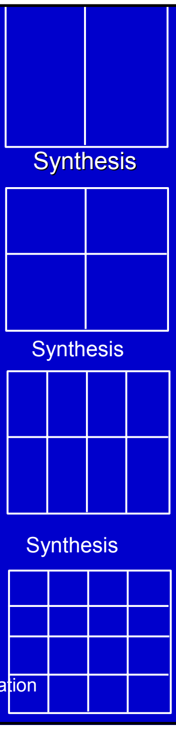


Interleaved Placement & Synthesis:

Characteristics:

- the placement flow is the same as in a placement only methodology

- in between each step of the placement progression, synthesis is invoked



Tightly Coupled

- Placement and synthesis algorithms become co-dependent
- Placement algorithms have awareness of synthesis activity
- Synthesis algorithms have awareness of placement activity

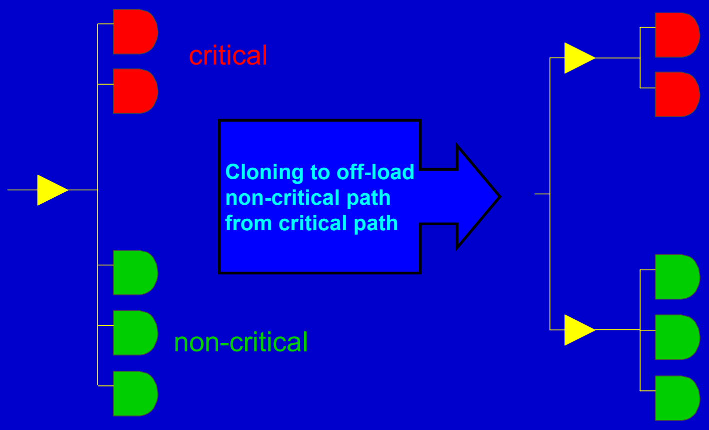
Section Outline

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing optimization

Summary of Techniques

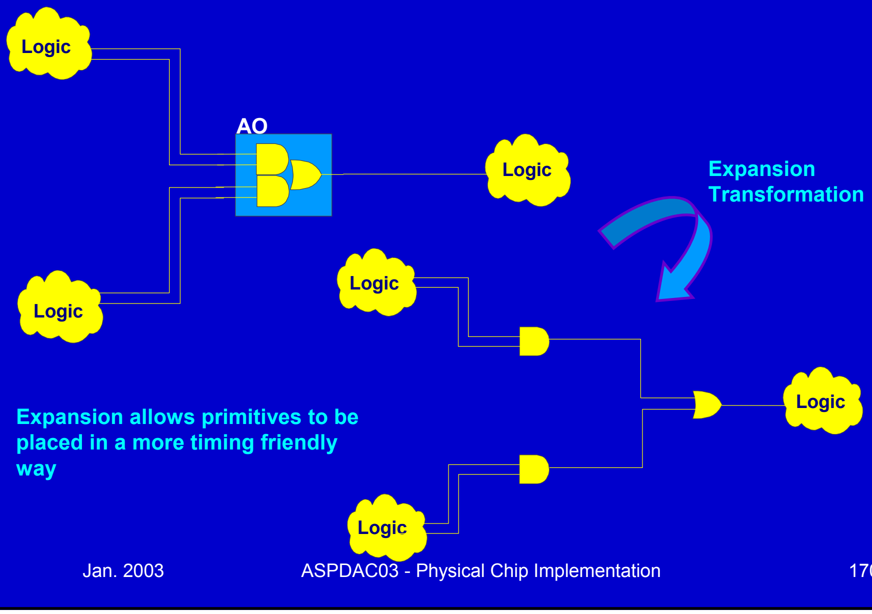
- Placement-Driven X (PDX)
  - Cloning, Spreading, Sizing, Fanout Reclustering, …
  - (Constant-Delay Methodology)
- Buffer Insertion
  - Key problem: Max RAT at Source Interconnect Tree synthesis
  - Heuristic search over topologies + VanGinneken dynamic programming (with practical limits on polarity, buffer location, buffer library, etc. richness of formulation)
  - C-Tree (IBM), recent Q-Tree (UCSD), P/S/U-Tree (UIC), etc.
  - Early timing analysis (slew rate, cap load control): UCSD+IBM
- Buffer Block Planning + Buffered Global Routing
  - DAC-2001 Best Paper from IBM (buffer bays)
  - ASPDAC-2002 Best Paper from UCSD (delay-bounded floorplan evaluation with given buffer plan)
  - Primal-Dual Multi-Commodity Flow approximation

Placement Driven Cloning



Cloning to off-load non-critical path from critical path

Placement Driven Expansion



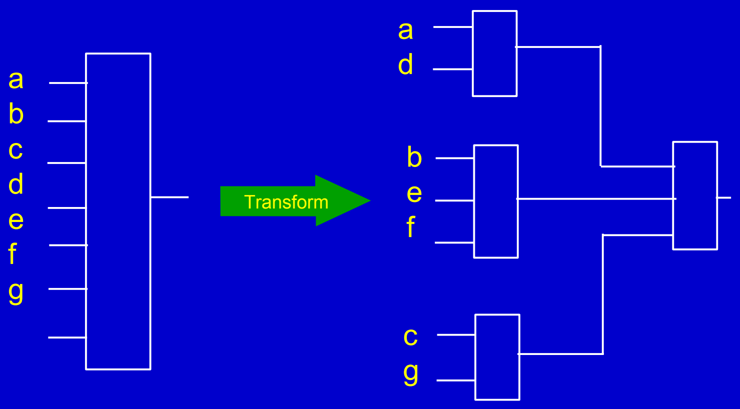
Expansion Transformation

Expansion allows primitives to be placed in a more timing friendly way

Example:

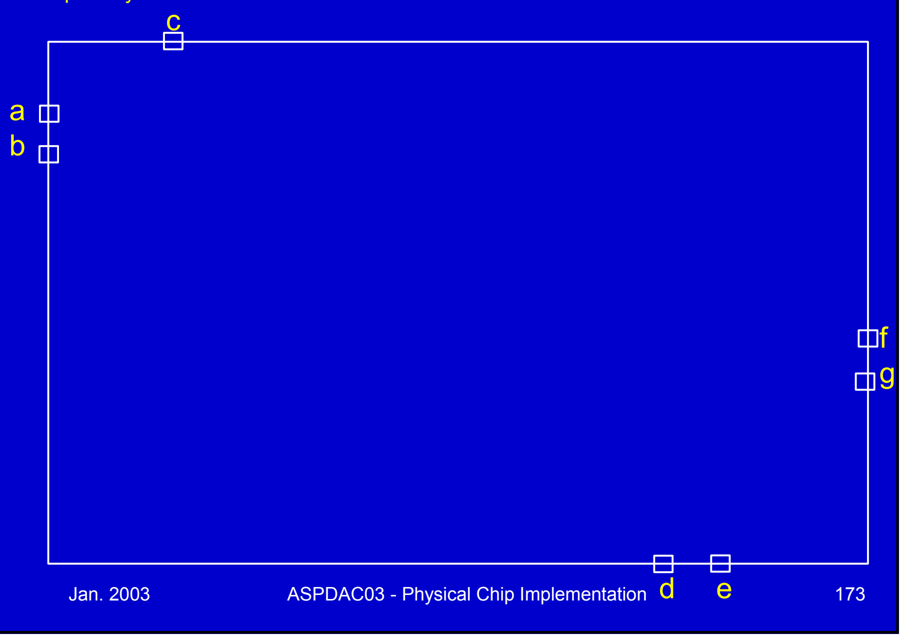
Tightly Coupled Placement Driven Expansion

Tightly Coupled Synthesis & Placement:



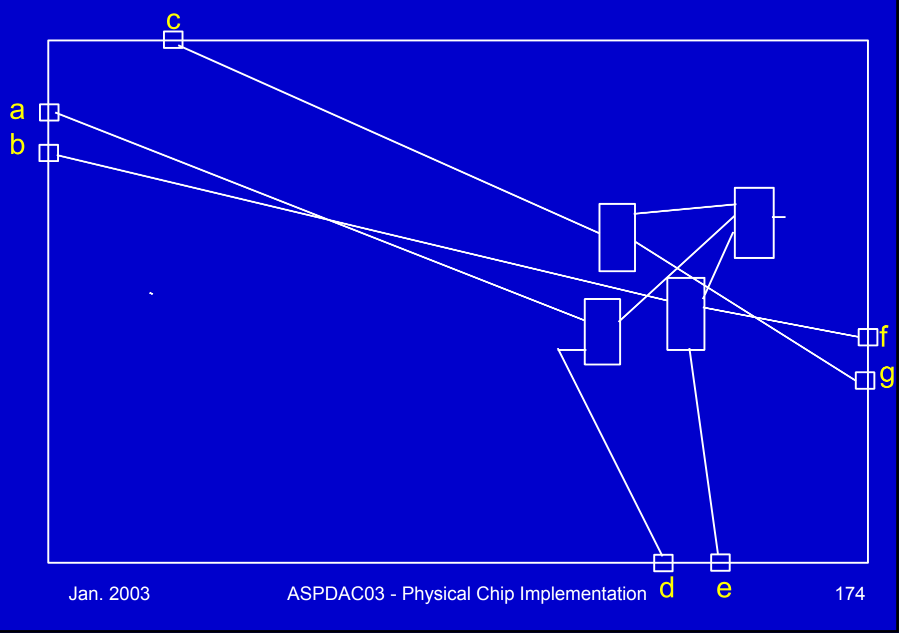
Tightly Coupled Synthesis & Placement Example:

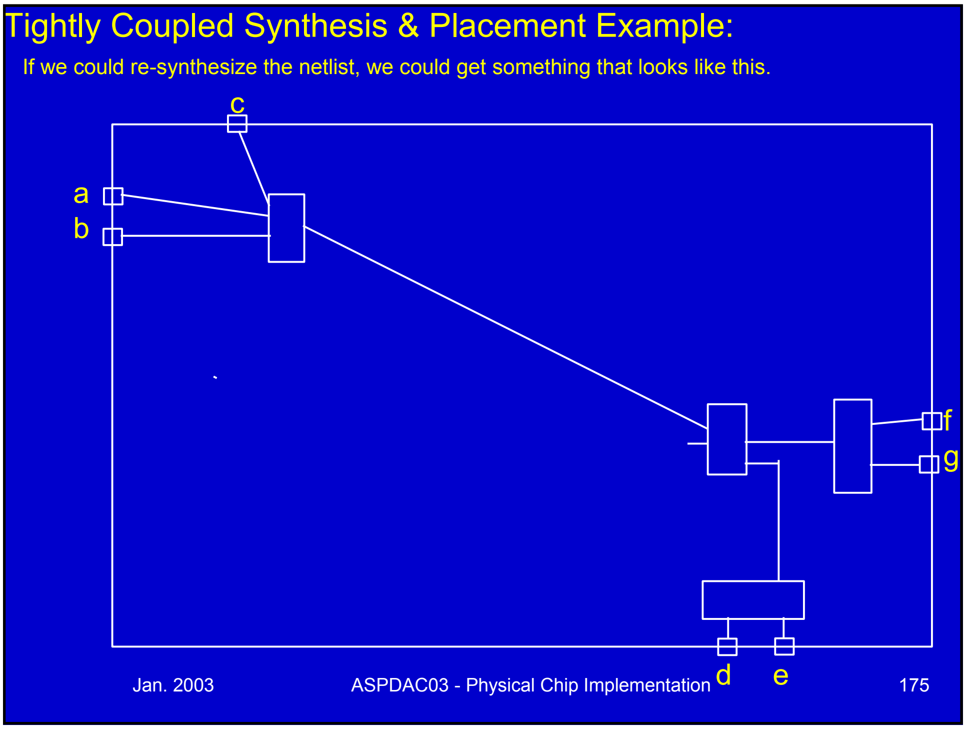
Suppose the primary IO constraints look like this:



Tightly Coupled Synthesis & Placement Example:

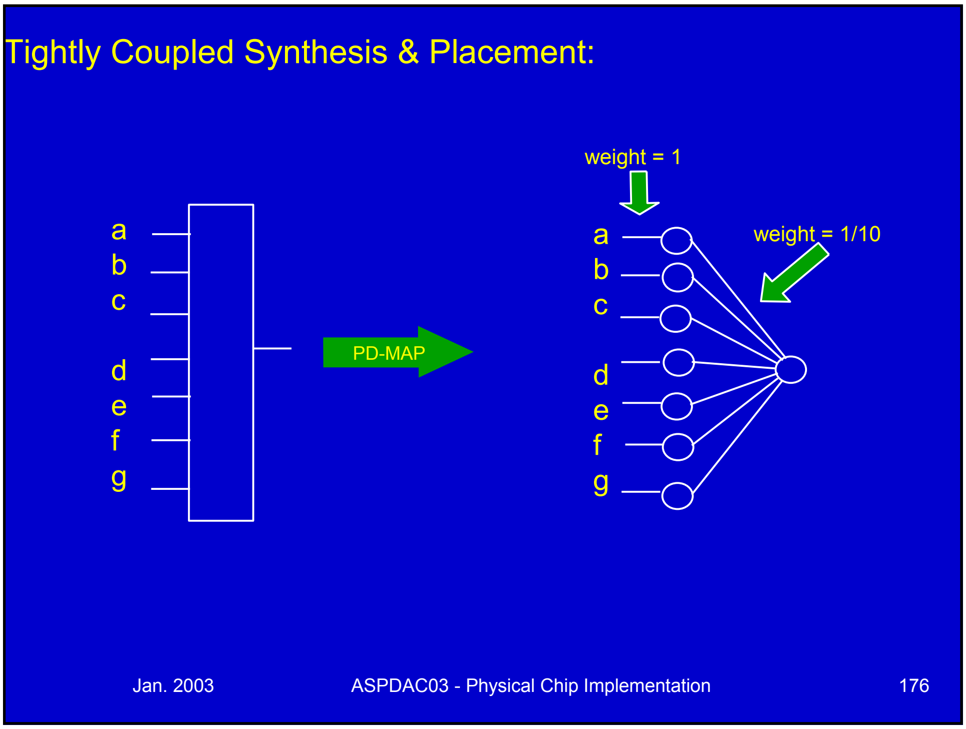
The placement of the synthesized netlist would look something like this:





Tightly Coupled Synthesis & Placement Example:

If we could re-synthesize the netlist, we could get something that looks like this.

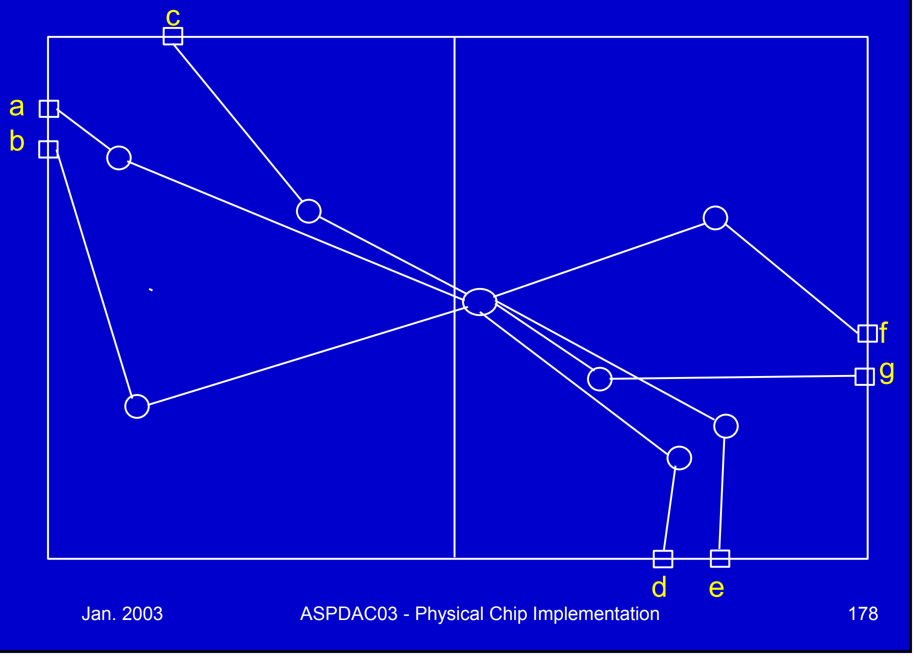


Tightly Coupled Synthesis & Placement:

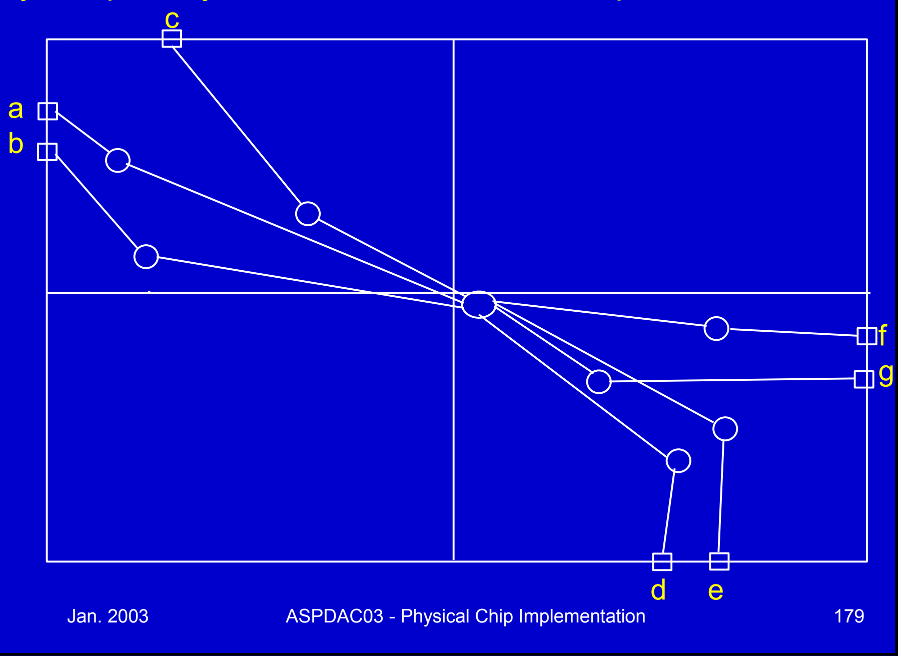
Tightly Coupled Synthesis & Placement example:

```
Map_TREE
For each cut
     partition_it
     For each partition
      If(partition number > M)
      {
          if(related_node_count < N)
               merge_nodes
          if(related_node_count == 1)
               merge_node into neighbor
           partition
       }
     end
end
```

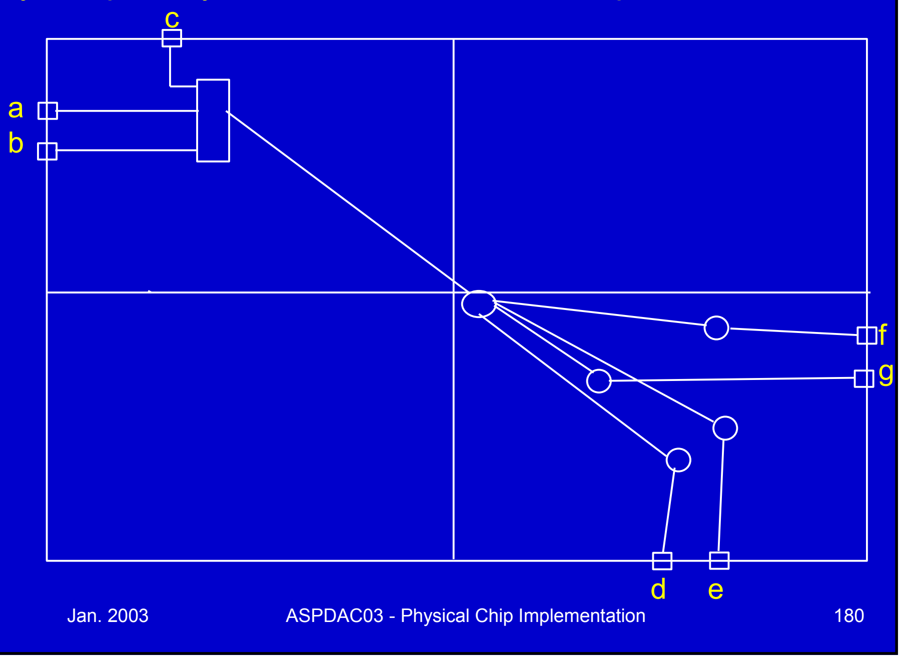
Tightly Coupled Synthesis & Placement Example:

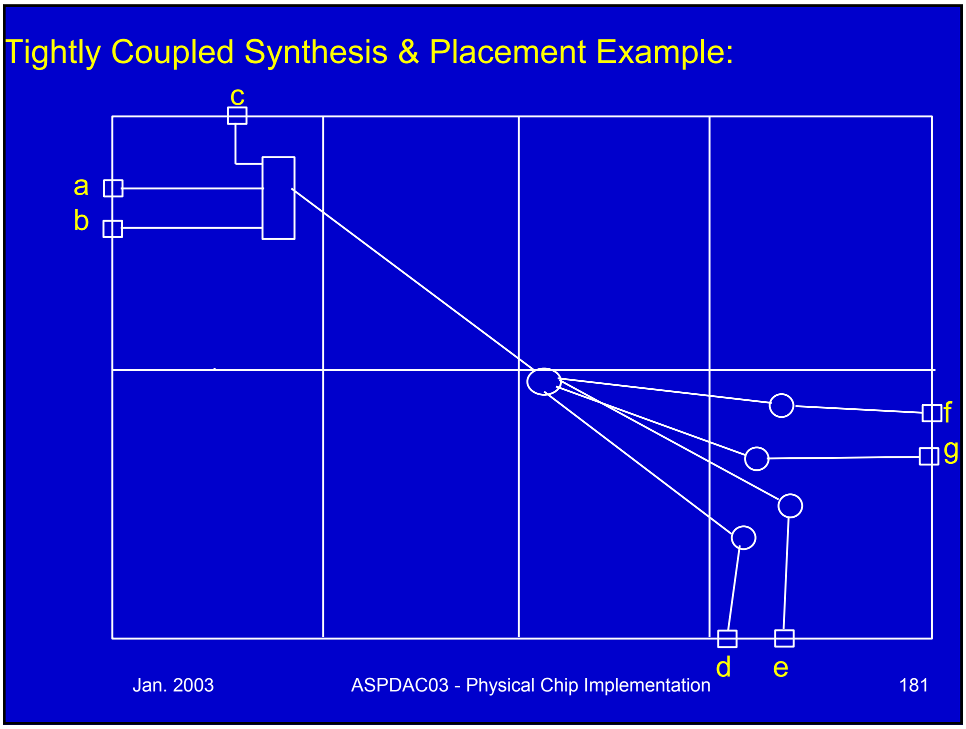


Tightly Coupled Synthesis & Placement Example:

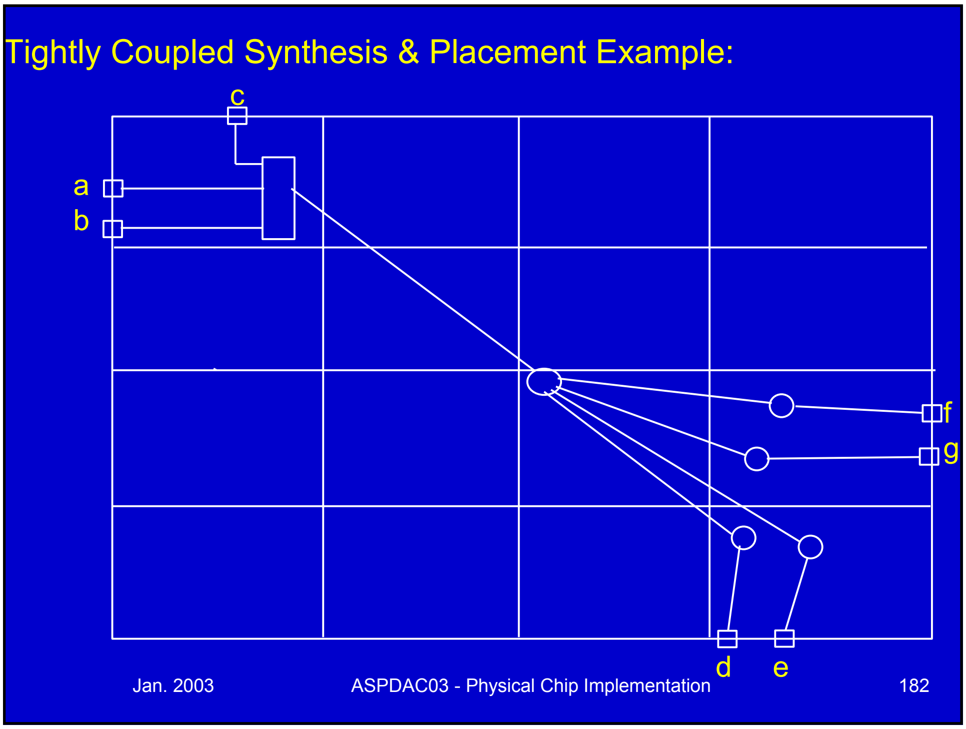


Tightly Coupled Synthesis & Placement Example:

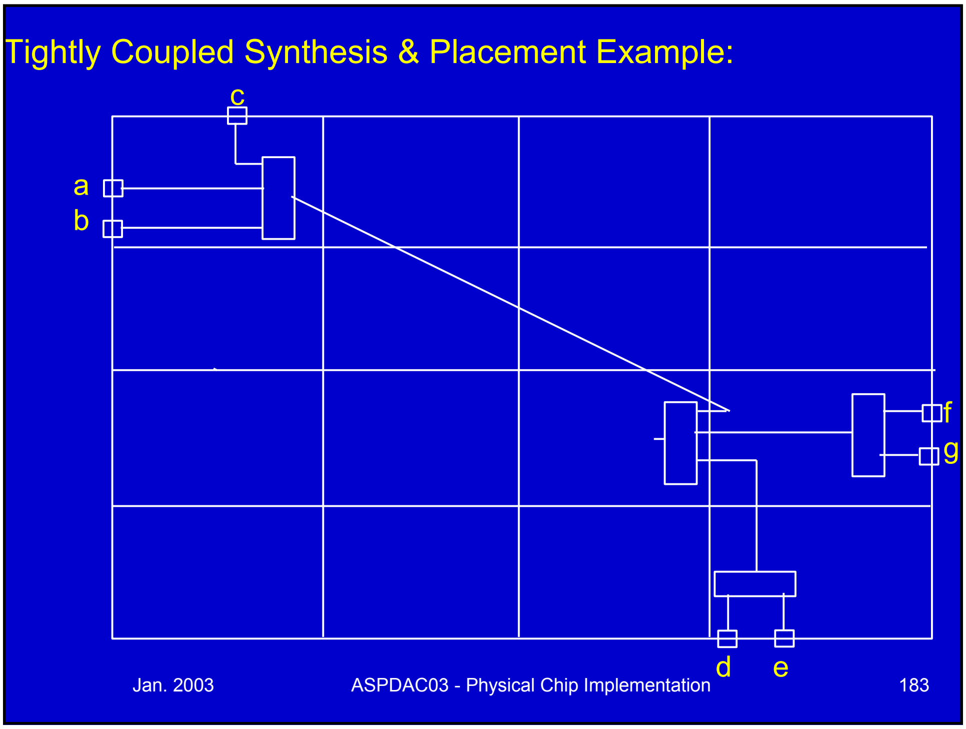




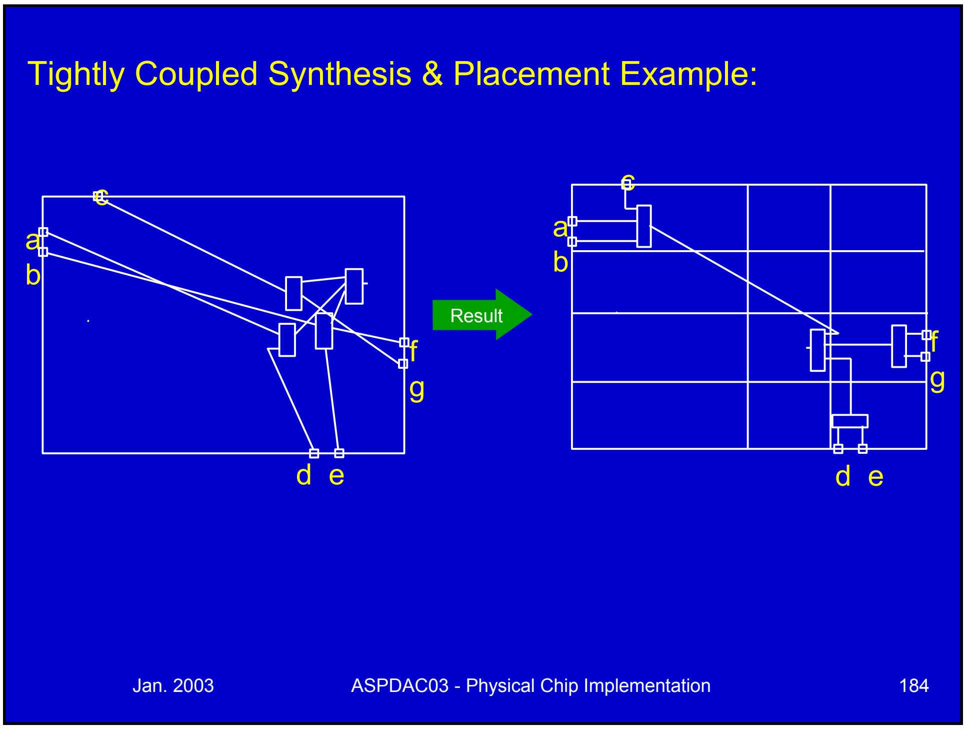
Tightly Coupled Synthesis & Placement Example:



Tightly Coupled Synthesis & Placement Example:

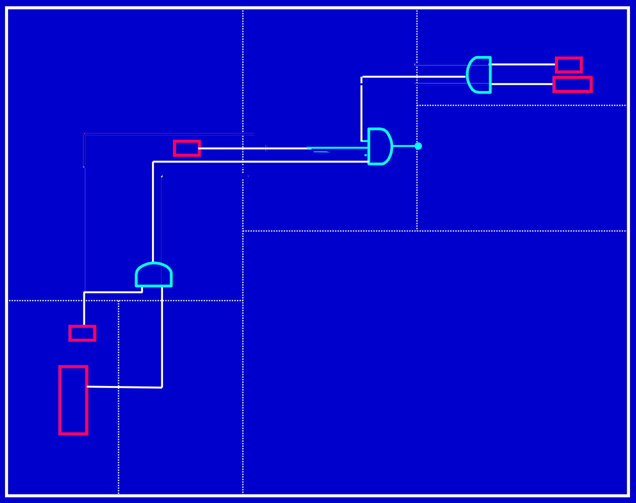


Tightly Coupled Synthesis & Placement Example:

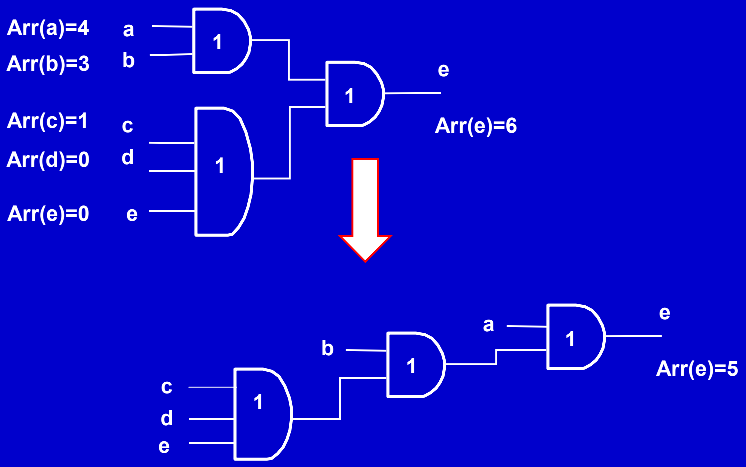


Tightly Coupled Synthesis & Placement Example:

Placement Driven Timing Correction



Redesign Fan-in Tree

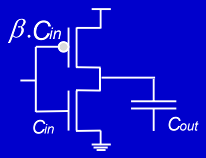


Placement Driven Repowering

- Repowering is traditionally done using load based cell characterization
- Placement changes continuously during partitioning
- Need high efficiency algorithms to do repowering in this environment
- Solution: Use Gain Based Formulation

Delay Models

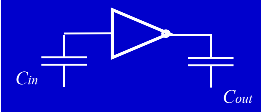
Load based formulation:



$$d = \underbrace{c_1.E_{inv.}\frac{C_{out}}{(1+\beta).C_{in}}}_{k_1} + p$$

$$d = k_1.C_{out} + p$$

Gain based formulation:



$$g = \frac{C_{out}}{C_{in}}$$

$$d = \underbrace{c_1.E_{inv.}}_{l}\frac{C_{out}}{(1+\beta).C_{in}} + p$$

$$d = l.g + p$$

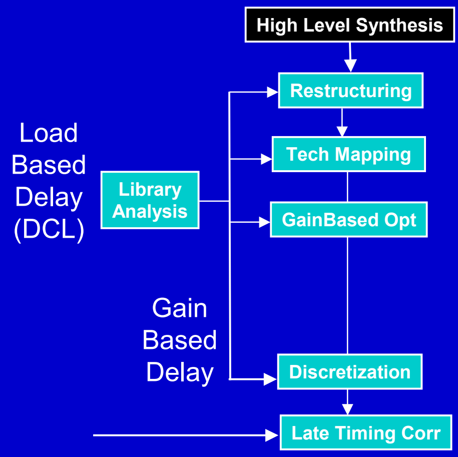
d: delay
l: logical effort
g: gain
p: intrinsic delay

Area vs Delay Centric

- Load Based Paradigm
  - (load-based delay eq.)
  - sized
- Know:
  - Size of each cell
  - Total Area ->
    - area centric
- Don't know:
  - Wire loads
  - Delay of each cell
  - Delay of a path
- Estimation error is in the delay:
  - Local 'path based' property.

- Gain Based Paradigm
  - (gain based delay eq.)
  - sizeless
- Know:
  - The delay of each cell.
  - The delay of a path ->
    - delay centric
- Don't know:
  - Wire loads
  - The area of each cell
  - The total area
- Estimation error is in the area
  - Global property.

Design Flow



High Level Synthesis

Restructuring

Tech Mapping

GainBased Opt

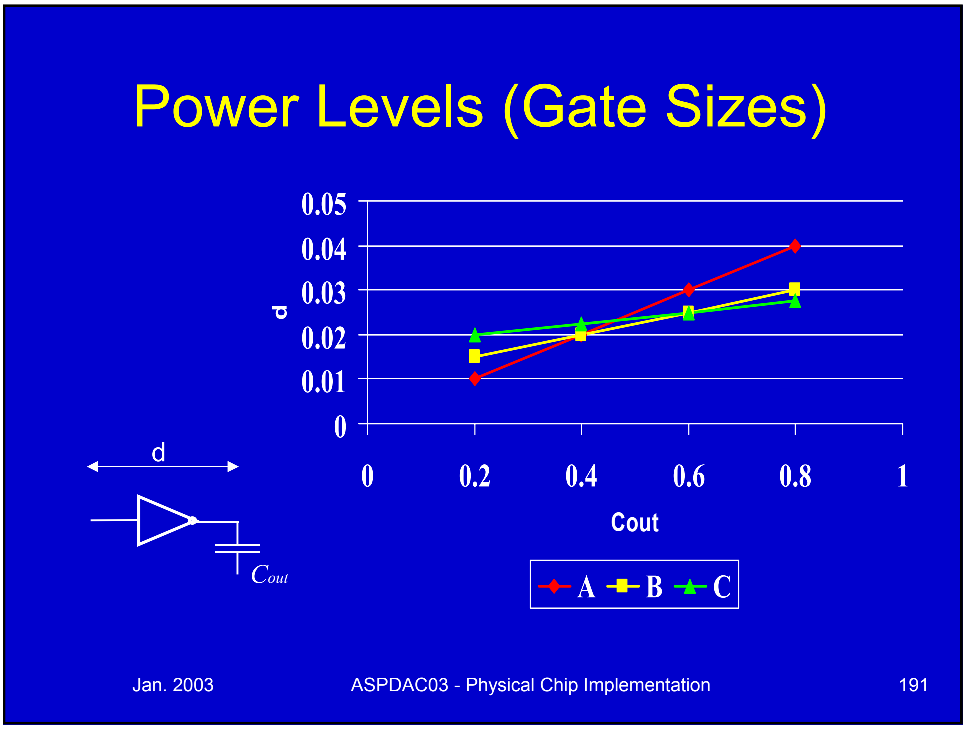
Discretization

Late Timing Corr

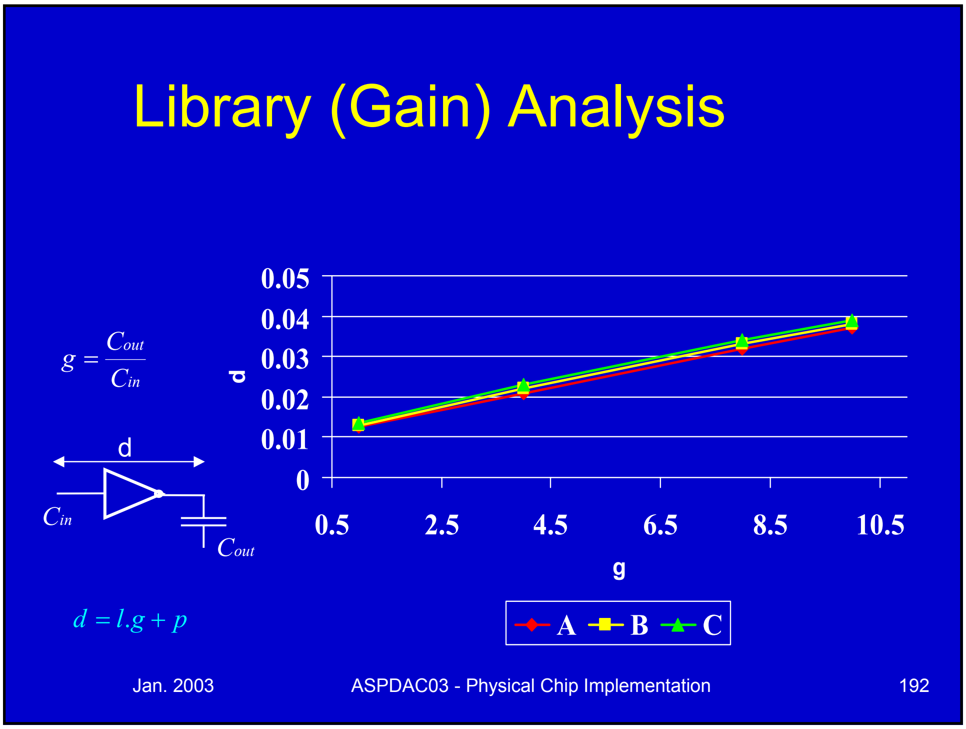
Load Based Delay (DCL)

Library Analysis

Gain Based Delay



Power Levels (Gate Sizes)

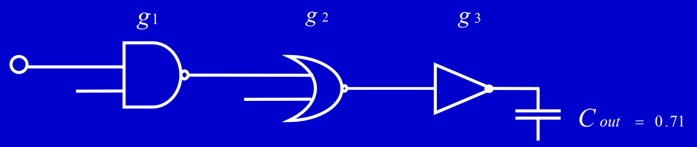


Library (Gain) Analysis

$$g = \frac{C_{out}}{C_{in}}$$

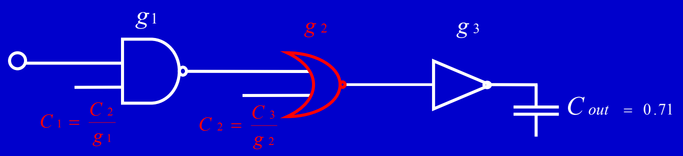
$$d = l.g + p$$

Area and Load Calculation

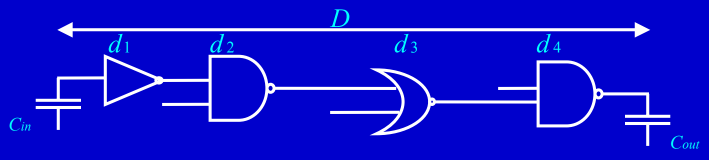


$C_1 = \frac{C_2}{g_1}$ $C_2 = \frac{C_3}{g_2}$ $C_3 = \frac{C_{out}}{g_3}$

- Start at primary outputs/ register inputs.
- Much like static timing analysis.
- Incremental.



Gain Calculation



$$G = \prod_{i=1}^{N} g_i = g_1.g_2.g_3.g_4. = \frac{C_2}{C_{in}}.\frac{C_3}{C_2}\frac{C_4}{C_3}\frac{C_{out}}{C_4} = \frac{C_{out}}{C_{in}}$$

$$D = \sum_{i=1}^{N} d_i = d_1 + d_2 + d_3 + d_4$$

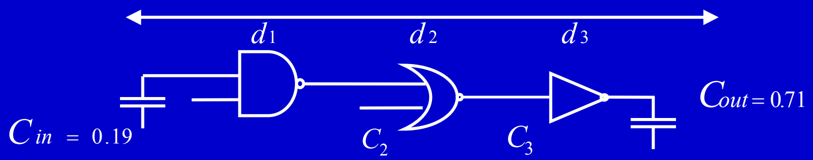
Minimize D such that: $G = \frac{C_{out}}{C_{in}}$

$$d = l.g + p = f + p$$

Minimize $D = \sum_{i=1}^{N} f_i + \sum_{i=1}^{N} p_i$ Such that: $F = \prod_{i=1}^{N} f_i = \prod_{i=1}^{N} l_i . \prod_{i=1}^{N} g_i = L.\frac{Cout}{Cin}$

Geometric Solution: $f_i = f$

Example I



$$F = f_1.f_2.f_3 = f^3 = L\frac{C_{out}}{C_{in}} = 0.000008$$

$$f = 0.0203$$

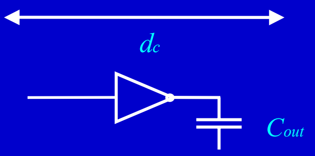
NAND2: $d = 0.0308 + 0.011 \times \frac{C_2}{C_{in}}$

NOR2: $d = 0.0496 + 0.021 \times \frac{C_3}{C_2}$

INV : $d = 0.0295 + 0.009 \times \frac{C_{out}}{C_3}$

	Nand2	Nor2	Inv	Path
p	0.0308	0.0496	0.0295	0.1099
f	0.0203	0.0203	0.0203	0.0609
d	0.0511	0.0699	0.0498	0.1708
Cin	0.19	0.3364	0.3283	.

Constant Delay Calculation



Inverter:

$d = l.g + p$

Set: $g_c = 3.6$

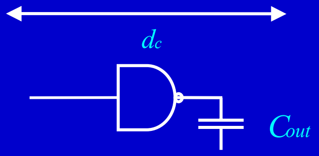
Calculate: $Cout = \frac{g_c}{Cin}$

Measure: d_c

Set: $Cout = 0$

Measure: $d_o = p$

Calculate: $l.g_c = d_c - p = f_c$



Other gates :

$l.g = f_c$

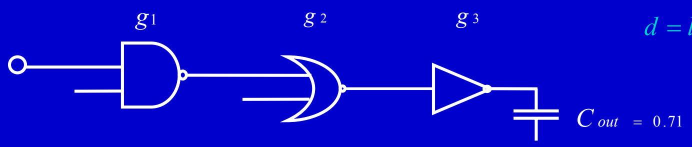
$g_{nand} = 2.5$

$g_{nor} = 1.8$

$d_{c.nand} = l_{.nand}.g_{nand} + p_{nand}$

Discretization

- From gain-based model back to appropriate power levels
- There is an error in timing/load when ‘ideal’ power levels are not available.
  - Goal: Minimize this error.
  - Can be tuned to delay error or capacitance error.



$d = l.g + p$

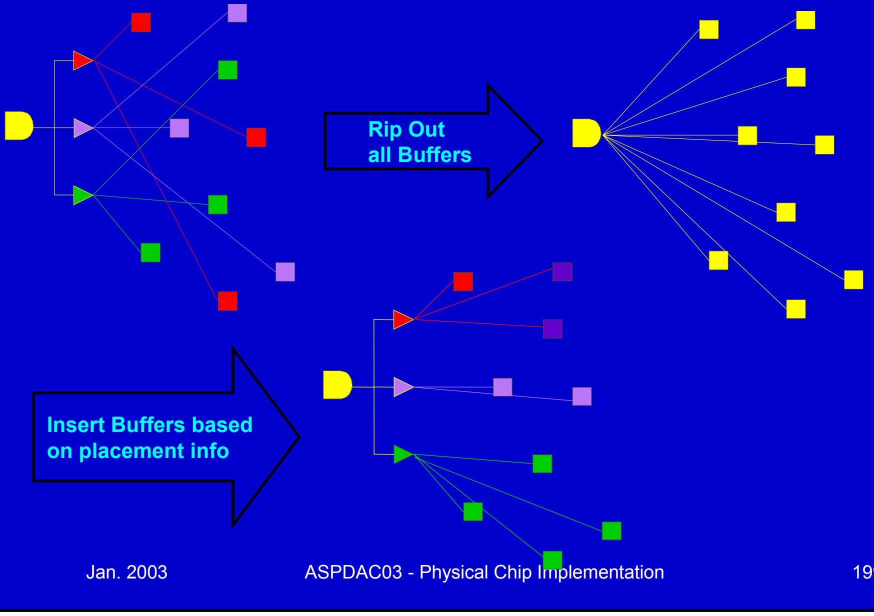
$C_1 = \frac{C_2}{g_1}$ $C_2 = \frac{C_3}{g_2}$ $C_3 = \frac{C_{out}}{g_3}$

[Kudva98]
[Beeftink98]

Gain Based: Observations:

- Gain Based algorithms: A major improvement.
  - More homogeneous (global) algorithms and designs.
  - Can be better targeted for area and/or delay.
- Reveal inherent cell characteristics to optimization tools, leading to improved QOR
- Good library design is required to facilitate discretization step
- Ideally suited for operation within Physical Synthesis

Placement Driven Buffering



What to do About Long Wires?

- Add buffers
- Tune wire sizes
- Modify the placement to reduce them

Placement Driven vs Logic Driven Buffer Insertion

- Logic driven buffer insertion focuses on logic topology and buffer sizing while assuming a statistical wire load model
- Placement driven buffering uses an existing placement as the fundamental constraint

Placement Driven Buffer Insertion: Buffopt (IBM)

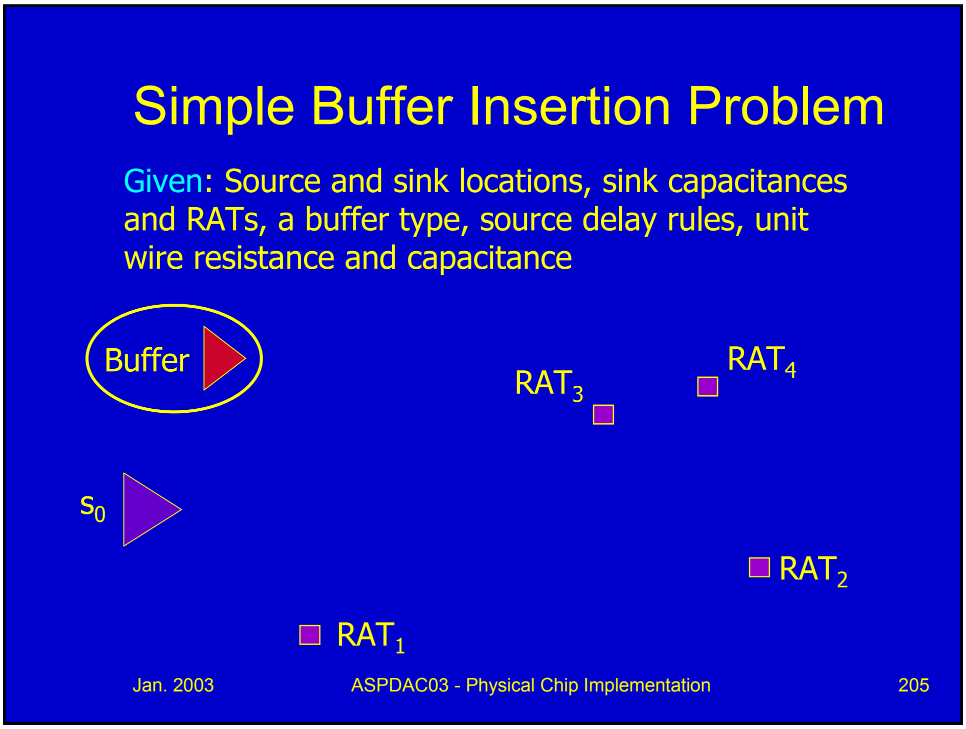
- Multiple buffer types
- Inverters
- Capacitance, Slew and Noise constraints
- Wire Sizing
- Simultaneous driver sizing
- High order interconnect delay and Ceffective
- Blockage handling

How Do Buffers Help?

- Reduce delay
  - Wire delay quadratic in length
  - Buffers make delay essentially linear
  - Delay gate dominated, not wire dominated
- Fix other problems
  - Bad slews at sinks
  - Capacitance range violations
  - Noise induced by capacitance coupling

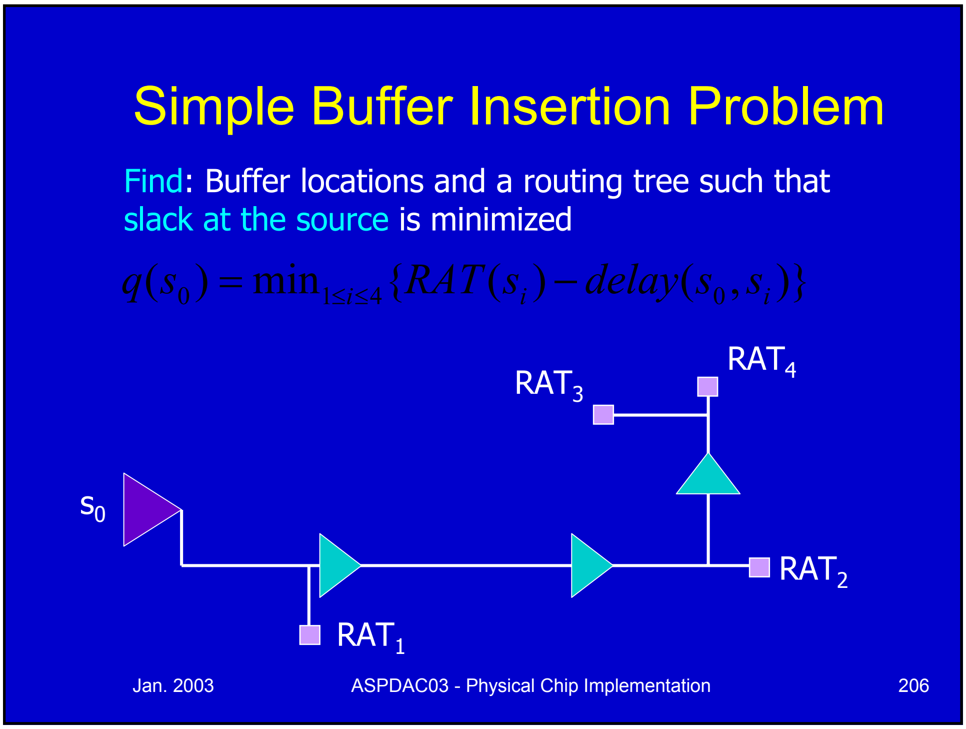
How Does Wire Sizing Help?

- Highly resistive lines increase delay
- Wider wires or thick metal layers reduces resistance, but can increase capacitance
- For long interconnect, resistance reduction outweighs capacitance increase



Simple Buffer Insertion Problem

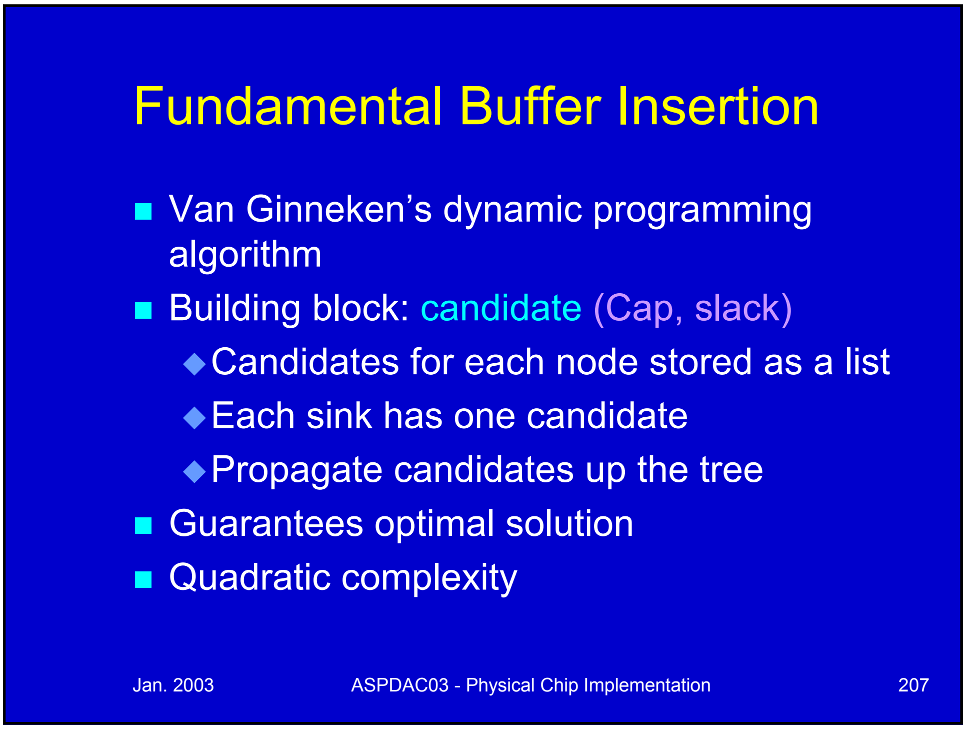
Given: Source and sink locations, sink capacitances and RATs, a buffer type, source delay rules, unit wire resistance and capacitance



Simple Buffer Insertion Problem

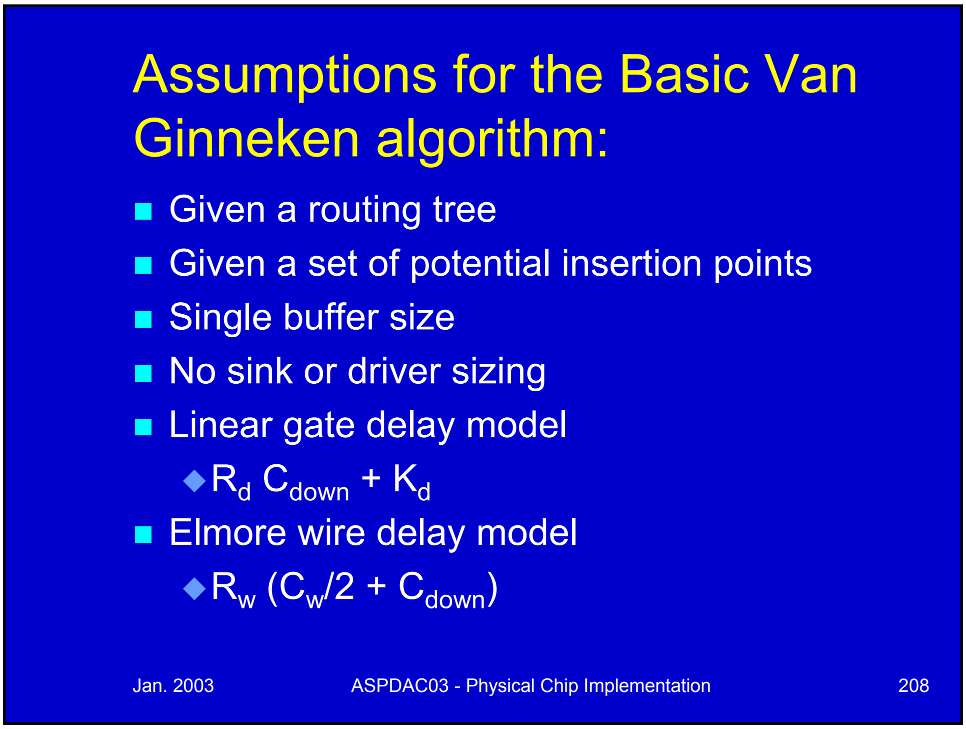
Find: Buffer locations and a routing tree such that slack at the source is minimized

$$q(s_0) = \min_{1\le i\le 4}\{RAT(s_i) - delay(s_0, s_i)\}$$



Fundamental Buffer Insertion

- Van Ginneken’s dynamic programming algorithm
- Building block: candidate (Cap, slack)
  - Candidates for each node stored as a list
  - Each sink has one candidate
  - Propagate candidates up the tree
- Guarantees optimal solution
- Quadratic complexity



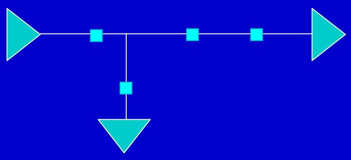
Assumptions for the Basic Van Ginneken algorithm:

- Given a routing tree
- Given a set of potential insertion points
- Single buffer size
- No sink or driver sizing
- Linear gate delay model
  - $R_d\ C_{down} + K_d$
- Elmore wire delay model
  - $R_w\ (C_w/2 + C_{down})$

Van Ginneken Extensions

- Multiple buffer types
- Inverters
- Capacitance, Slew and Noise constraints
- Wire Sizing
- Simultaneous driver sizing
- High order interconnect delay and Ceffective
- Blockage recognition

Example



- Connect the end points of the net using a steiner route

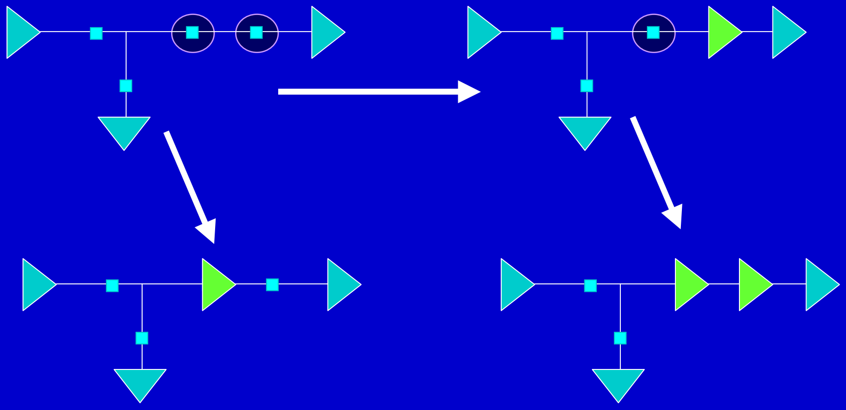
- Add Candidate Nodes

- Final buffer solution is optimal for this route, and this set of candidate nodes.

- Other routes may produce better final solutions.

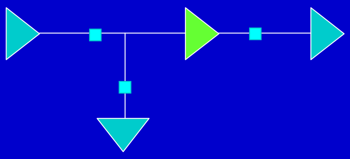
- Net routing topology is an input to Van Ginneken’s algorithm

Example



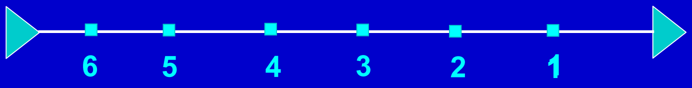
How Many Candidates?

- Number of candidates seems to double with each additional node



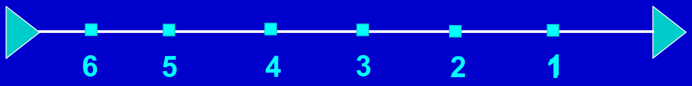
- Prune candidate with worst slack when capacitances is greater or equal
- Linear number of candidates

Pseudo Code:



```
List = NULL;
For each node (bottom up traversal of graph)
{
   augment each item in list with wire  segment up to node
   duplicate the list
   for each element of the duplicate list
            add a buffer at node
  analyze each element  in list
  analyze each element  in buffered (duplicate) list
  pick best element of buffered list and delete the rest
  new list is union of list and "best" element of buffered list
}
Pick best solution;
```

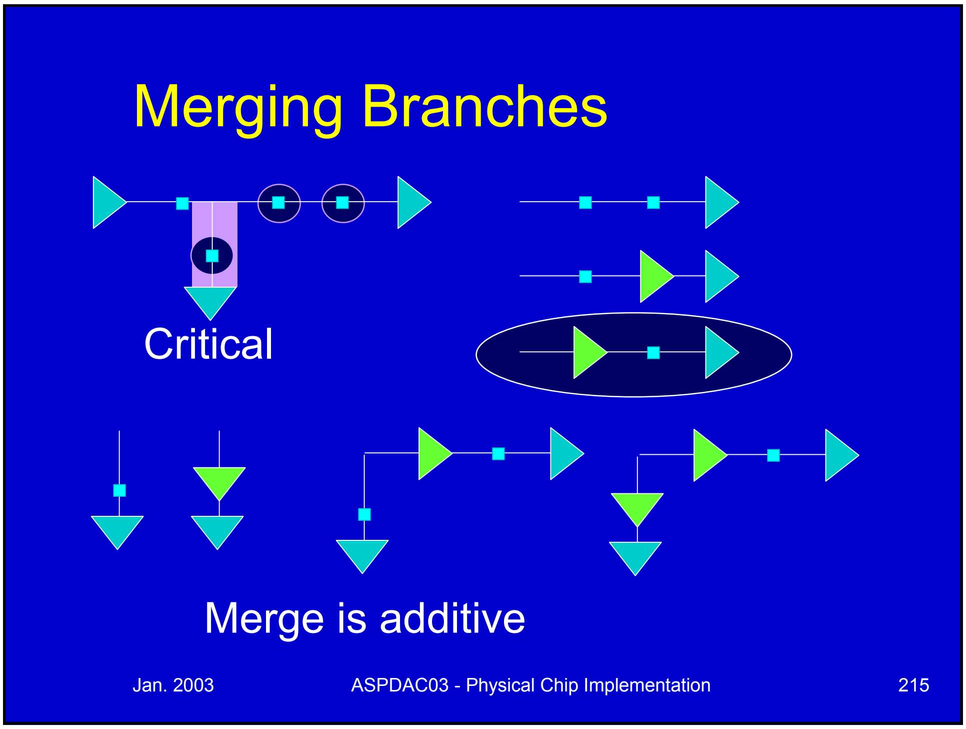
Example



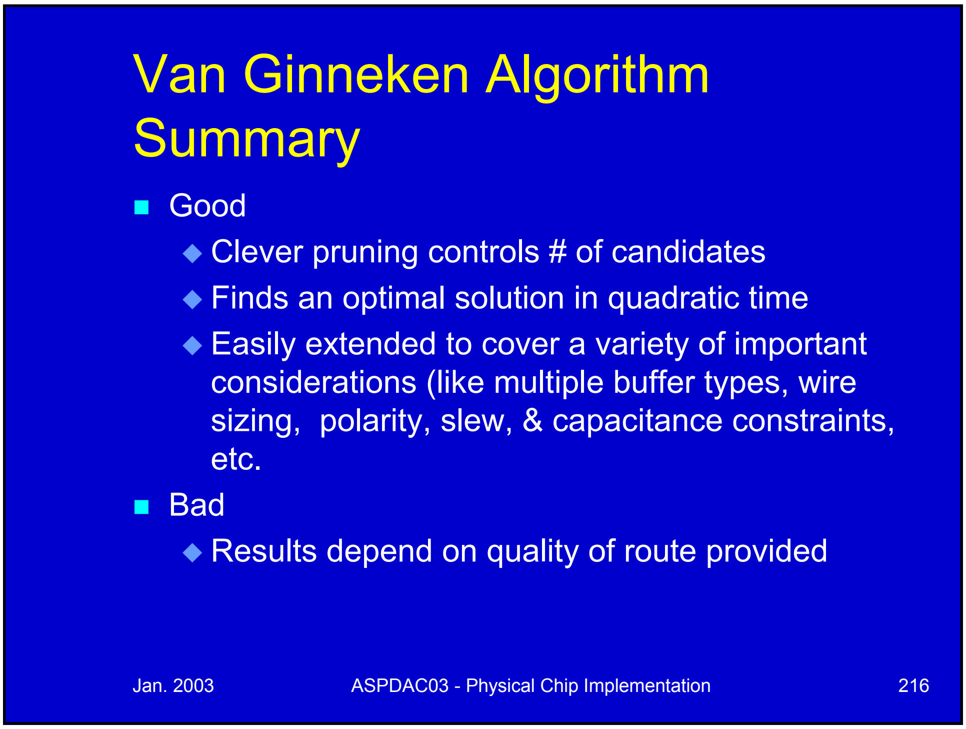
- Node 1 processing: 2 evaluations, at most 2 candidates kept
- Node 2 processing: 4 evaluations, at most 3 candidates kept
- Node 3 processing: 6 evaluations, at most 4 candidates kept
- Node 4 processing: 8 evaluations, at most 5 candidates kept
- Node 5 processing: 10 evaluations, at most 6 candidates kept
- Node 6 processing: 12 evaluations, at most 7 candidates kept
  - Now pick the best one: Optimal solution

$$Num\_evaluations = 2\sum_{i=1}^{N} i = (N)(N+1)$$

$$Num\_candidates <= N+1$$

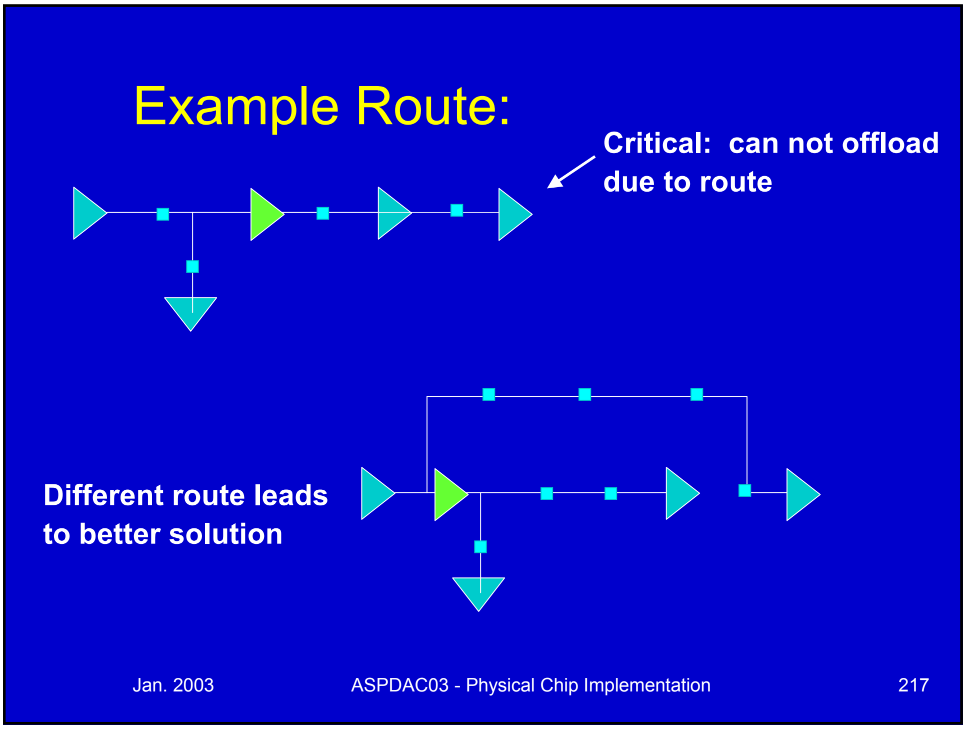


Merging Branches



Van Ginneken Algorithm Summary

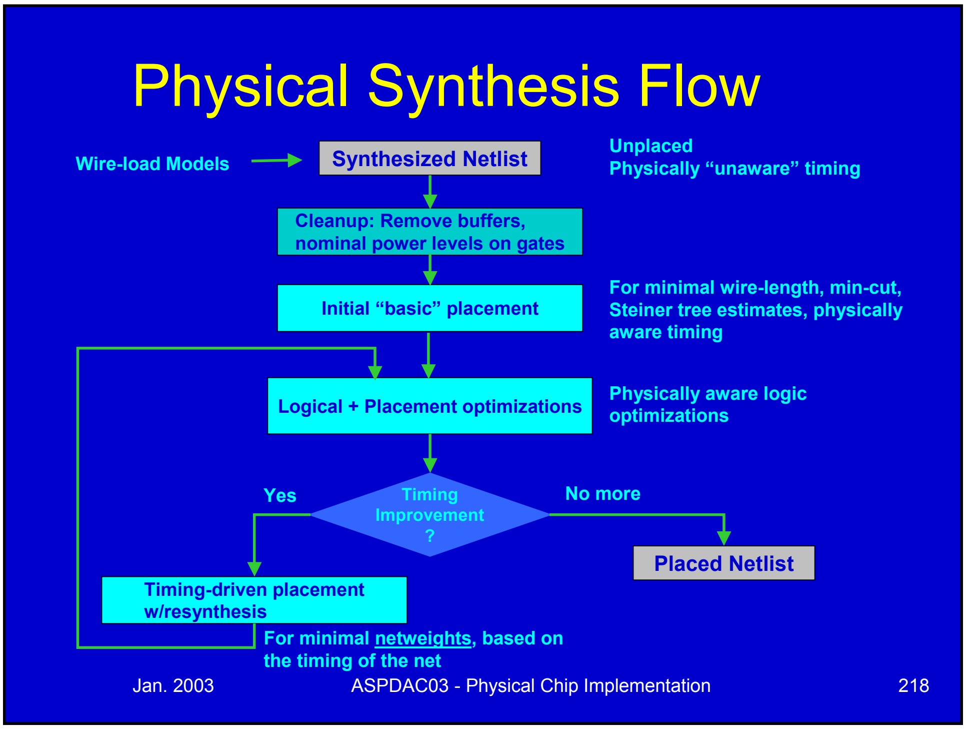
- Good
  - Clever pruning controls # of candidates
  - Finds an optimal solution in quadratic time
  - Easily extended to cover a variety of important considerations (like multiple buffer types, wire sizing, polarity, slew, & capacitance constraints, etc.
- Bad
  - Results depend on quality of route provided



Example Route:

Critical: can not offload due to route

Different route leads to better solution



Physical Synthesis Flow

Wire-load Models

Synthesized Netlist

Unplaced
Physically “unaware” timing

Cleanup: Remove buffers, nominal power levels on gates

Initial “basic” placement

For minimal wire-length, min-cut, Steiner tree estimates, physically aware timing

Logical + Placement optimizations

Physically aware logic optimizations

Timing Improvement ?

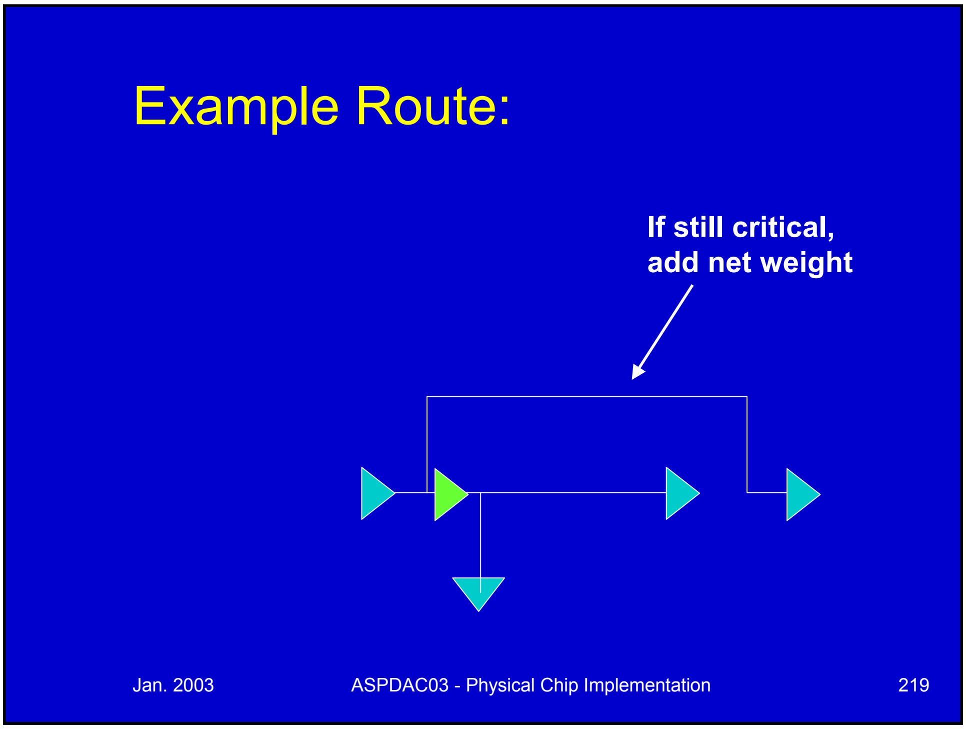
Yes

No more

Placed Netlist

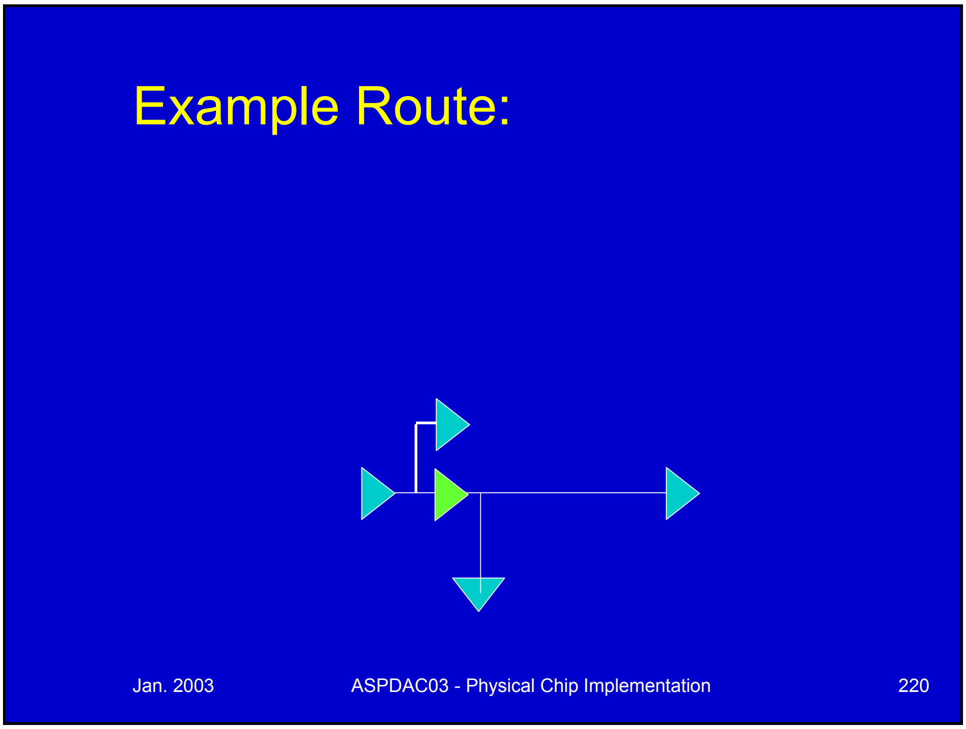
Timing-driven placement w/resynthesis

For minimal netweights, based on the timing of the net

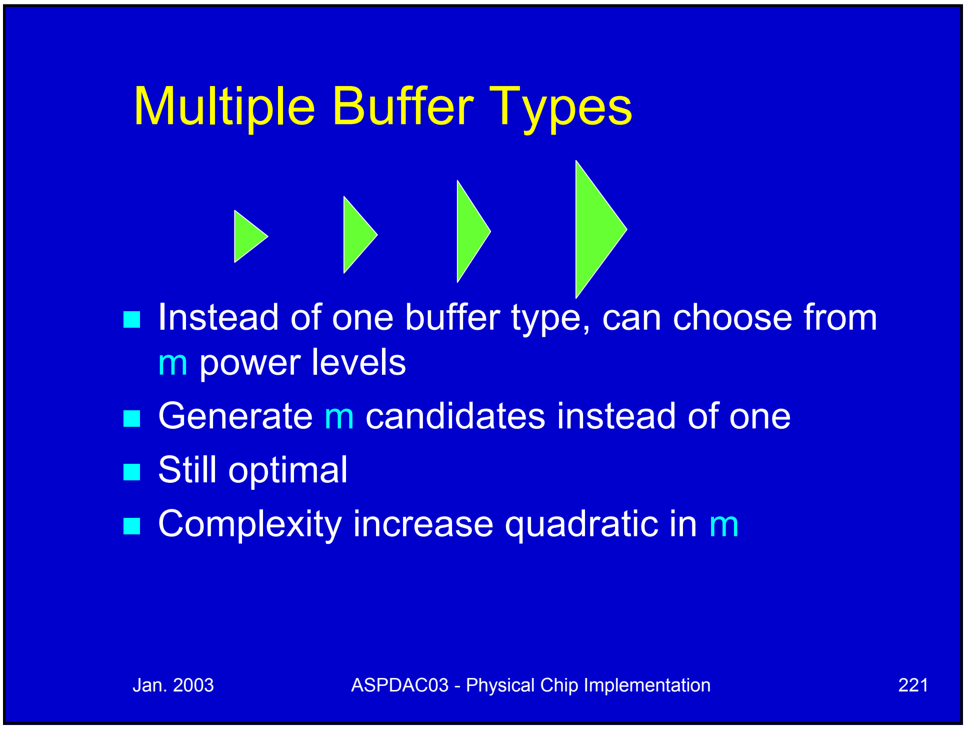


Example Route:

If still critical,
add net weight

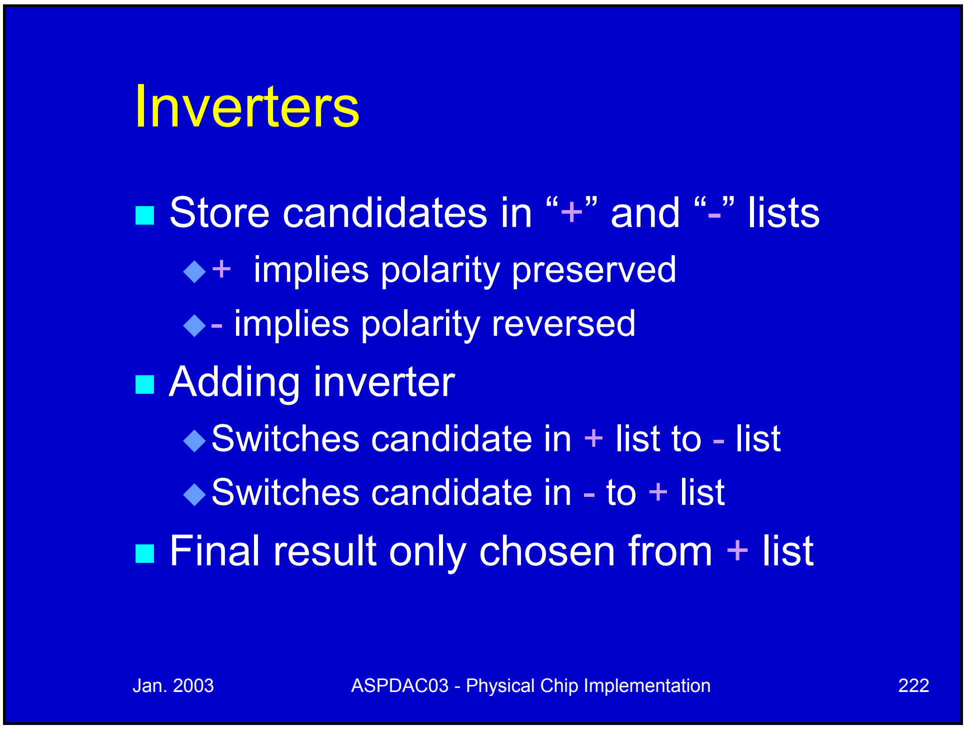


Example Route:



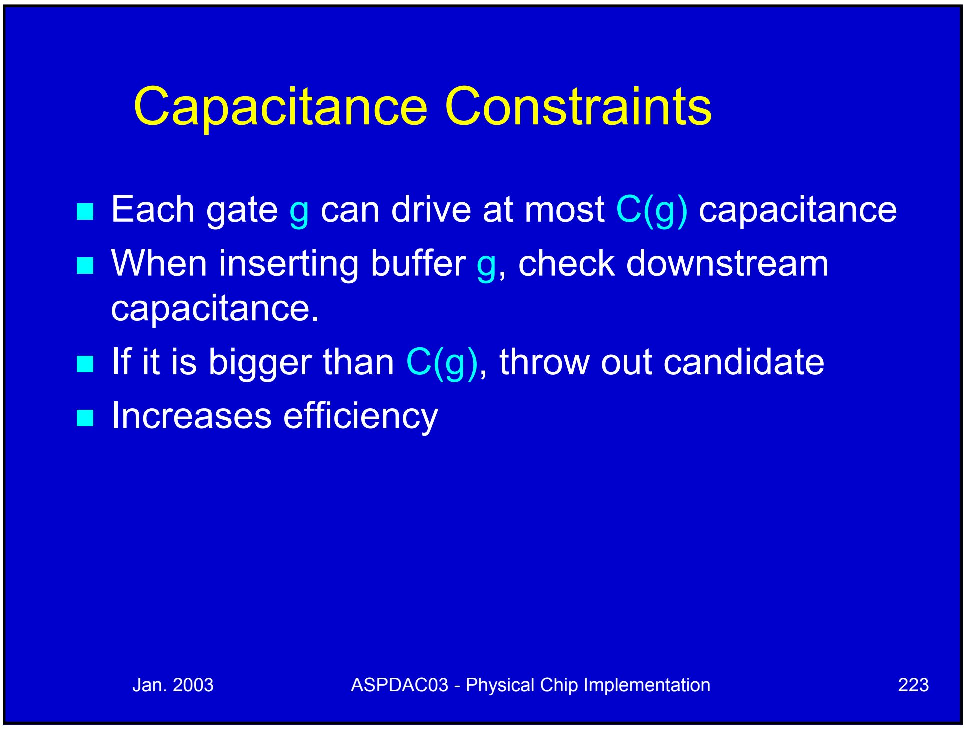
Multiple Buffer Types

- Instead of one buffer type, can choose from m power levels
- Generate m candidates instead of one
- Still optimal
- Complexity increase quadratic in m



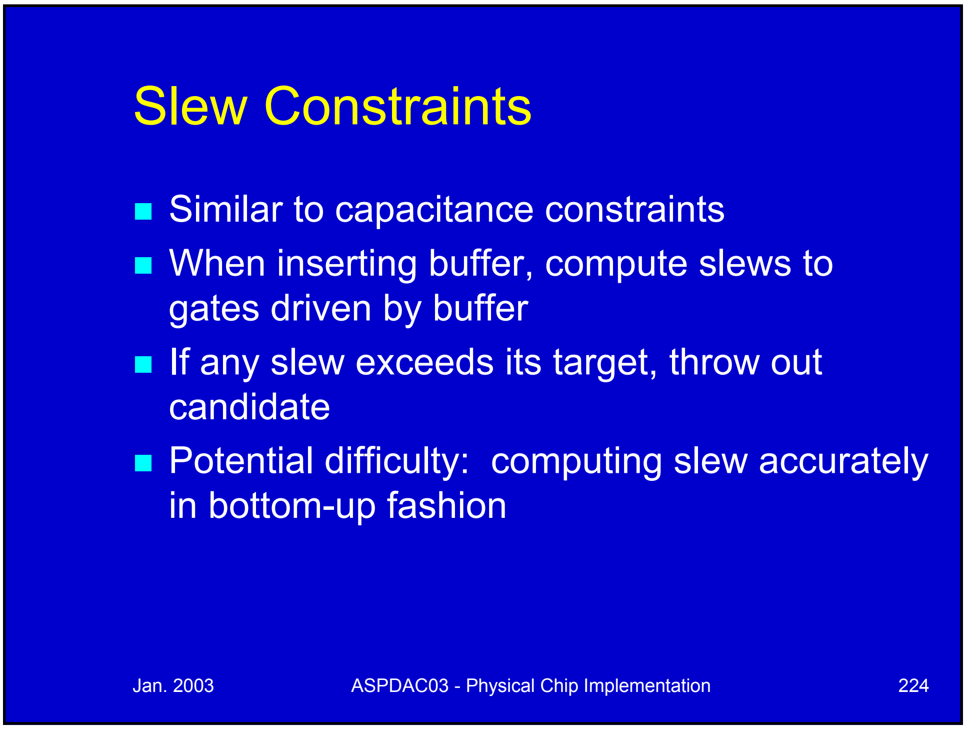
Inverters

- Store candidates in “+” and “-” lists
  - + implies polarity preserved
  - - implies polarity reversed
- Adding inverter
  - Switches candidate in + list to - list
  - Switches candidate in - to + list
- Final result only chosen from + list



Capacitance Constraints

- Each gate g can drive at most C(g) capacitance
- When inserting buffer g, check downstream capacitance.
- If it is bigger than C(g), throw out candidate
- Increases efficiency



Slew Constraints

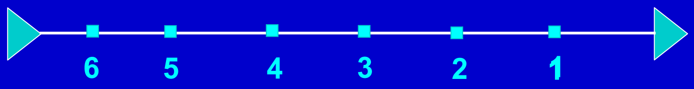
- Similar to capacitance constraints
- When inserting buffer, compute slews to gates driven by buffer
- If any slew exceeds its target, throw out candidate
- Potential difficulty: computing slew accurately in bottom-up fashion

Noise Constraints

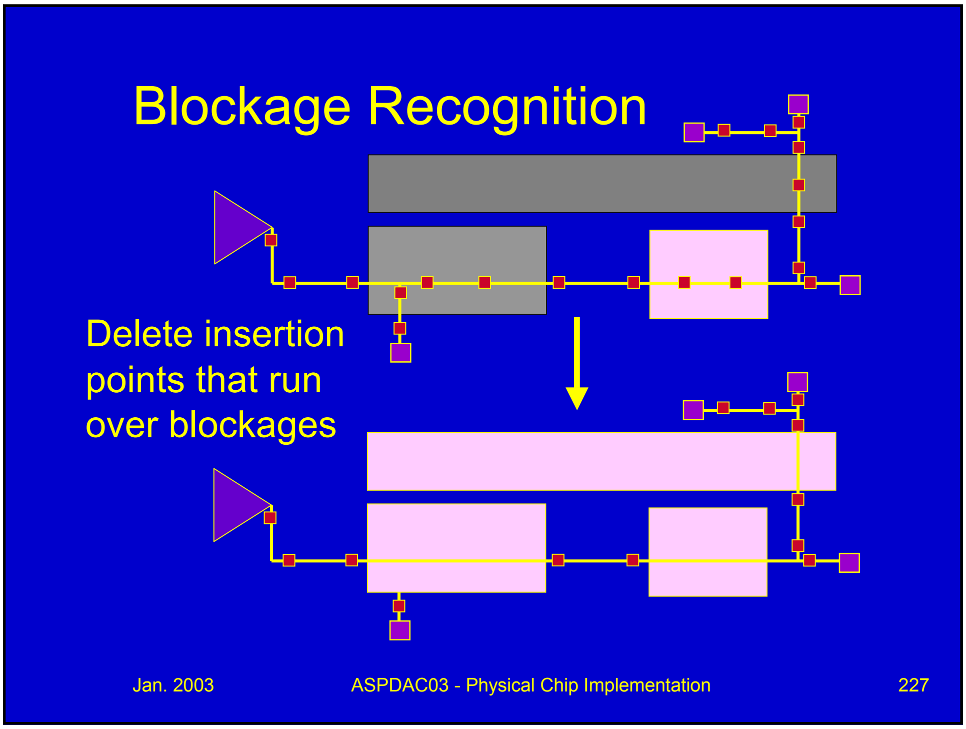
- Each gate has acceptable noise threshold
- Compute cumulative noise for each wire via Devgan noise metric
- Throw out candidates that violate noise

Can avoid noise while optimizing timing!

Wire Sizing:

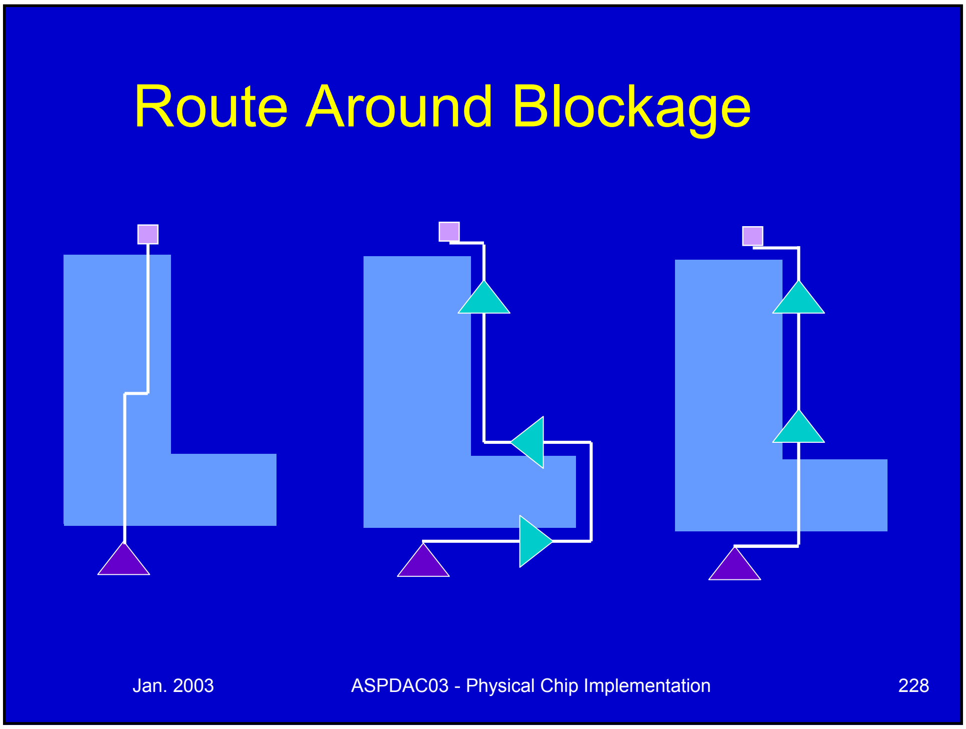


```
For each node (bottom up traversal of graph)
{
  for each Wire Size
  {
    augment each item in list with Sized wire  segment
    duplicate the list
    for each element of the duplicate list
         add a buffer at node
   analyze each element  in list
   analyze each element  in buffered (duplicate) list
   pick best element of buffered list and delete the rest
   new list is union of list and “best” element of buffered list
  }
}
Do Final pruning & Pick best solution;
```

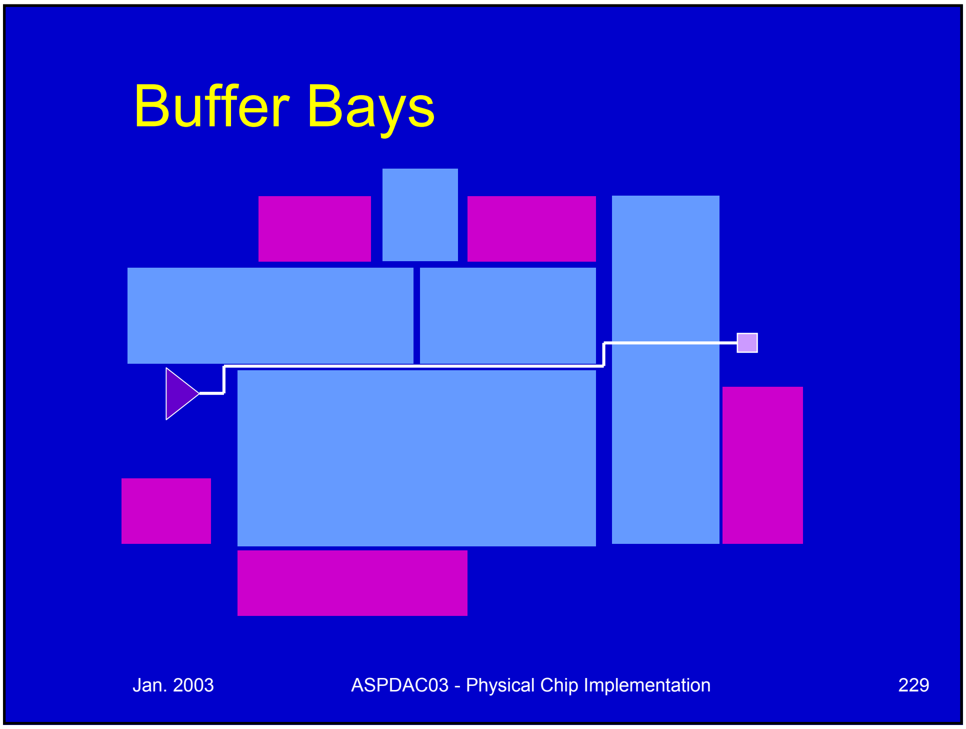


Blockage Recognition

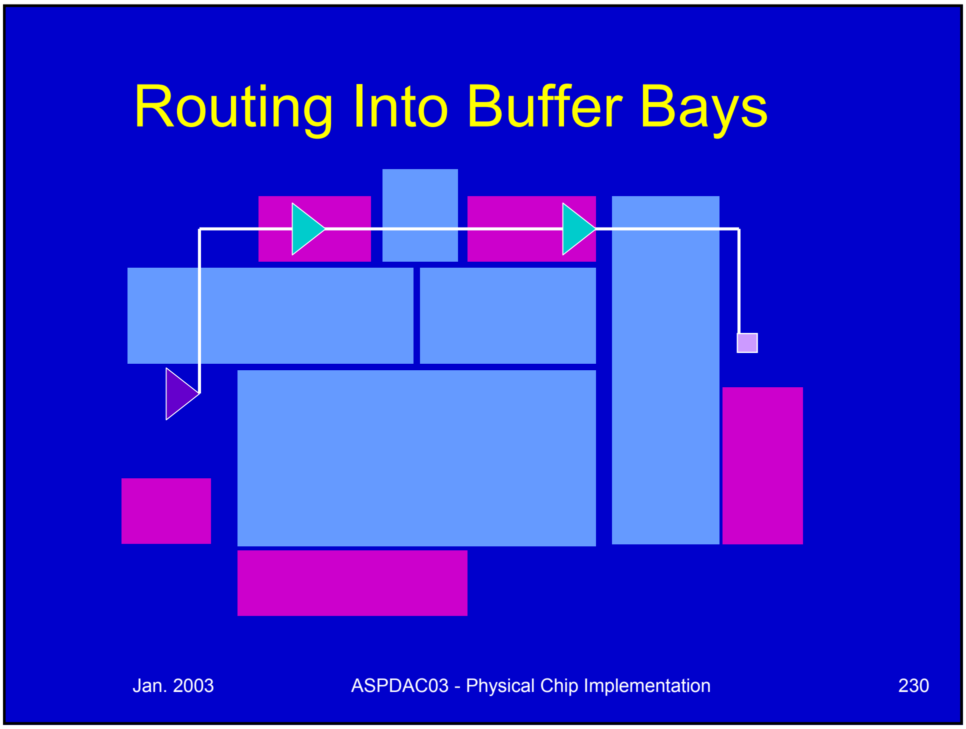
Delete insertion points that run over blockages



Route Around Blockage



Buffer Bays

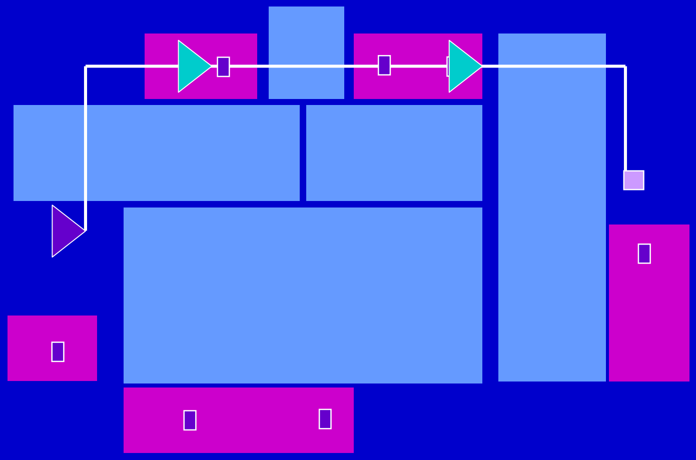


Routing Into Buffer Bays

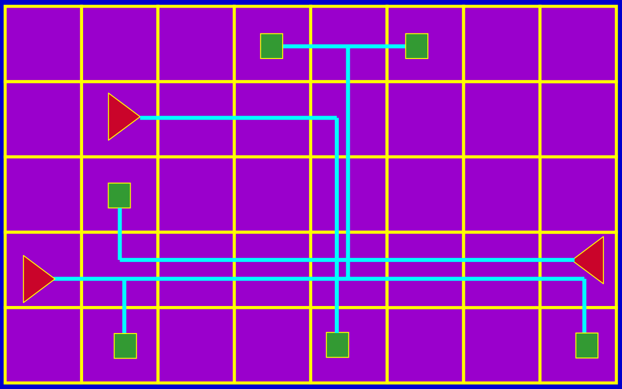
“Buffer Site”

- Similar to buffer bays, only exact buffer locations are pre-specified, not just areas
- Useful as a mechanism for IP blocks and microprocessor design
- Dummy cell that holds a buffer
- Not connected to any net
- Becomes buffer when assigned to a net
- Extra sites → decoupling caps
- Sprinkle sites throughout design
- Allocate percentage within macros

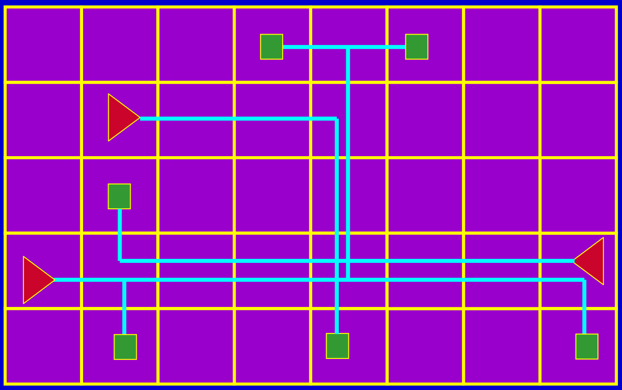
Routing Into Buffer Sites

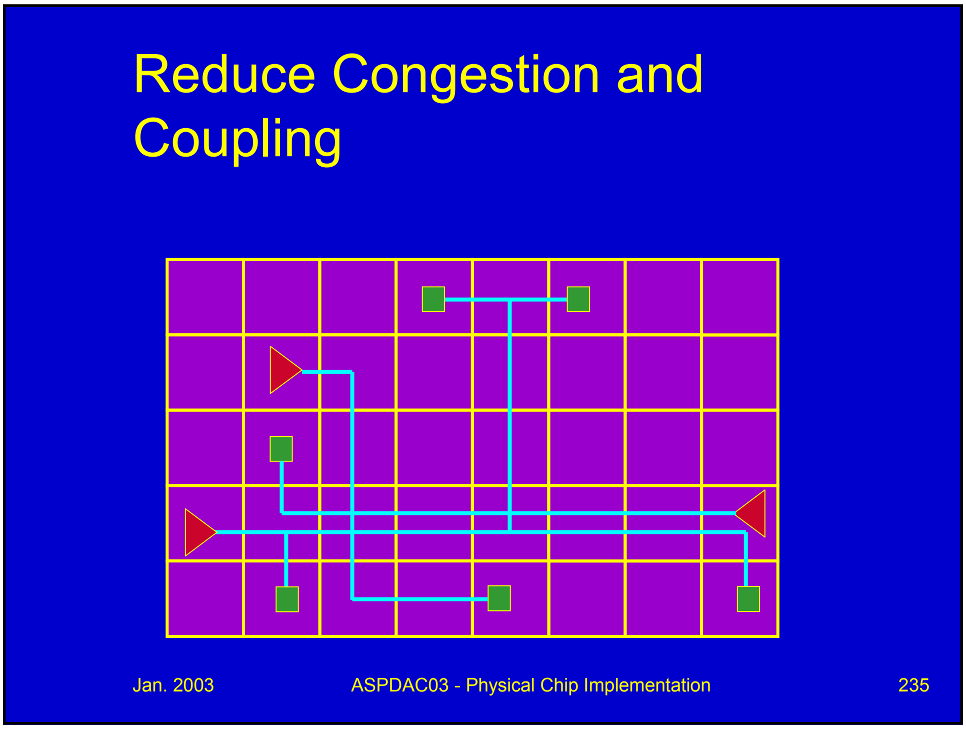


Generate Steiner Tree

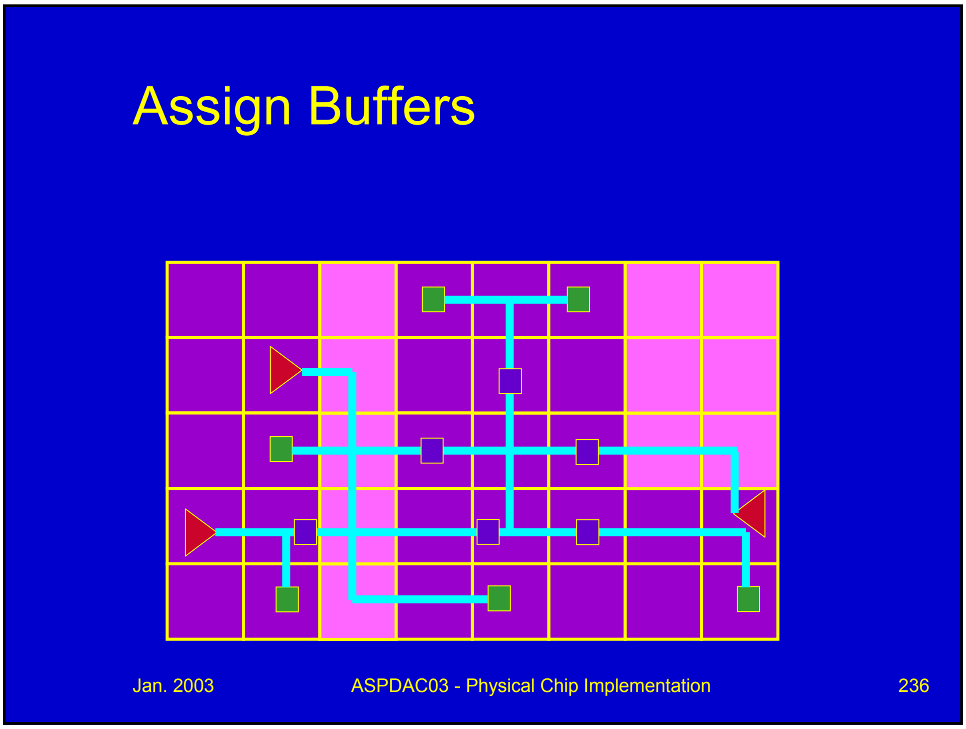


Reduce Congestion and Coupling





Reduce Congestion and Coupling



Assign Buffers

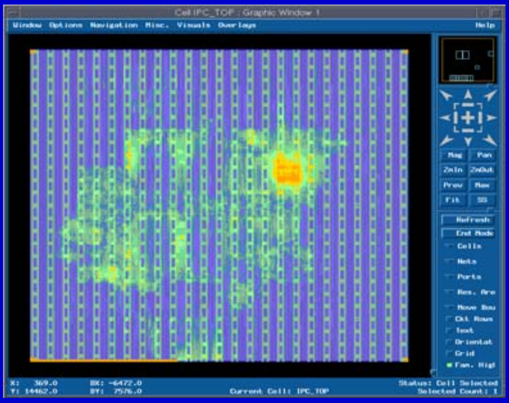
Comments about Buffering and Wire Sizing:

- Extremely critical: One of the highest leverage timing closure items
- There are extended provably correct algorithms for dealing with the problem.
- Steiner route & Blockage avoidance are mostly heuristic: Hot research area!

Section Outline

- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing Optimization

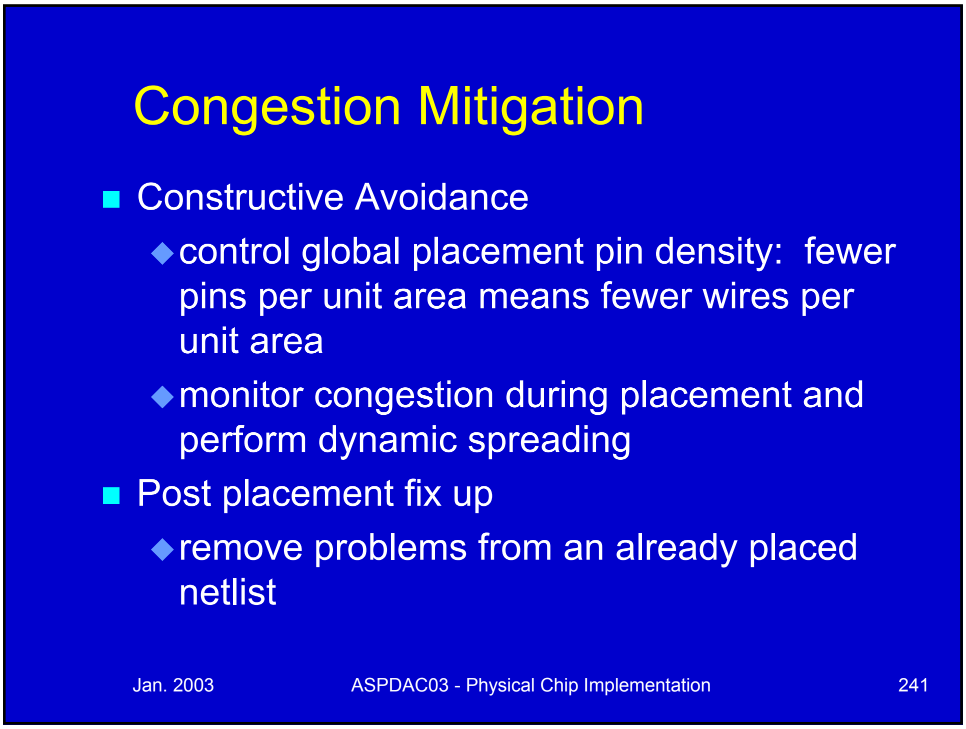
Congestion Mitigation



```
X:   369.0     DX: -6472.0                                Status: Cell Selected
Y: 14462.0     DY:  7576.0      Current Cell: IPC_TOP      Selected Count: 1
```

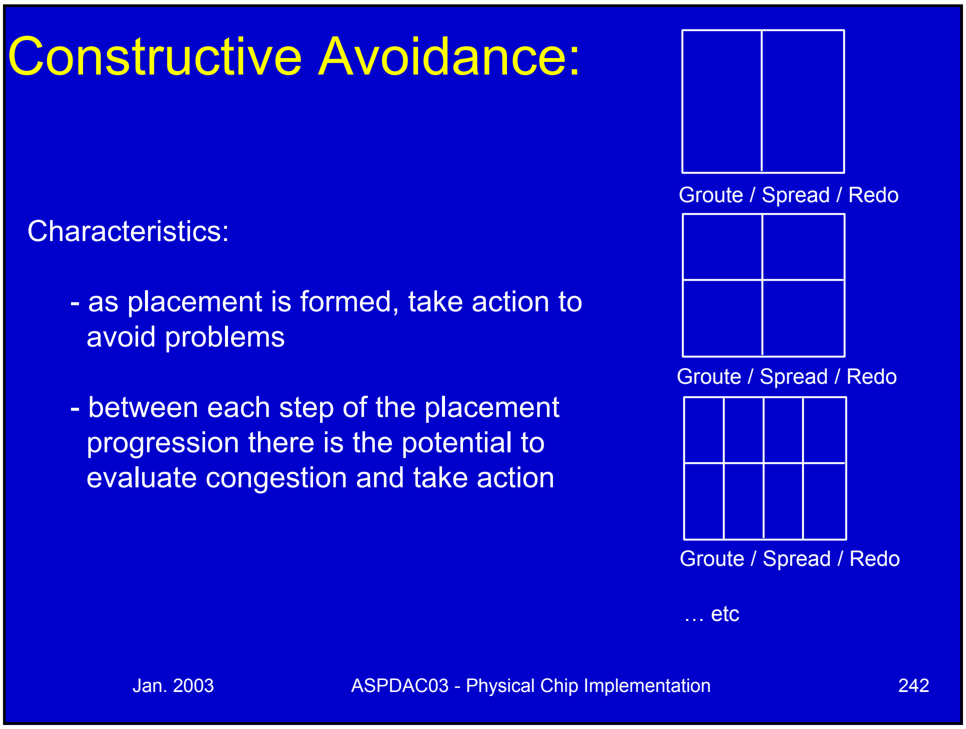
Sources of Congestion

- Placement Quality: Do we have a good relative ordering of cells?
- Placement Density: Do we have appropriate cell spreading?
- Preplacement of large cells: Is there a better location for these cells?
- Floorplan quality: Is this a good floorplan / hierarchy?
- Netlist complexity: Are some logic groupings inherently difficult to route
- Library characteristics: Do some cells block too much metal internally?



Congestion Mitigation

- Constructive Avoidance
  - control global placement pin density: fewer pins per unit area means fewer wires per unit area
  - monitor congestion during placement and perform dynamic spreading
- Post placement fix up
  - remove problems from an already placed netlist



Constructive Avoidance:

Characteristics:

- as placement is formed, take action to avoid problems

- between each step of the placement progression there is the potential to evaluate congestion and take action

Constructive Avoidance Deficiencies:

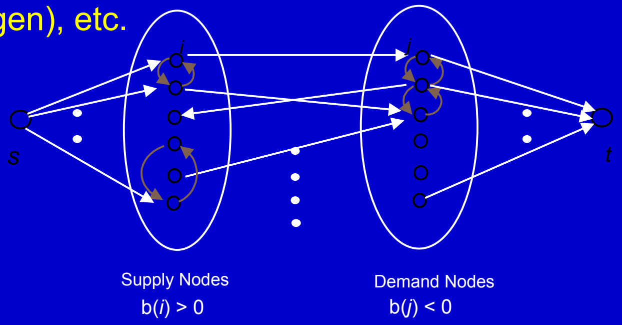
- Depends on early estimates of congestion that may not be accurate enough to avoid all problems
- Post placement actions such as clock tree insertion, repowering, buffering, etc may add congestion to the design
- Guard banding with conservative “constructive avoidance” causes lose of performance and density

Post Placement Congestion Mitigation

- Use production global router, not internal placement based global router
- Translate congestion values into density targets for placement regions
- Perform flow based circuit spreading
- Preserve relative logic ordering of cells

Network Flow Based Spreading

Min-cost max-flow formulation, similar to any “fix-up spreader”: “thermal” placement, Bonn’s top-down placer (Vygen), etc.

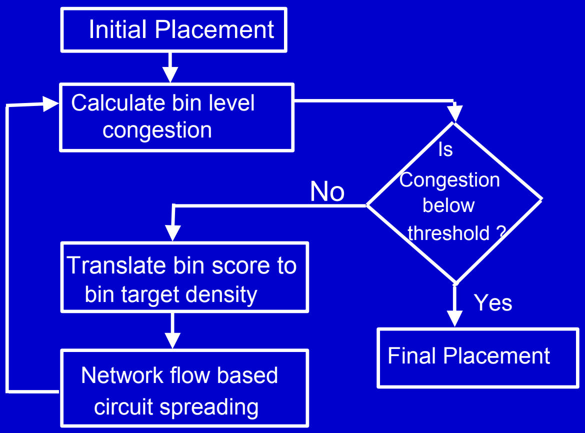


🕐 *i if* b(*i*) > *0,*
Cap(*esi*) = b(*i*)
Cost(*esi*) = 0

🕐 *i* ✏ *s* , *j* ✏ *t,*
Cap(*eij*) = *Infinity* (Large Int)
Cost(*eij*) = *K*

🕐 *j if* b(*j*) < *0,*
Cap(*ejt*) = -b(*j*)
Cost(*ejt*) = 0

Congestion Driven Circuit Spreading



Initial Placement

Calculate bin level congestion

Is Congestion below threshold ?

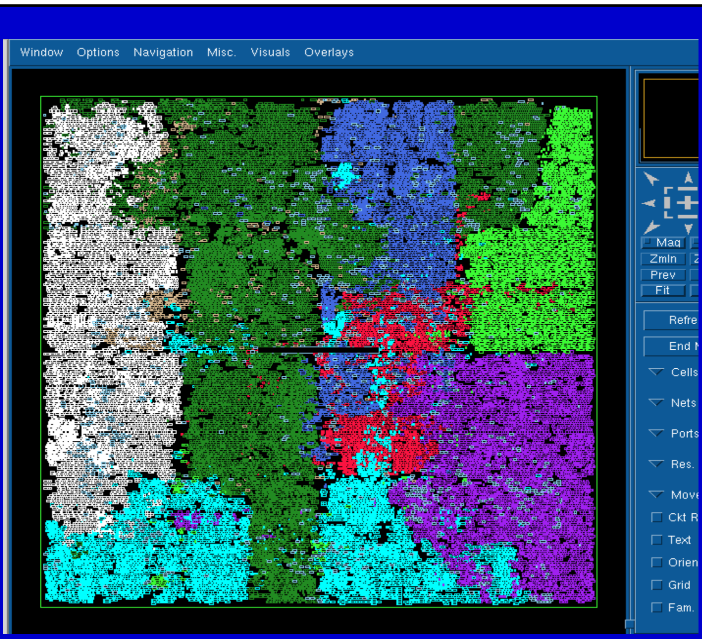
No

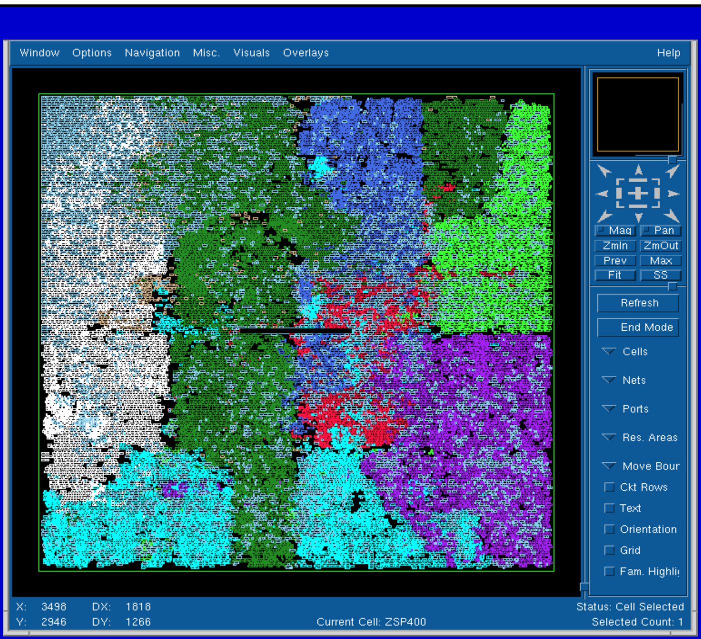
Translate bin score to bin target density

Network flow based circuit spreading

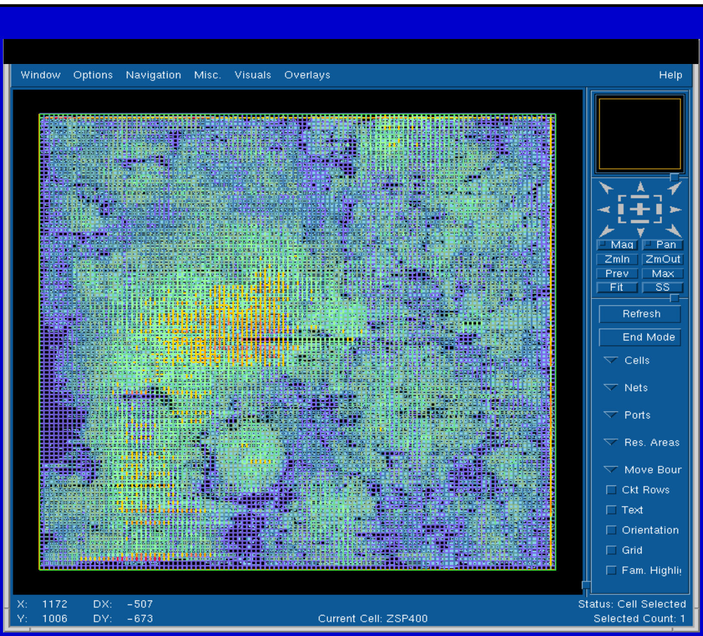
Yes

Final Placement

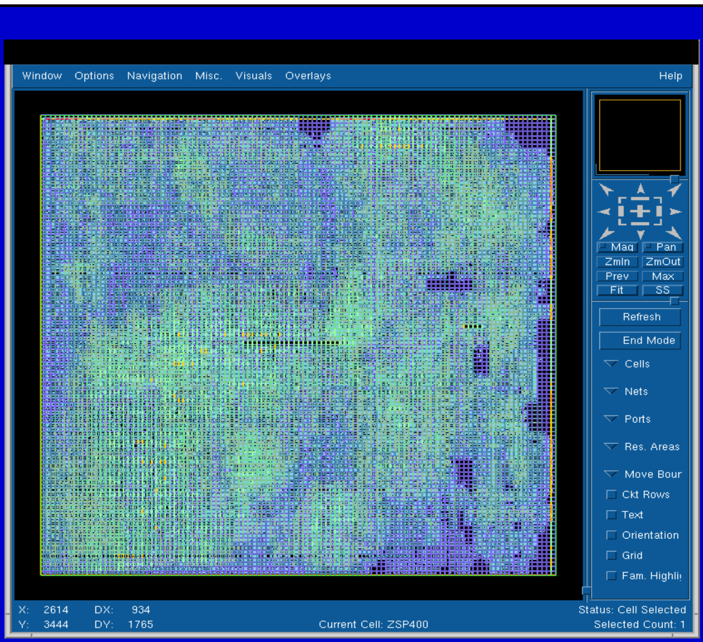




X: 3498 DX: 1818
Y: 2946 DY: 1266
Current Cell: ZSP400
Status: Cell Selected
Selected Count: 1



X: 1172 DX: -507
Y: 1006 DY: -673
Current Cell: ZSP400
Status: Cell Selected
Selected Count: 1

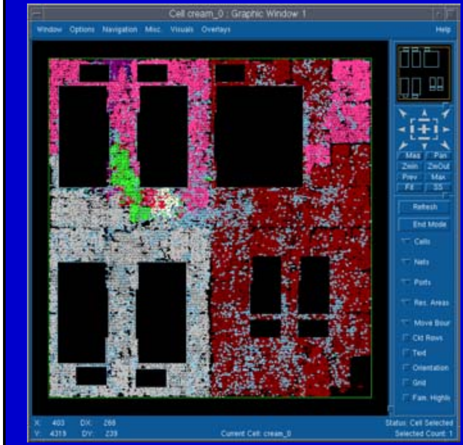


X: 2614 DX: 934
Y: 3444 DY: 1765
Current Cell: ZSP400
Status: Cell Selected
Selected Count: 1

We’ve Talked About

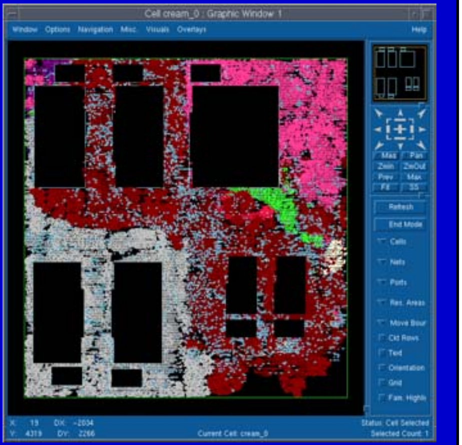
- Placement algorithms
- Placement / Synthesis interaction
- Placement aware synthesis techniques
- The Constant Delay paradigm
- Physical Buffer insertion / Wire sizing
- Congestion Mitigation

Let’s Look at some Examples:


Cell cream_0 : Graphic Window 1
Window Options Navigation Misc. Visuals Overlays Help
Mag Pan ZmIn ZmOut Prev Max Fit SS
Refresh
End Mode
Cells
Nets
Ports
Res. Areas
Move Bour
Ckt Rows
Text
Orientation
Grid
Fam. Highli

X: 403 DX: 266
Y: 4319 DY: 239
Current Cell: cream_0
Status: Cell Selected
Selected Count: 1

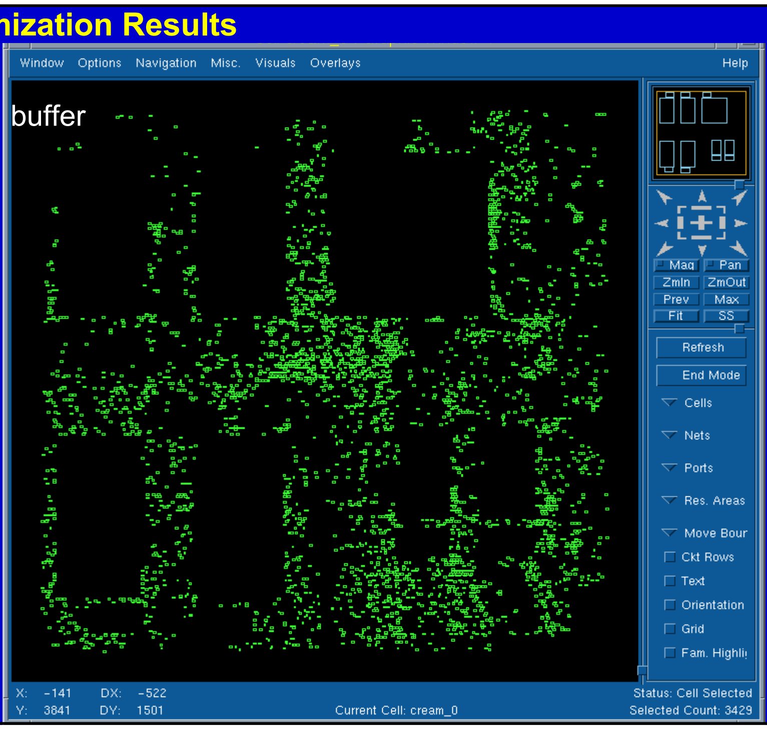
Pure MLP


Cell cream_0 : Graphic Window 1
Window Options Navigation Misc. Visuals Overlays Help
Mag Pan ZmIn ZmOut Prev Max Fit SS
Refresh
End Mode
Cells
Nets
Ports
Res. Areas
Move Bour
Ckt Rows
Text
Orientation
Grid
Fam. Highli

X: 19 DX: -2004
Y: 4319 DY: 2266
Current Cell: cream_0
Status: Cell Selected
Selected Count: 1

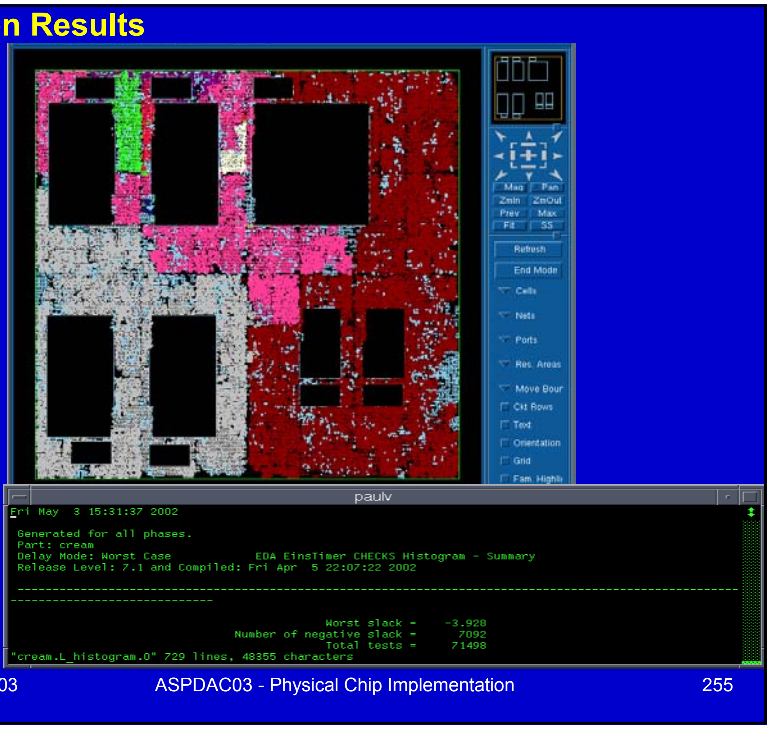
Quadratic

Optimization Results



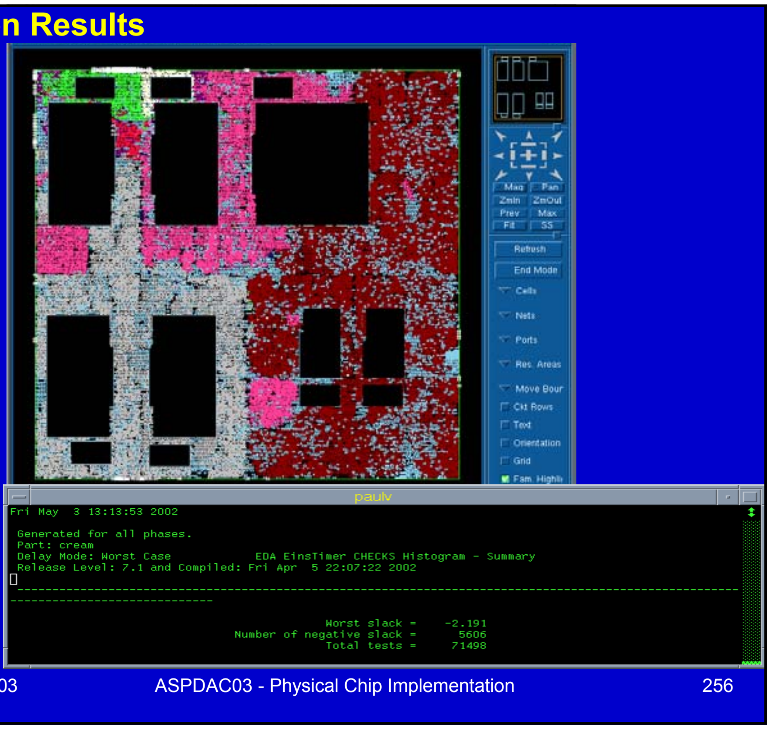
X: -141 DX: -522
Y: 3841 DY: 1501
Current Cell: cream_0
Status: Cell Selected
Selected Count: 3429

Optimization Results



```
Fri May  3 15:31:37 2002

 Generated for all phases.
 Part: cream
 Delay Mode: Worst Case             EDA EinsTimer CHECKS Histogram - Summary
 Release Level: 7.1 and Compiled: Fri Apr  5 22:07:22 2002

 -------------------------------------------------------------------------------------------------------
----------------------------

                                       Worst slack =    -3.928
                          Number of negative slack =      7092
                                       Total tests =     71498
"cream.L_histogram.0" 729 lines, 48355 characters
```

Optimization Results

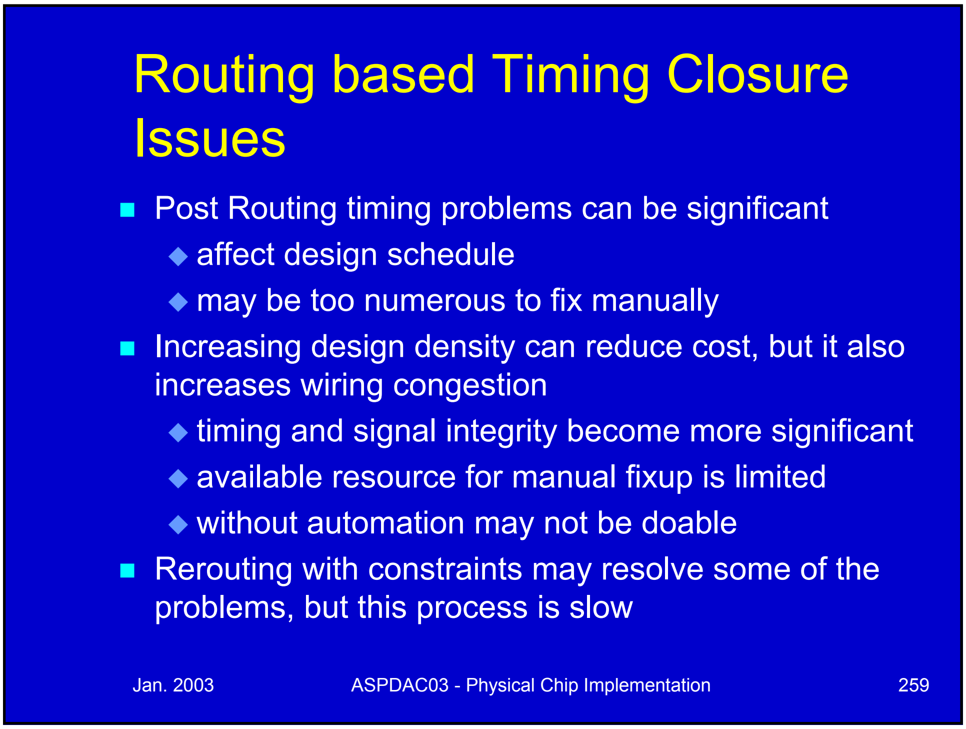


```
Fri May  3 13:13:53 2002

 Generated for all phases.
 Part: cream
 Delay Mode: Worst Case             EDA EinsTimer CHECKS Histogram - Summary
 Release Level: 7.1 and Compiled: Fri Apr  5 22:07:22 2002

 -------------------------------------------------------------------------------------------------------
----------------------------

                                       Worst slack =    -2.191
                          Number of negative slack =      5606
                                       Total tests =     71498
```

Section outline

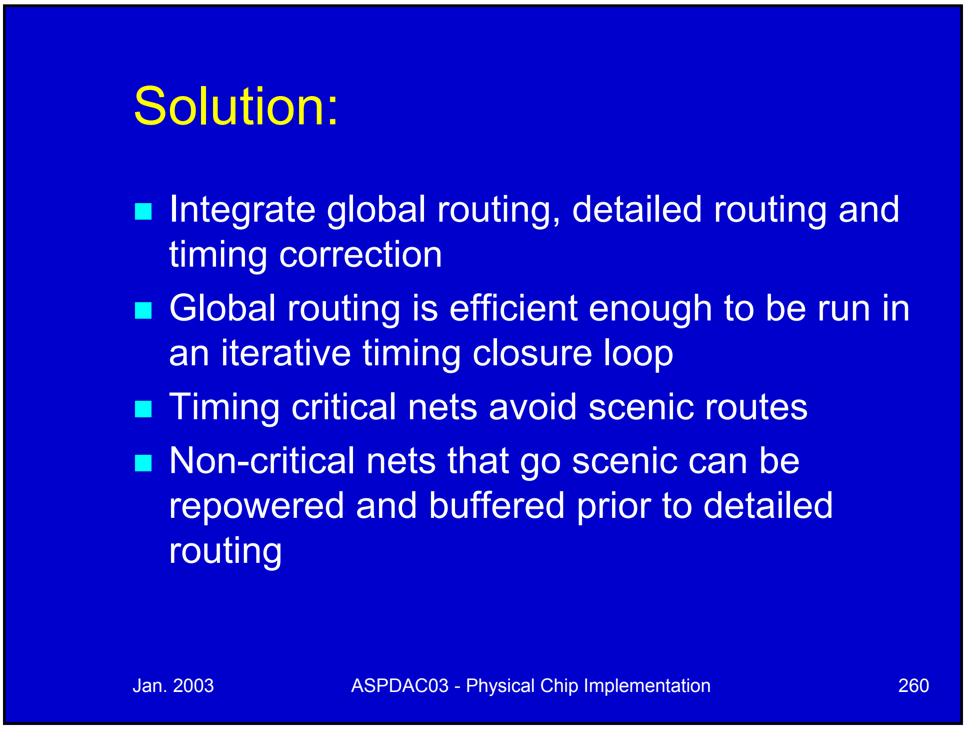
- Introduction
- Review material (timing and synthesis)
- Introduction to placement
- Placement algorithms
- Paradigms for placement-synthesis integration
- Placement aware synthesis techniques
- Congestion avoidance / mitigation techniques
- Routing optimization

Routing Based Optimization: RBO (IBM)



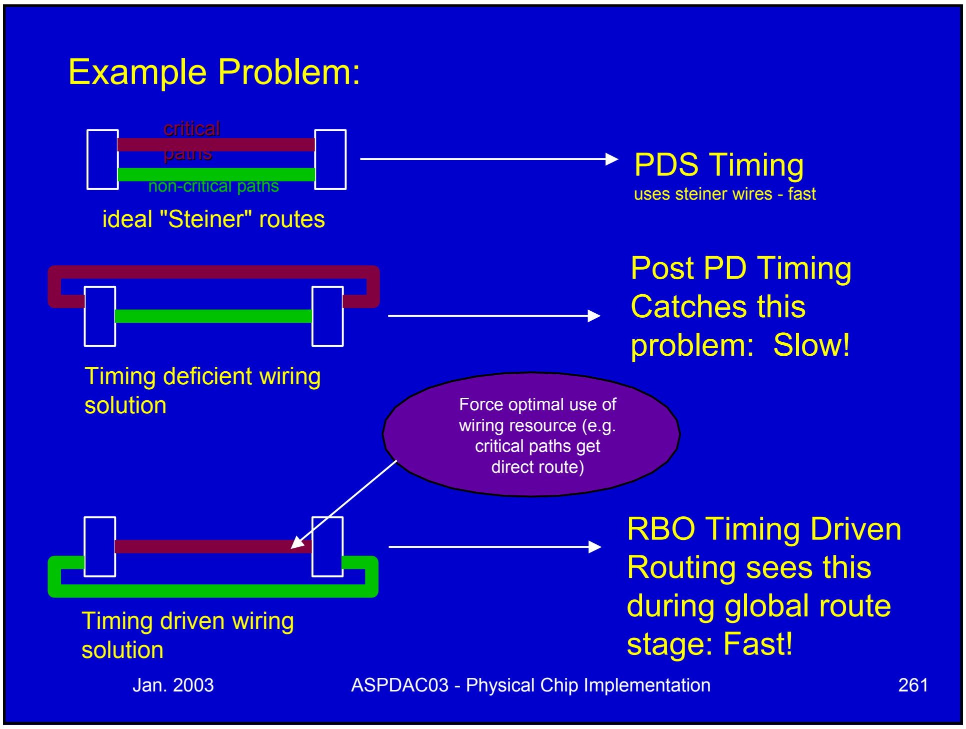
Routing based Timing Closure Issues

- Post Routing timing problems can be significant
  - affect design schedule
  - may be too numerous to fix manually
- Increasing design density can reduce cost, but it also increases wiring congestion
  - timing and signal integrity become more significant
  - available resource for manual fixup is limited
  - without automation may not be doable
- Rerouting with constraints may resolve some of the problems, but this process is slow



Solution:

- Integrate global routing, detailed routing and timing correction
- Global routing is efficient enough to be run in an iterative timing closure loop
- Timing critical nets avoid scenic routes
- Non-critical nets that go scenic can be repowered and buffered prior to detailed routing



Example Problem:

critical paths

non-critical paths

ideal "Steiner" routes

PDS Timing

uses steiner wires - fast

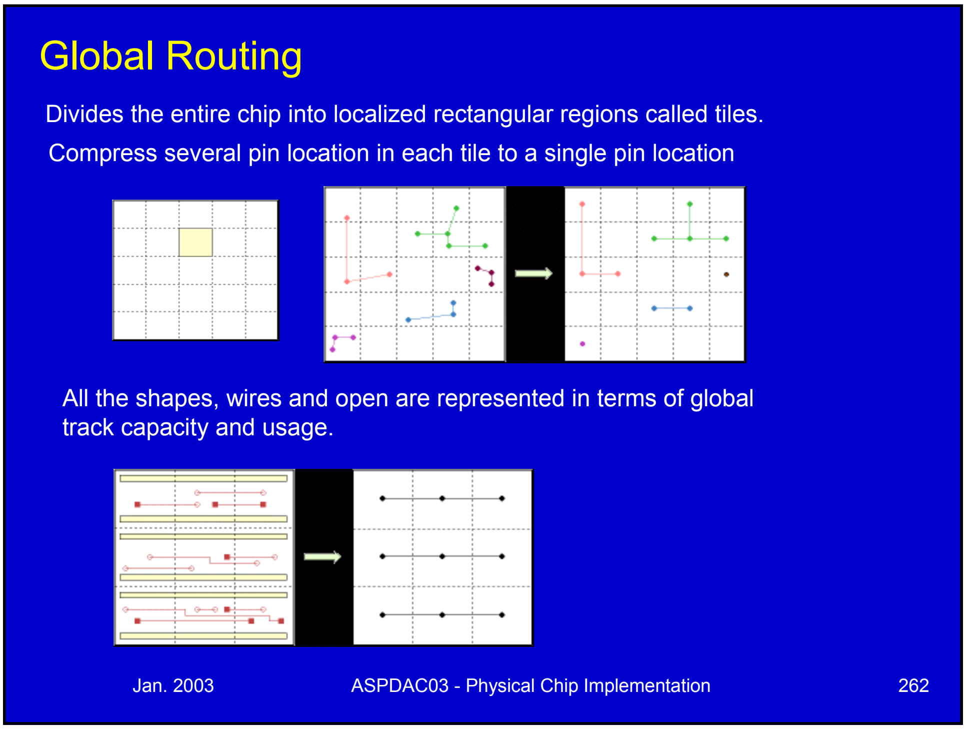
Timing deficient wiring solution

Post PD Timing Catches this problem: Slow!

Force optimal use of wiring resource (e.g. critical paths get direct route)

Timing driven wiring solution

RBO Timing Driven Routing sees this during global route stage: Fast!



Global Routing

Divides the entire chip into localized rectangular regions called tiles.

Compress several pin location in each tile to a single pin location

All the shapes, wires and open are represented in terms of global track capacity and usage.

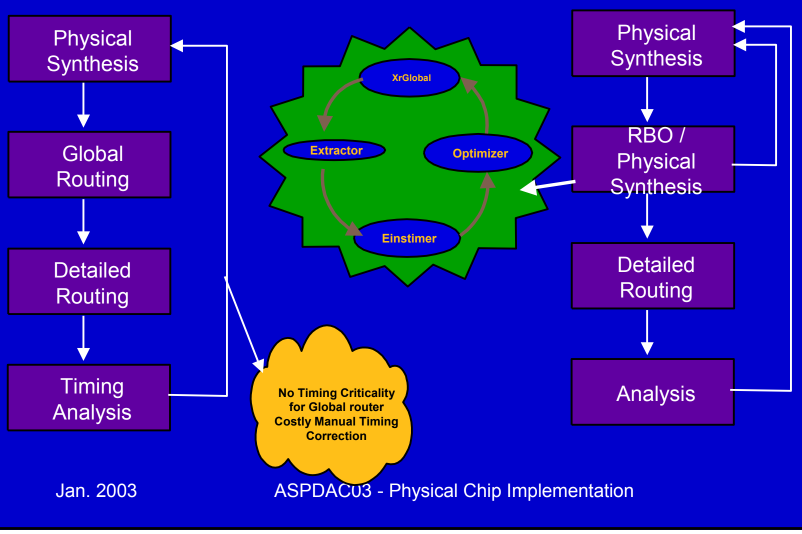
Global Routing

- Two step approach
  - Create the initial steiner routes
  - Compute the edge congestion's on the grid
  - Perform a rip-up reroute using shortest path algorithm to reduce the overall congestion of the design
- Advantages
  - Can communicate with detail router
  - Good correlation with final detail routing solution

Routing Based Optimization

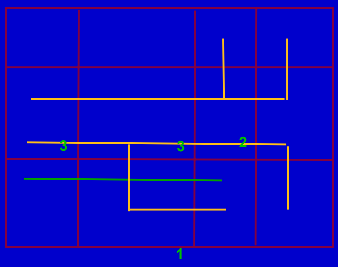
Current Methodology

RBO Methodology



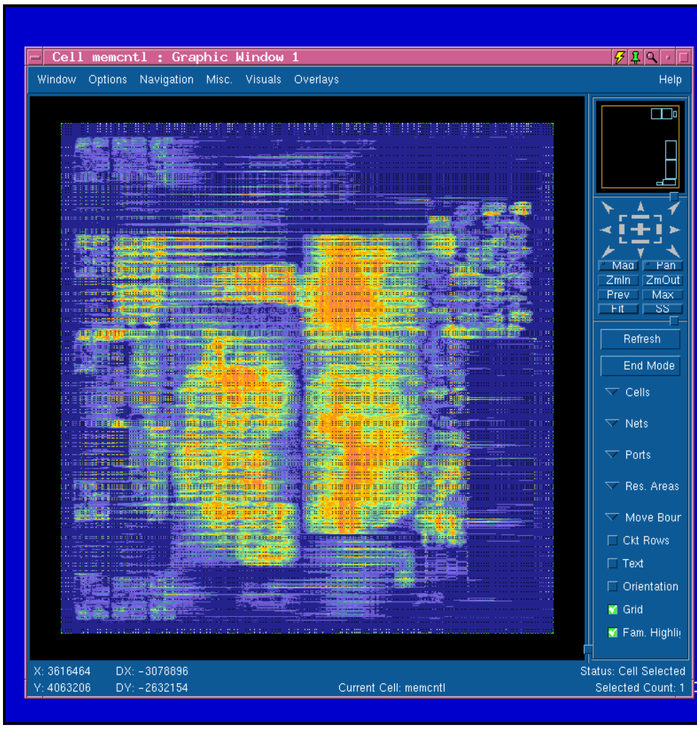
RBO Extraction Process

- Very fast
  - Excellent correlation with final 3D extraction
- Uses global routes for extraction
- Neighbor information probabilistically determined based on the global routing congestion information
- Based on extraction tables



Capacity of All Edges = 5

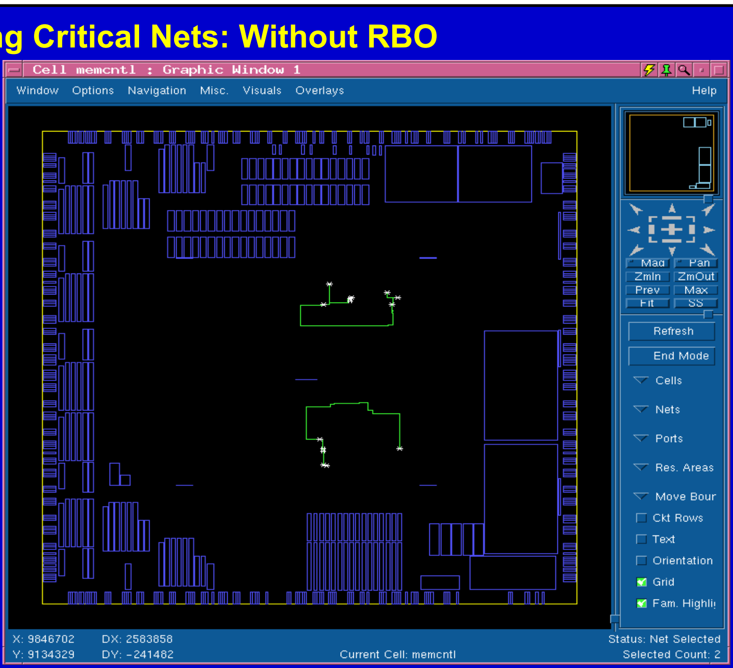
Probability of having a neighbor = (#OccupiedTracks)/(#Capacity) = 2/4 = 0.5



X: 3616464 DX: -3078896
Y: 4063206 DY: -2632154
Current Cell: memcntl
Status: Cell Selected
Selected Count: 1

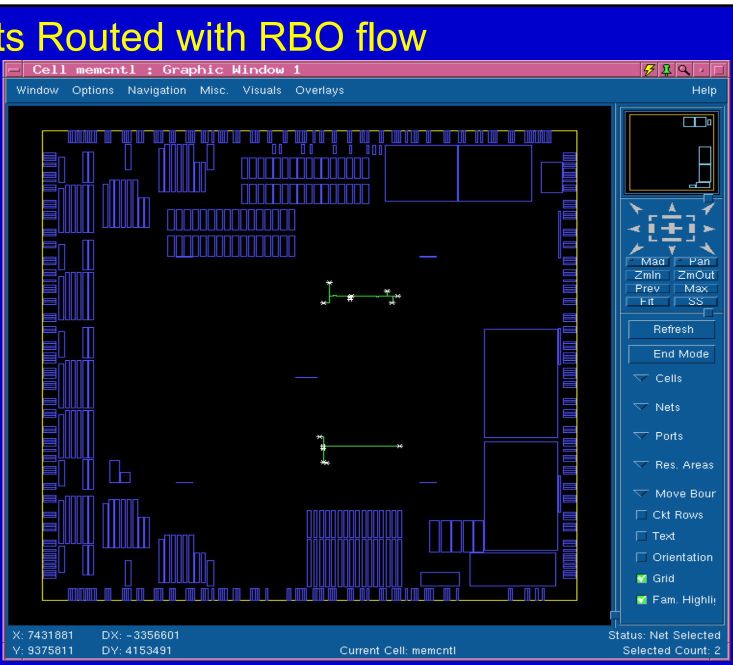
Design : Example 1
Nets : ~1.6M
Size : 23193 x 23193
Congestion : Attached is a display of Global congestion

Timing Critical Nets: Without RBO



X: 9846702 DX: 2583858 Status: Net Selected
Y: 9134329 DY: -241482 Current Cell: memcntl Selected Count: 2

Nets Routed with RBO flow

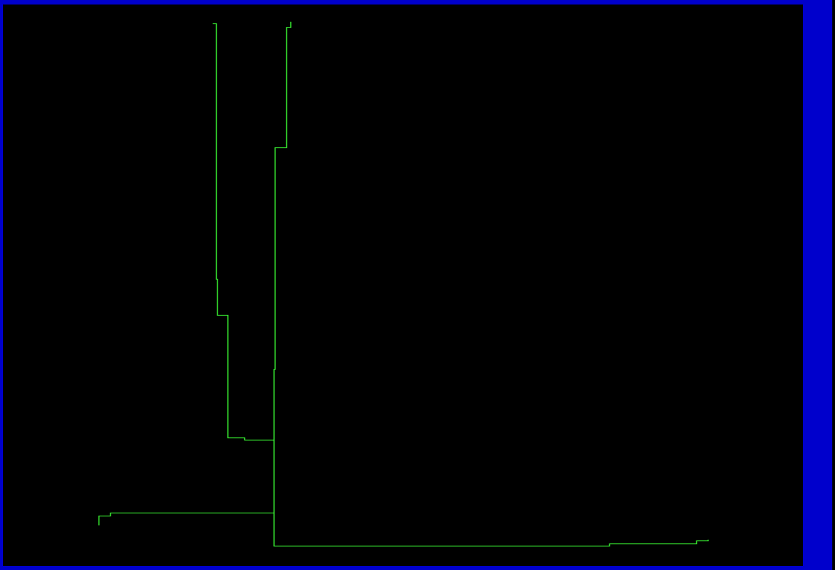


X: 7431881 DX: -3356601 Status: Net Selected
Y: 9375811 DY: 4153491 Current Cell: memcntl Selected Count: 2

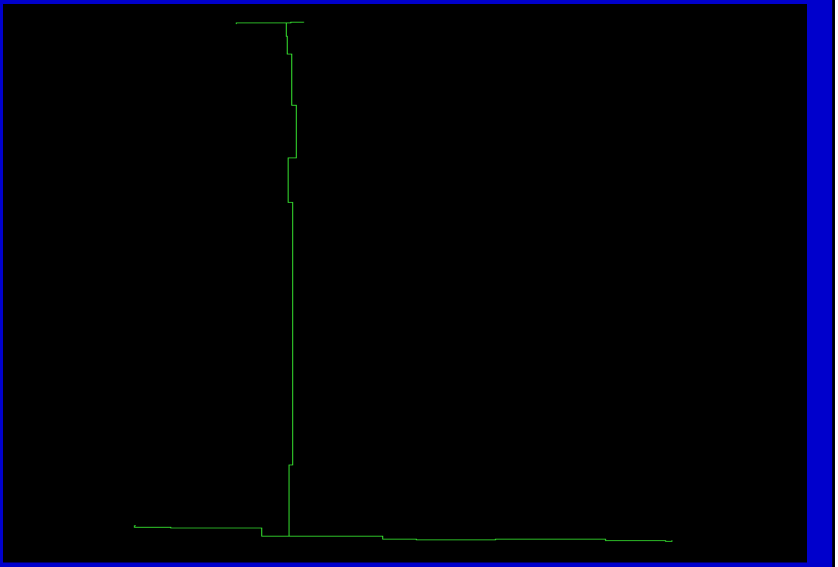
RBO Results - Example 1

	Worst Slack	#Slack Violations	#Cap Violations	#Slew Violations	#Opens	#Loops
Steiner Estimates	-0.47	17	1	18		
XrLocal without RBO	-1.57	4687	14	128	50	87020
RBO Timing Closure (Global Routes)	-0.48	209				
Detailed routing with RBO	-0.43	14	18	1	54	

Example 2: - Critical Net Routed Without RBO

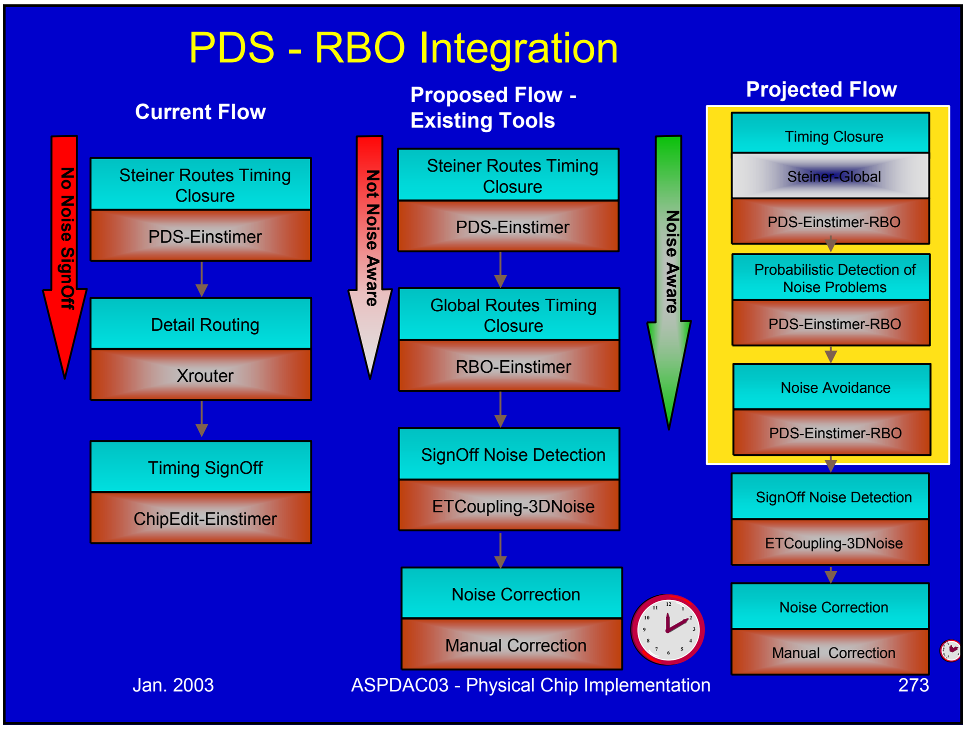


Example 2: Critical Net Routed With RBO



Example 2 Results Summary

	Worst Slack	#Slack Violations	#Cap Violations	#Slew Violations	#Opens	#Loops
Final routing without RBO	-0.54	1224	33	270	0	1152
Using RBO	-0.29	909	32	274	0	1070



PDS - RBO Integration

Current Flow

No Noise SignOff

Steiner Routes Timing Closure
PDS-Einstimer

Detail Routing
Xrouter

Timing SignOff
ChipEdit-Einstimer

Proposed Flow - Existing Tools

Not Noise Aware

Steiner Routes Timing Closure
PDS-Einstimer

Global Routes Timing Closure
RBO-Einstimer

SignOff Noise Detection
ETCoupling-3DNoise

Noise Correction
Manual Correction

Projected Flow

Noise Aware

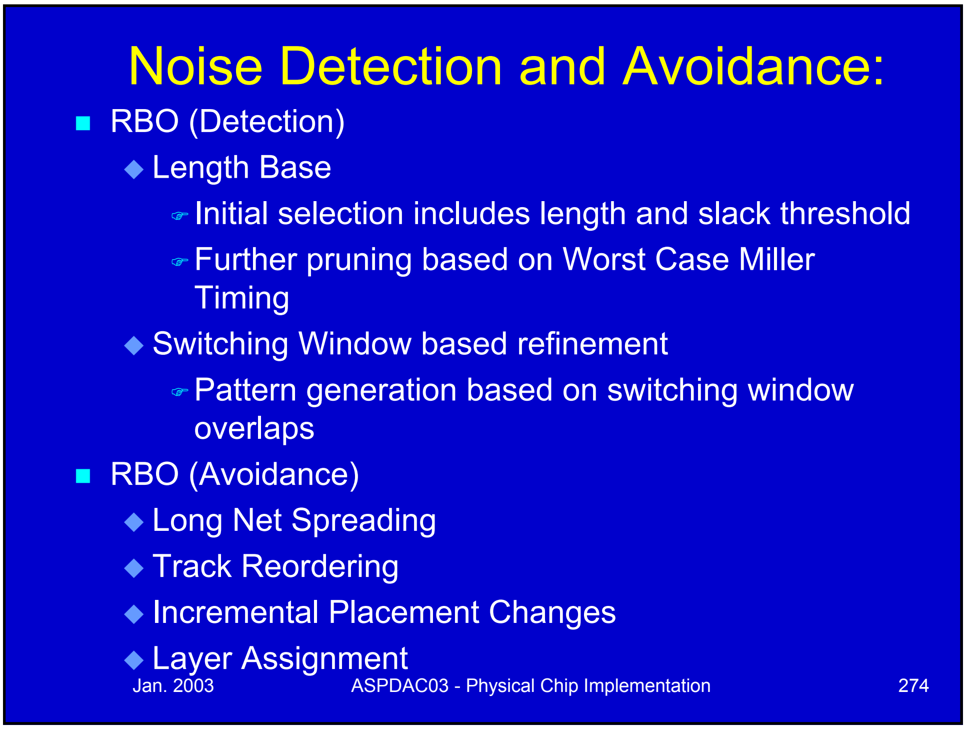
Timing Closure
Steiner-Global
PDS-Einstimer-RBO

Probabilistic Detection of Noise Problems
PDS-Einstimer-RBO

Noise Avoidance
PDS-Einstimer-RBO

SignOff Noise Detection
ETCoupling-3DNoise

Noise Correction
Manual Correction



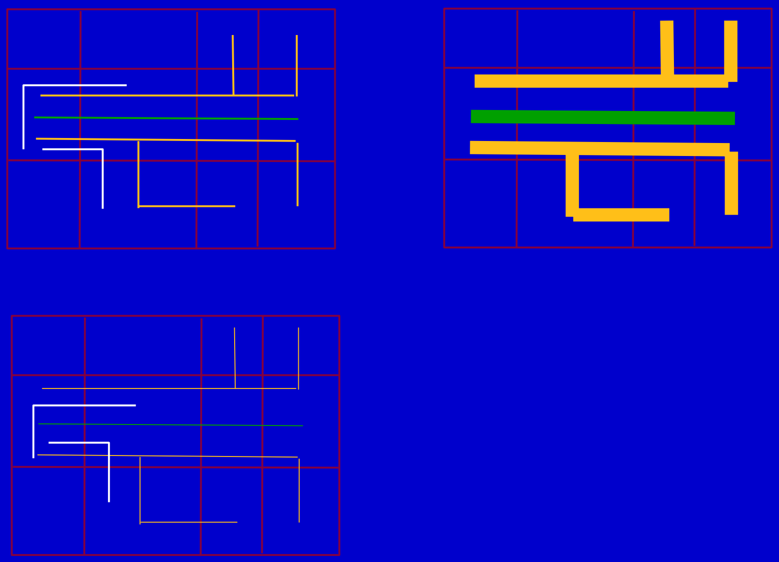
Noise Detection and Avoidance:

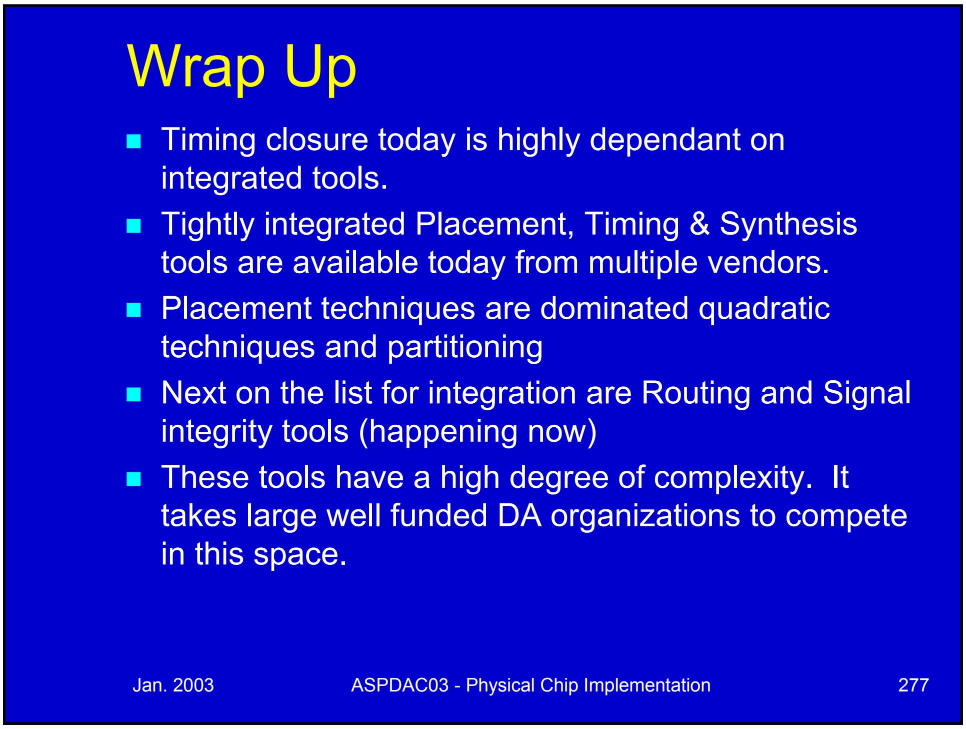
- RBO (Detection)
  - Length Base
    - Initial selection includes length and slack threshold
    - Further pruning based on Worst Case Miller Timing
  - Switching Window based refinement
    - Pattern generation based on switching window overlaps
- RBO (Avoidance)
  - Long Net Spreading
  - Track Reordering
  - Incremental Placement Changes
  - Layer Assignment

Noise Detection and Avoidance:

- Wire width selection
- Physical Synthesis
  - Integration
  - Fix Cap And Slew Violations with Global Routes
  - Interface to RBO
  - Noise Alleviation Resizing
  - Noise Aware Buffering

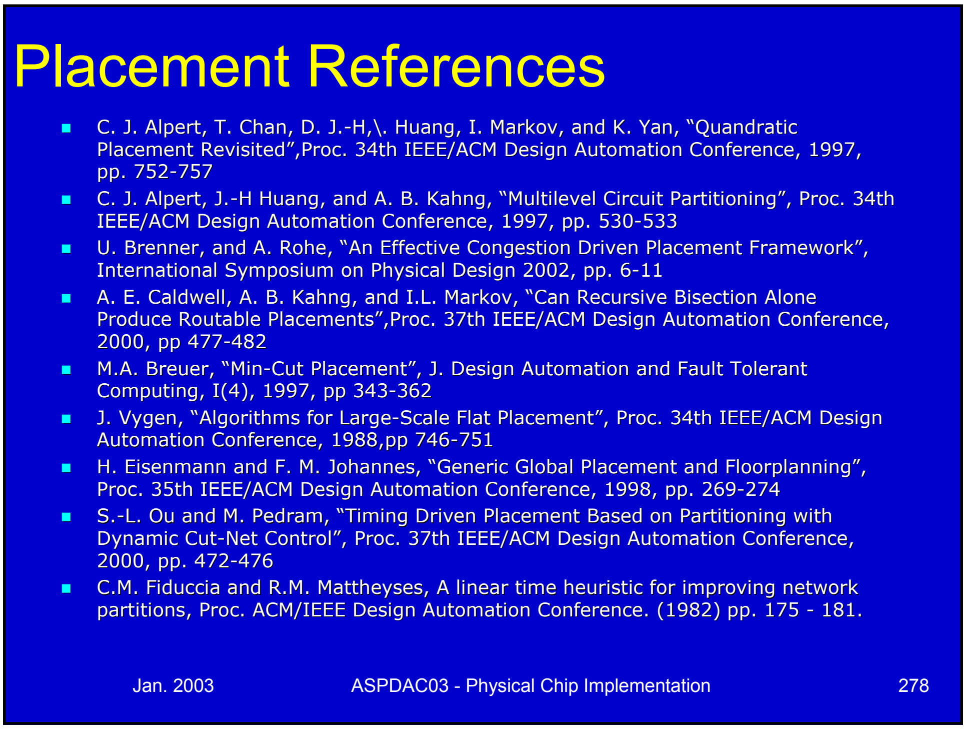
Long Net Spreader





Wrap Up

- Timing closure today is highly dependant on integrated tools.
- Tightly integrated Placement, Timing & Synthesis tools are available today from multiple vendors.
- Placement techniques are dominated quadratic techniques and partitioning
- Next on the list for integration are Routing and Signal integrity tools (happening now)
- These tools have a high degree of complexity. It takes large well funded DA organizations to compete in this space.

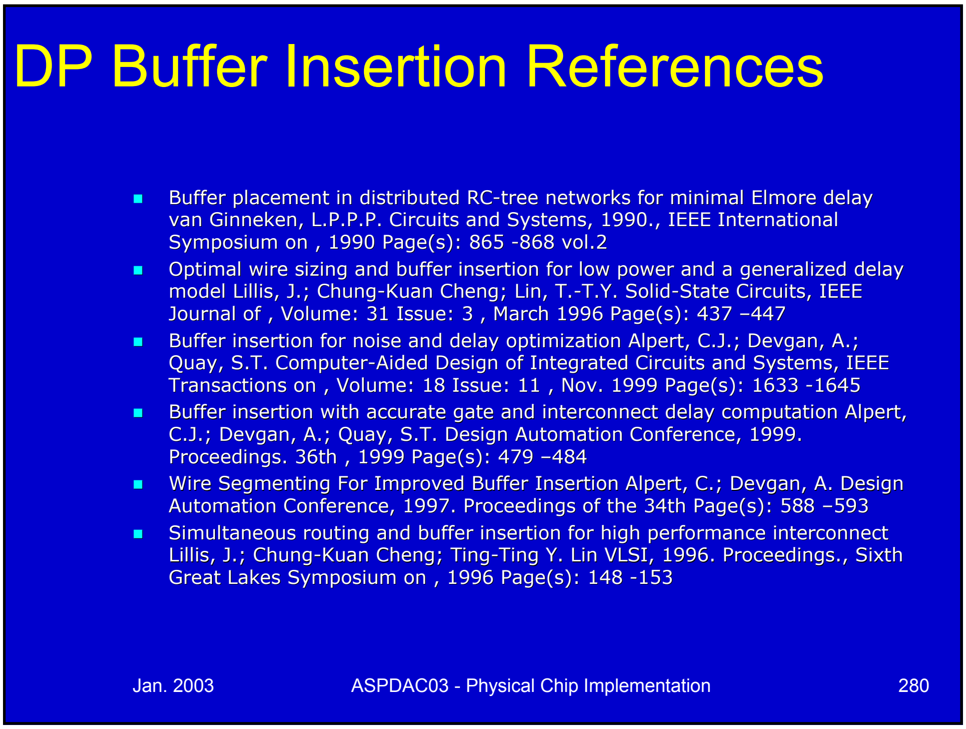


Placement References

- C. J. Alpert, T. Chan, D. J.-H,\. Huang, I. Markov, and K. Yan, “Quandratic Placement Revisited”,Proc. 34th IEEE/ACM Design Automation Conference, 1997, pp. 752-757
- C. J. Alpert, J.-H Huang, and A. B. Kahng, “Multilevel Circuit Partitioning”, Proc. 34th IEEE/ACM Design Automation Conference, 1997, pp. 530-533
- U. Brenner, and A. Rohe, “An Effective Congestion Driven Placement Framework”, International Symposium on Physical Design 2002, pp. 6-11
- A. E. Caldwell, A. B. Kahng, and I.L. Markov, “Can Recursive Bisection Alone Produce Routable Placements”,Proc. 37th IEEE/ACM Design Automation Conference, 2000, pp 477-482
- M.A. Breuer, “Min-Cut Placement”, J. Design Automation and Fault Tolerant Computing, I(4), 1997, pp 343-362
- J. Vygen, “Algorithms for Large-Scale Flat Placement”, Proc. 34th IEEE/ACM Design Automation Conference, 1988,pp 746-751
- H. Eisenmann and F. M. Johannes, “Generic Global Placement and Floorplanning”, Proc. 35th IEEE/ACM Design Automation Conference, 1998, pp. 269-274
- S.-L. Ou and M. Pedram, “Timing Driven Placement Based on Partitioning with Dynamic Cut-Net Control”, Proc. 37th IEEE/ACM Design Automation Conference, 2000, pp. 472-476
- C.M. Fiduccia and R.M. Mattheyses, A linear time heuristic for improving network partitions, Proc. ACM/IEEE Design Automation Conference. (1982) pp. 175 - 181.

Synthesis References

- C.L. Berman, J. L. Carter, and K.F. Day. The Fanout Problem: From Theory to Practice. In Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conference, pages 69-99, 1989
- C. L. Berman, D. J. Hathaway, A. S. LaPaugh, and L. H. Trevillyan. Efficient Techniques for Timing Corrections. In International Symposium on Circuits and Systems, Pages 415-419, 1990
- F. Beefting, P. N. Kudva, D. S. Kung, R. Puri, and L. Stok. Combinatorial Cell Design for CMOS Libraries INTEGRATION, the VLSI Journal, 29:67-93, 2000
- W. Donath, P. Kudva, L. Stok, P. Villarrubia, L. Reddy, and A. Sullivan. Transformational placement and synthesis. In DATE, pages 194-201, 2000
- D. J. Hathaway, R.P. Abato, A.D. Drumm, and L.P.P.P . Van Ginneken. Incremental timing analysis. Technical report, IBM Corp., 1996. U.S. patent 5,508,937.
- D. Kung, P. Kudva, and A. Sullivan. A Gate Sizing Algorithm using Geometric Programming. In Proc. Of the International Workshop on Logic Synthesis, 1997
- T. Kutzschebauch and L. Stok. Regularity driven logic synthesis. In Proc of the Int. Conf. On Computer Aided Design, Nov 2000.
- P. Rezvani, A.H. Ajami, M. Pedram, and H. Savoj. LEOPARD: A Logical Effort based fanout Optimizer for Area and Delay. In IEEE/ACM International Conference on CAD, pages 516-519, 1999.
- L. Stok, M. Iyer, and A. Sullivan. Wavefront technology mapping. In DATE, pages 531-536, 1999
- D. S. Kung. A Fast Fanout Optimization for New-Continuous Buffer Libraries. In IEEE/ACM Design Automation Conference, pages 352-355, 1998



DP Buffer Insertion References

- Buffer placement in distributed RC-tree networks for minimal Elmore delay van Ginneken, L.P.P.P. Circuits and Systems, 1990., IEEE International Symposium on , 1990 Page(s): 865 -868 vol.2
- Optimal wire sizing and buffer insertion for low power and a generalized delay model Lillis, J.; Chung-Kuan Cheng; Lin, T.-T.Y. Solid-State Circuits, IEEE Journal of , Volume: 31 Issue: 3 , March 1996 Page(s): 437 –447
- Buffer insertion for noise and delay optimization Alpert, C.J.; Devgan, A.; Quay, S.T. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 18 Issue: 11 , Nov. 1999 Page(s): 1633 -1645
- Buffer insertion with accurate gate and interconnect delay computation Alpert, C.J.; Devgan, A.; Quay, S.T. Design Automation Conference, 1999. Proceedings. 36th , 1999 Page(s): 479 –484
- Wire Segmenting For Improved Buffer Insertion Alpert, C.; Devgan, A. Design Automation Conference, 1997. Proceedings of the 34th Page(s): 588 –593
- Simultaneous routing and buffer insertion for high performance interconnect Lillis, J.; Chung-Kuan Cheng; Ting-Ting Y. Lin VLSI, 1996. Proceedings., Sixth Great Lakes Symposium on , 1996 Page(s): 148 -153

Blockage Avoidance References

- Steiner tree optimization for buffers, blockages, and bays Alpert, C.J.; Gandham, G.; Jiang Hu; Neves, J.I.; Quay, S.T.; Sapatnekar, S.S. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 20 Issue: 4 , April 2001 Page(s): 556 –562.
- A fast algorithm for context-aware buffer insertion Jagannathan, A.; Sung-Woo Hur; Lillis, J. Design Automation Conference, 2000. Proceedings 2000 Page(s): 368 –373.
- Simultaneous routing and buffer insertion with restrictions on buffer locations Hai Zhou; Wong, D.F.; I-Min Liu; Aziz, A. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 19 Issue: 7 , July 2000 Page(s): 819 -824
- Maze routing with buffer insertion and wire sizing Minghorng Lai; Wong, D.F. Design Automation Conference, 2000. Proceedings 2000 Page(s): 374 -378
- Routing tree construction under fixed buffer locations Cong, J.; Xin Yuan Design Automation Conference, 2000. Proceedings 2000 Page(s): 379 -384

Interconnect Planning References

- A practical methodology for early buffer and wire resource allocation Alpert, C.J.; Jiang Hu; Sapatnekar, S.S.; Villarrubia, P.G. Design Automation Conference, 2001. Proceedings , 2001 Page(s): 189 –194
- An interconnect-centric design flow for nanometer technologies Cong, J. Proceedings of the IEEE , Volume: 89 Issue: 4 , April 2001 Page(s): 505 -528
- Buffer block planning for interconnect-driven floorplanning Cong, J.; Tianming Kong; Pan, D.Z. Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on , 1999 Page(s): 358 –363
- Provably good global buffering using an available buffer block plan Dragan, F.F.; Kahng, A.B.; Mandoiu, I.; Muddu, S.; Zelikovsky, A. Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on , 2000 Page(s): 104 -109
- Provably good global buffering by multiterminal multicommodity flow approximation Dragan, F.F.; Kahng, A.B.; Mandoiu, I.; Muddu, S.; Zelikovsky, A. Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific , 2001 Page(s): 120 –125
- Planning buffer locations by network flows Tang, X.; Wong, D.F.; International Symposium on Physical Design, April 2001 Page(s): 180-185
- Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning Sarkar, P.; Sundararaman, V.; Koh, C.-K.; International Symposium on Physical Design, April 2001 Page(s): 186-191