Automatic Layout of Analog and Digital Mixed Macro/Standard Cell Integrated Circuits

William Swartz

Yale University

Outline

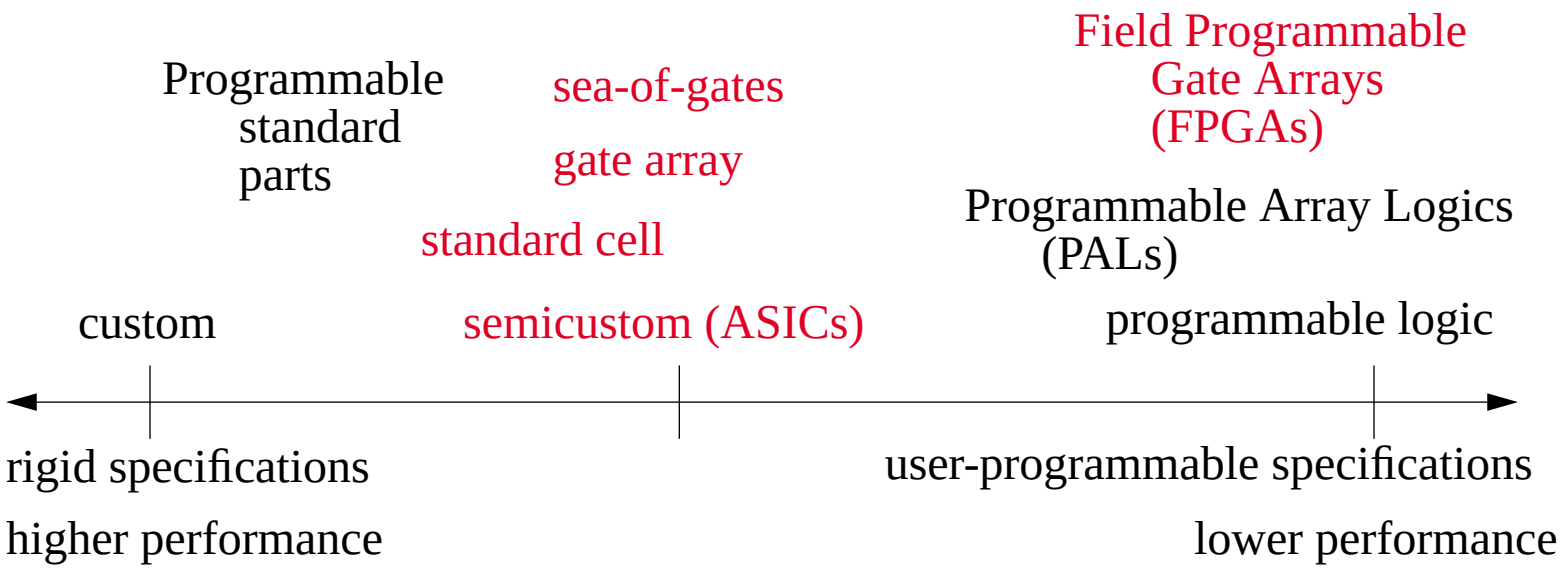
- Introduction
- Mixed Design Flow
- New Placement Techniques in TimberWolf
  - Statistical Wire Estimation
  - Timing Driven Placement
  - Analog Crosstalk Minimization
- New Global Routing Techniques
- Results
- Conclusions

Why Integrated Circuits?

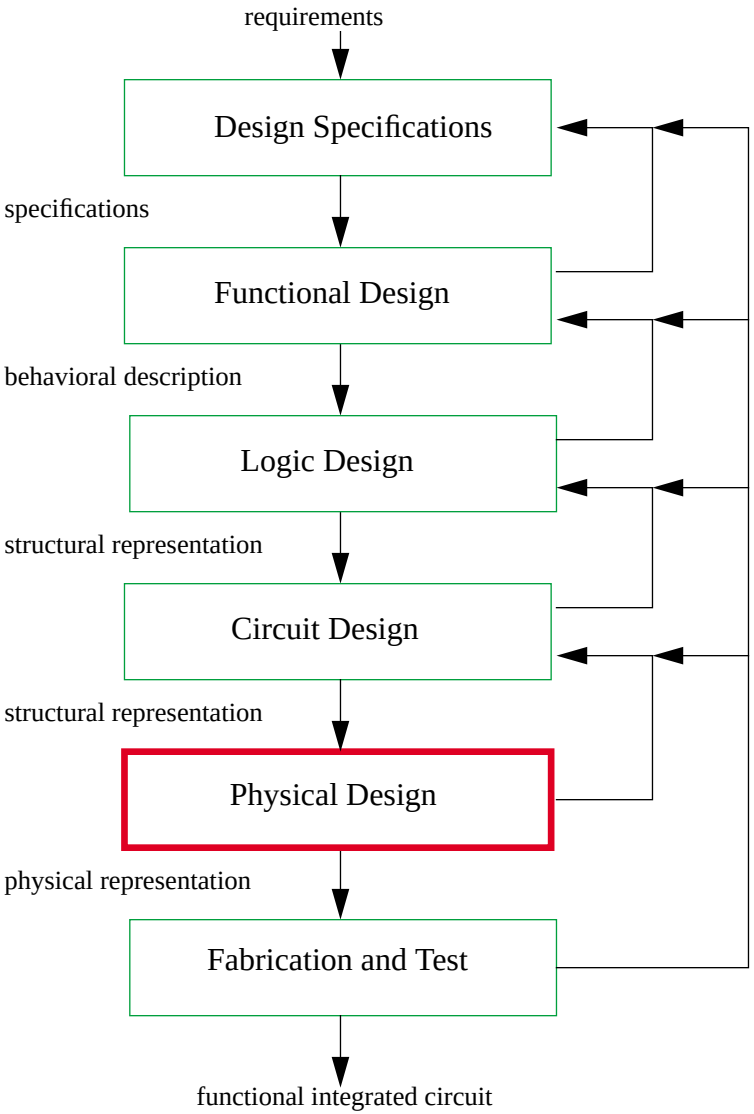
- System integration and consolidation
- Weight and size reduction
- Increased reliability
- Component matching

☛ Cost reduction

Spectrum of Design Methodologies for ICs



Designing Integrated Circuits

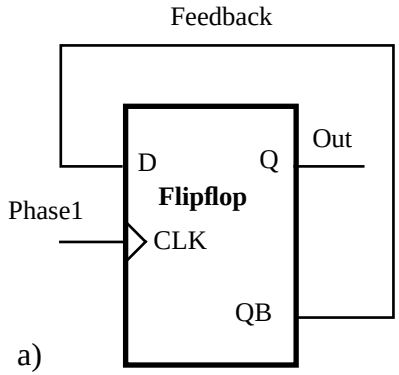


Physical Design Transformation

Schematic Representation

Textual Representation

Logical or Circuit Level

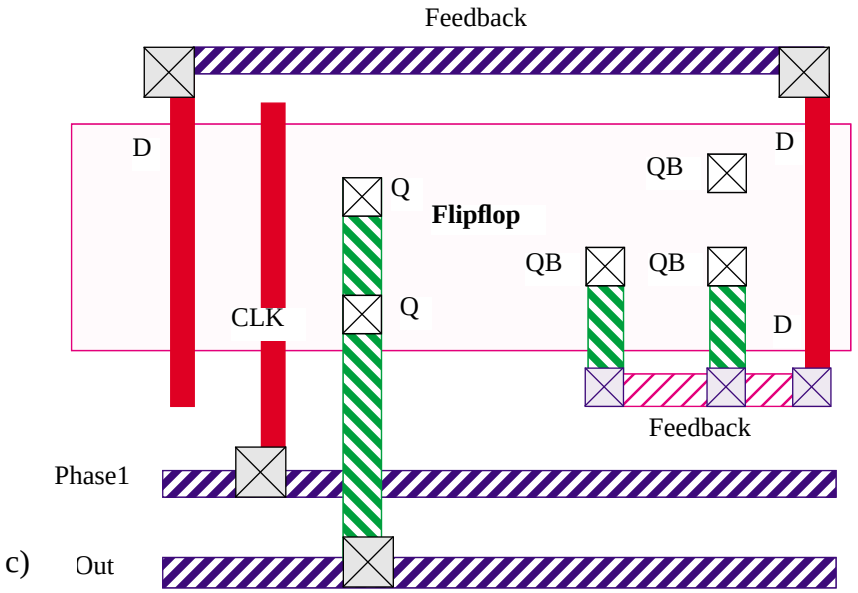


a)

```
INSTANCE 1 Flipflop
    Phase1, Feedback, Out, Feedback
                .
                .
                .
CELL FlipFlop
    CLK, D, Q, QB
```

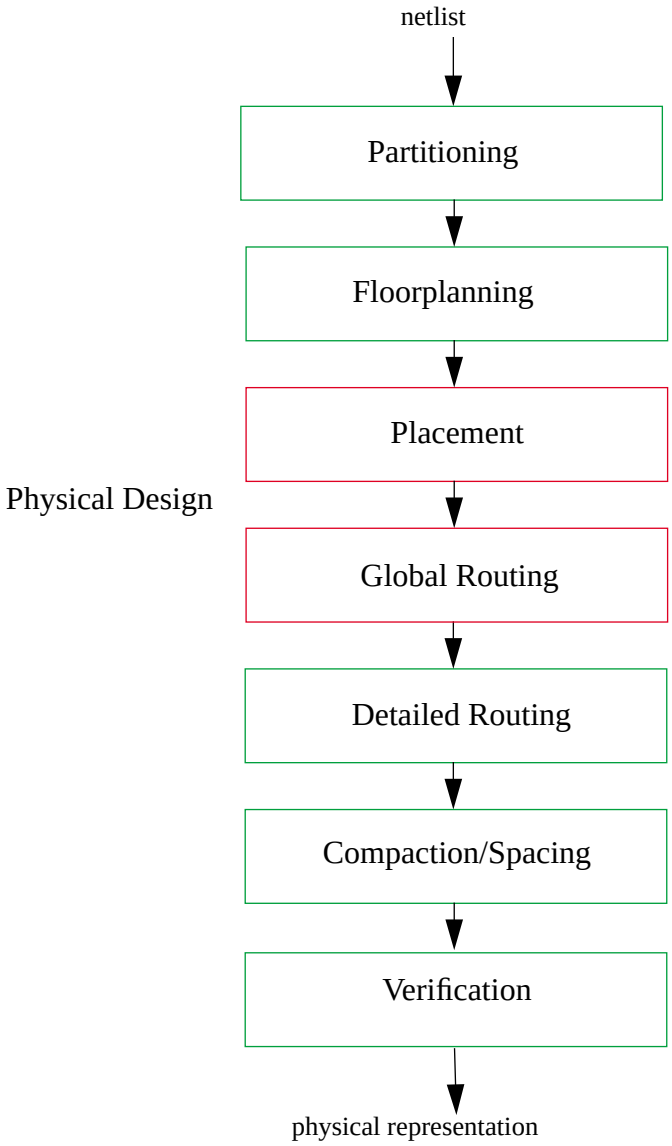
b)

Physical Level

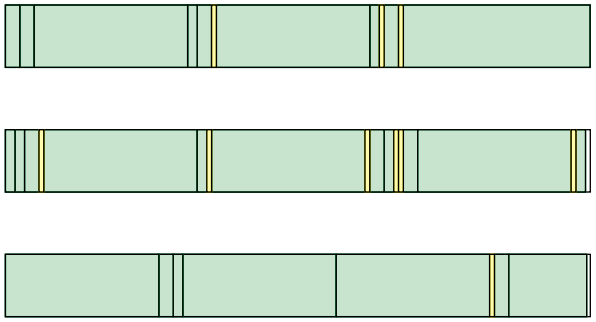


c)

The Physical Design Stages

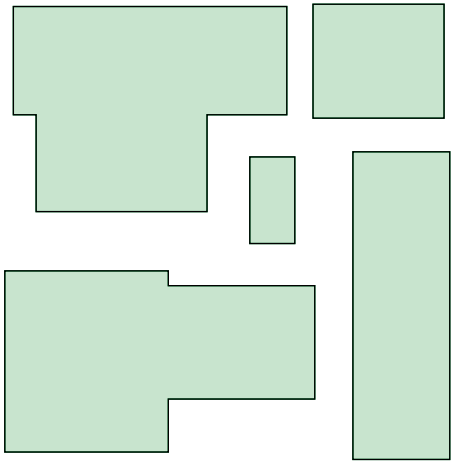


Design Styles



Standard cell design style

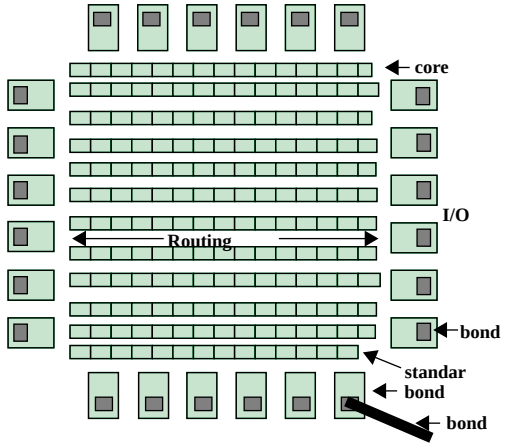
Suits random logic



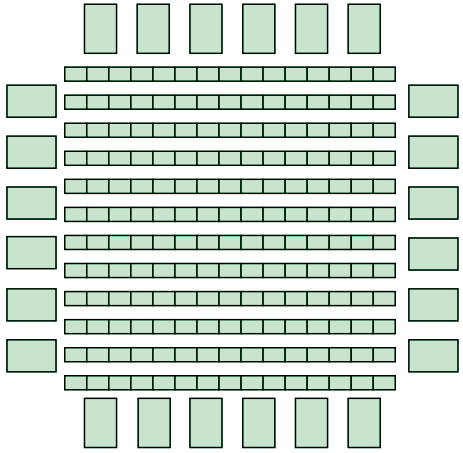
Macro cell design style

Suits array architectures

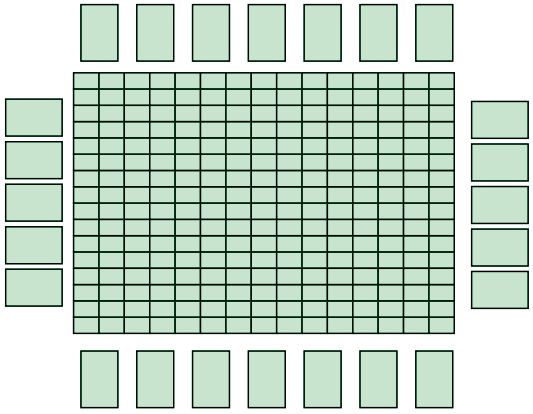
Row-based Design Methodologies



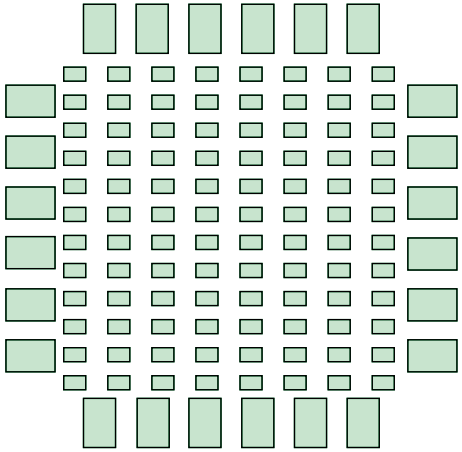
Standard Cell



Gate Array

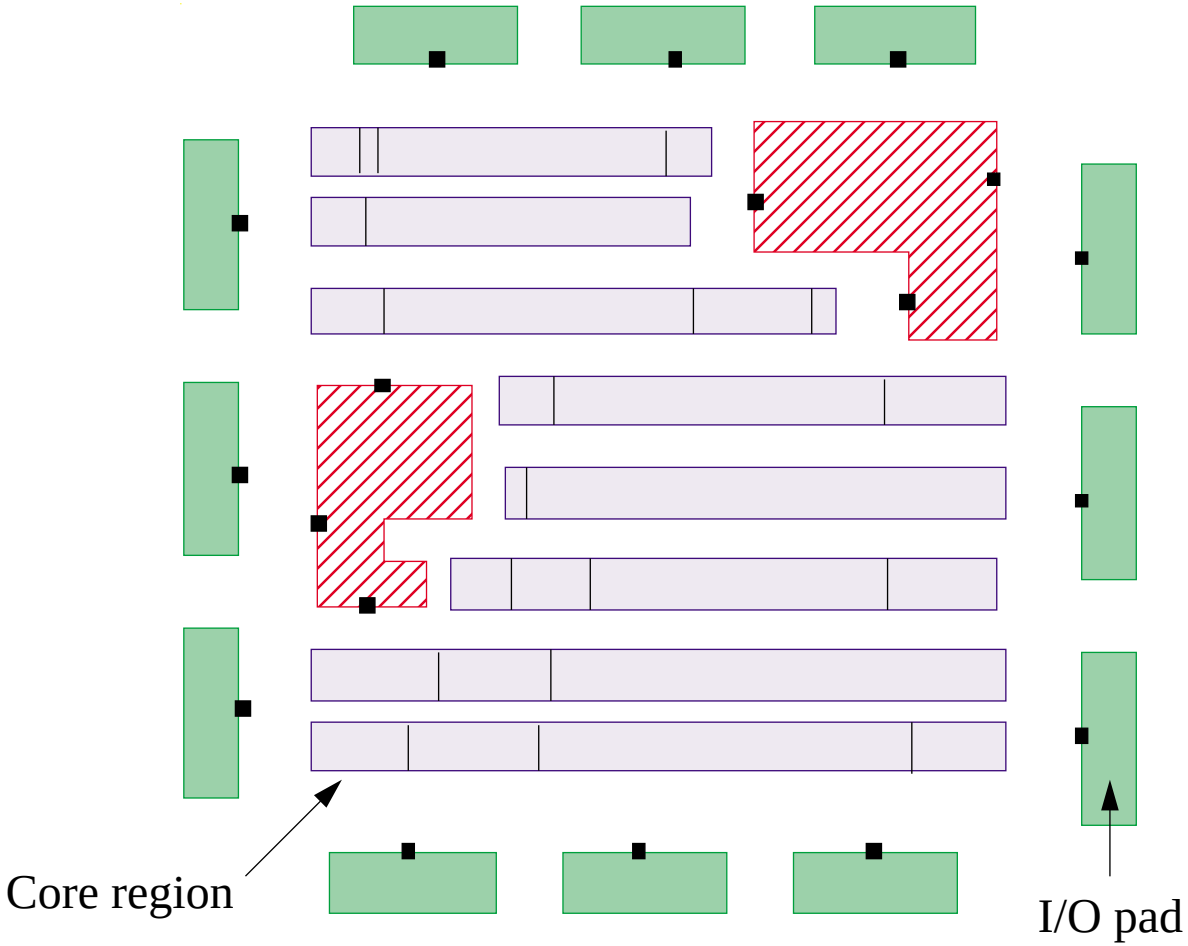


Sea-of-Gates

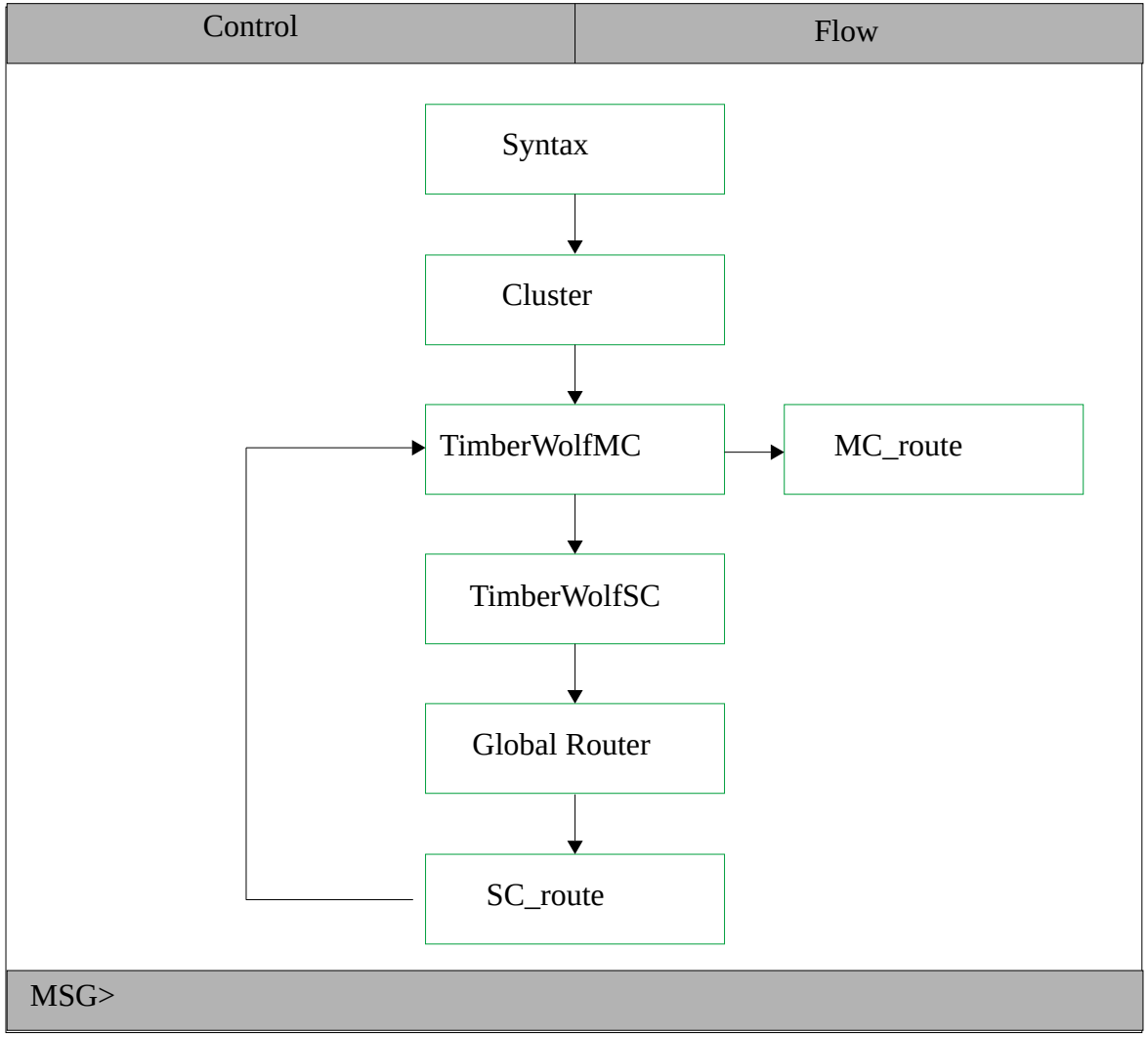


Island style gate array

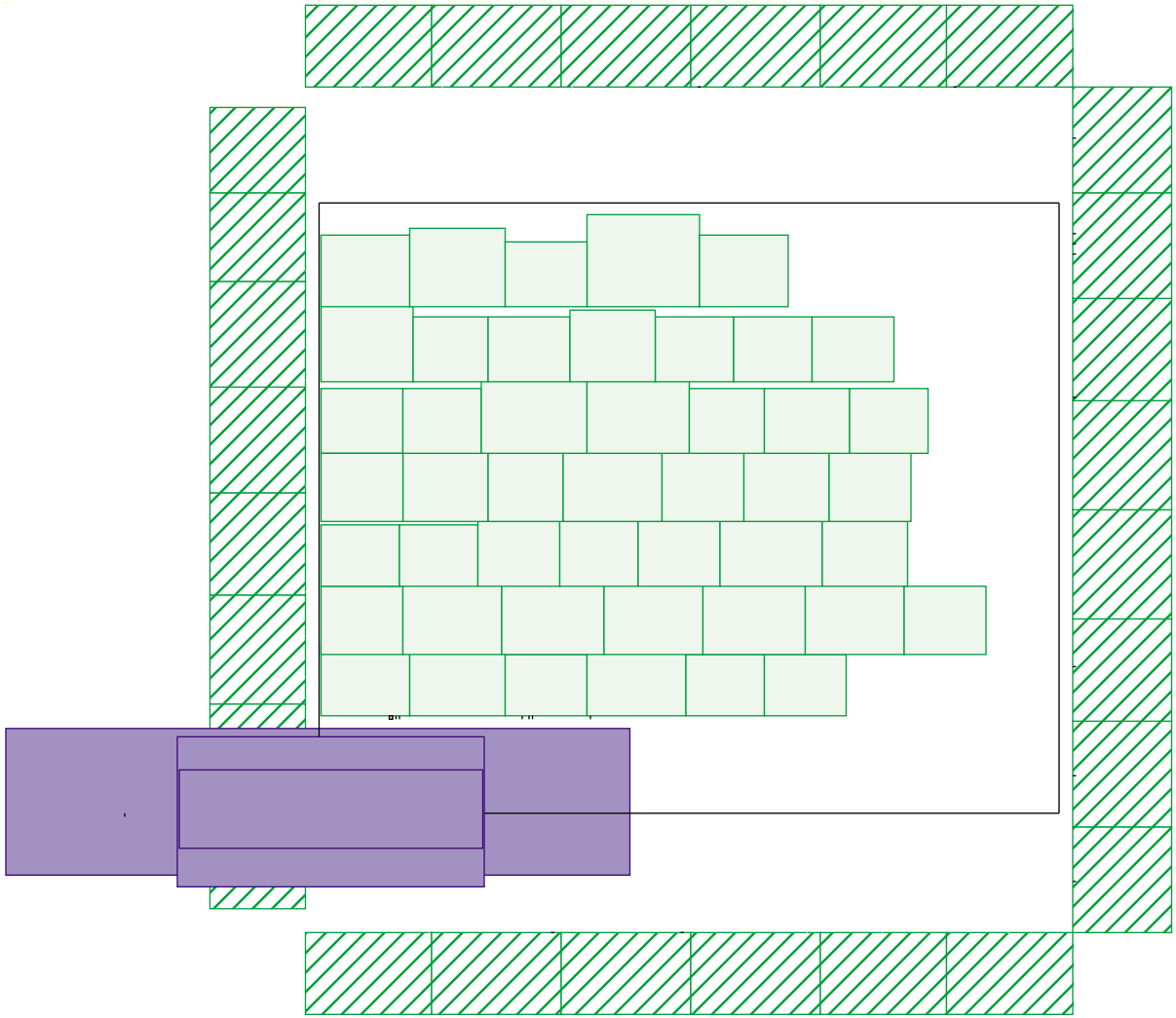
Mixed Design Style



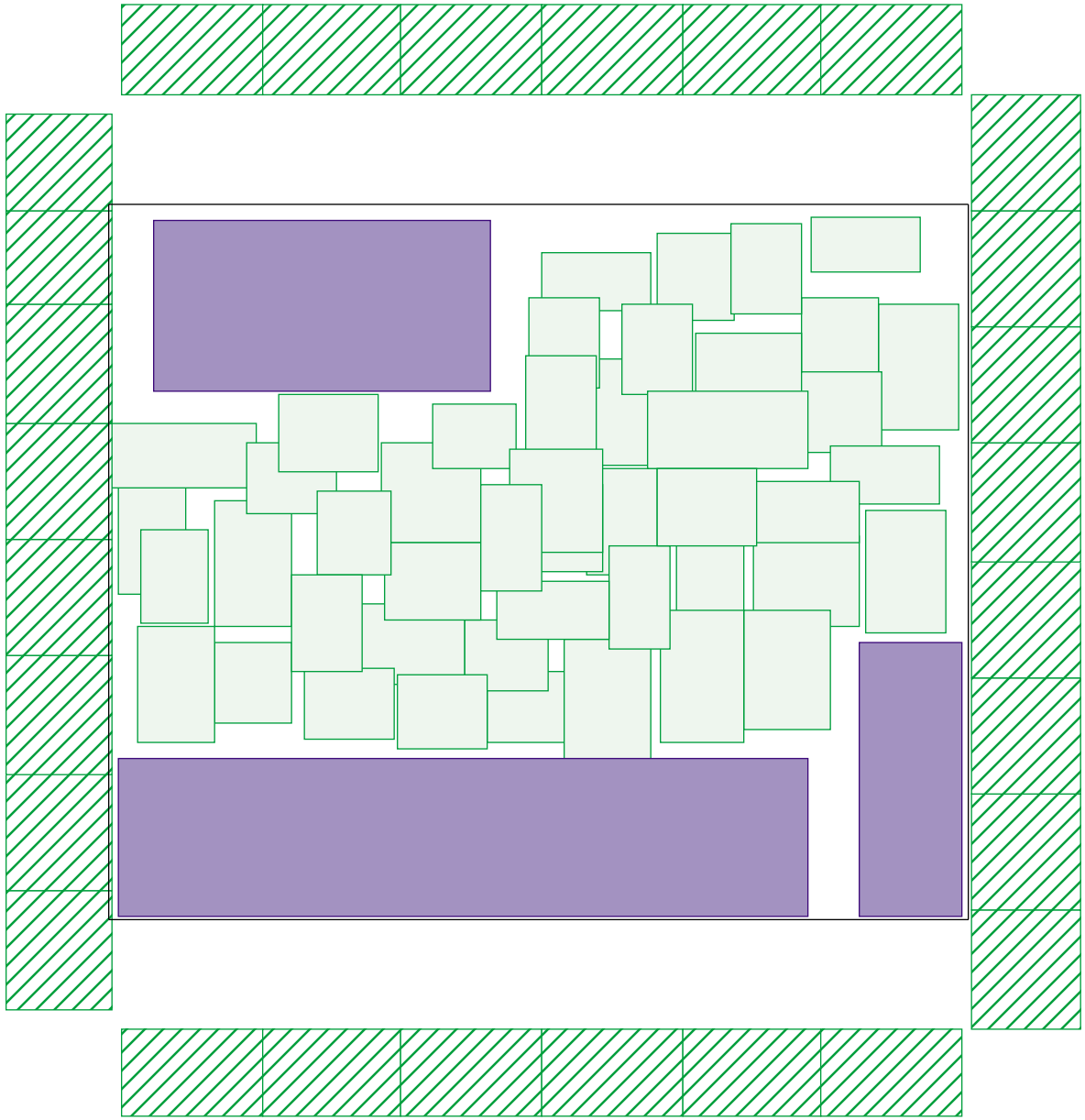
Tour of Mixed Macro / Standard Cell Physical Design



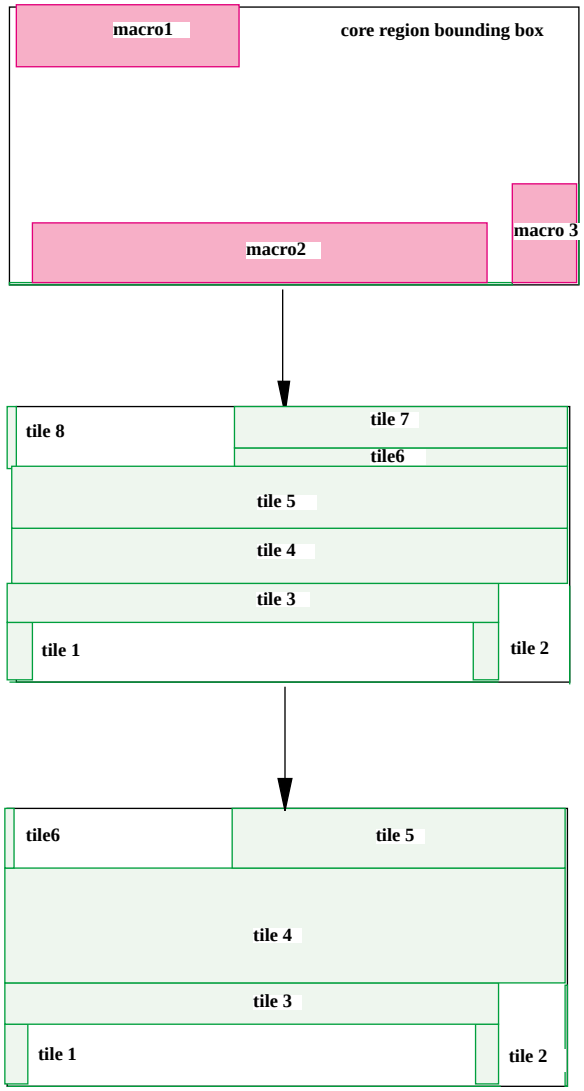
Cluster : Partitioning



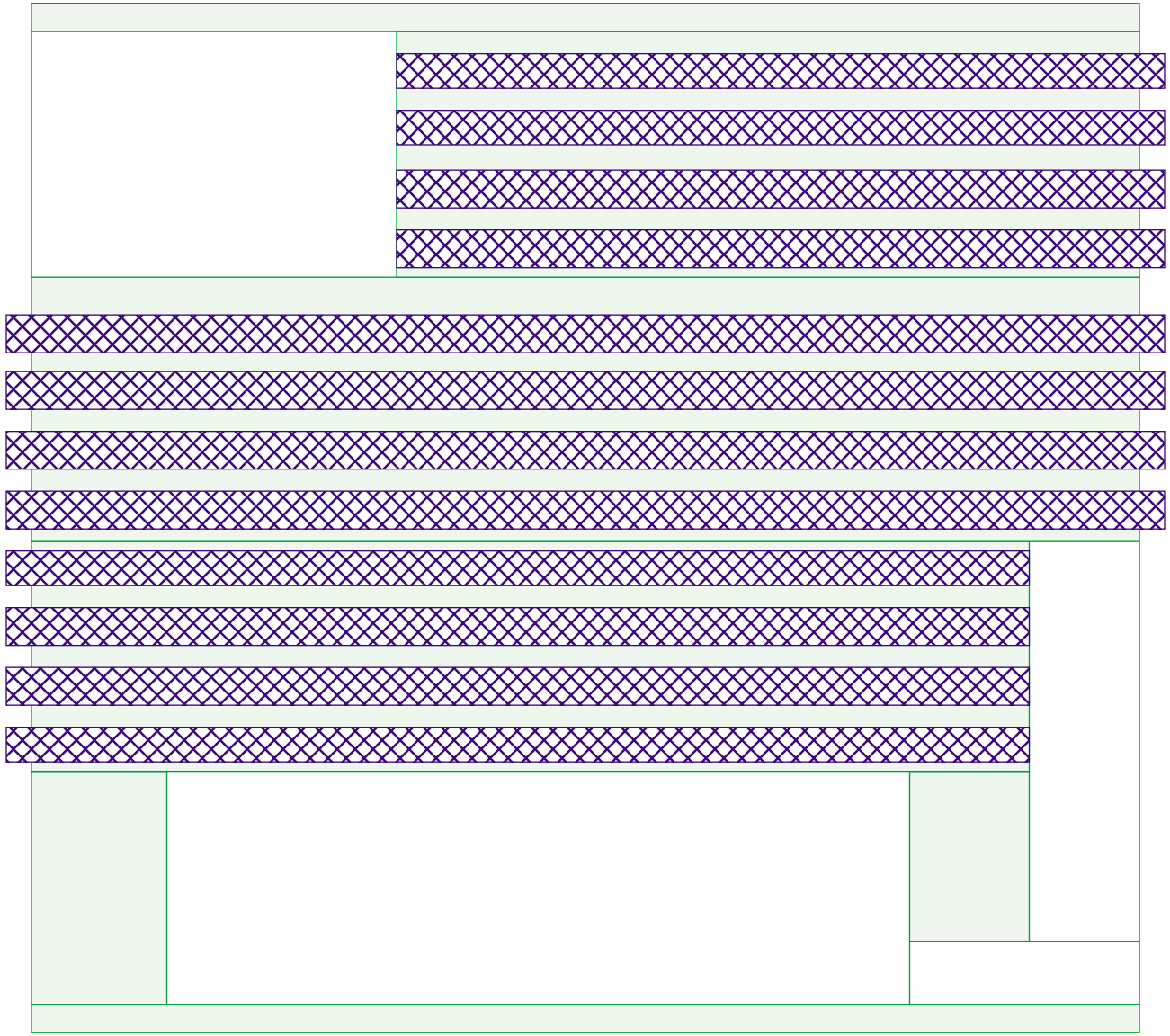
TimberWolfMC : Floorplanning



Genrows : Core region floorplanning



Genrows : Row topology generation

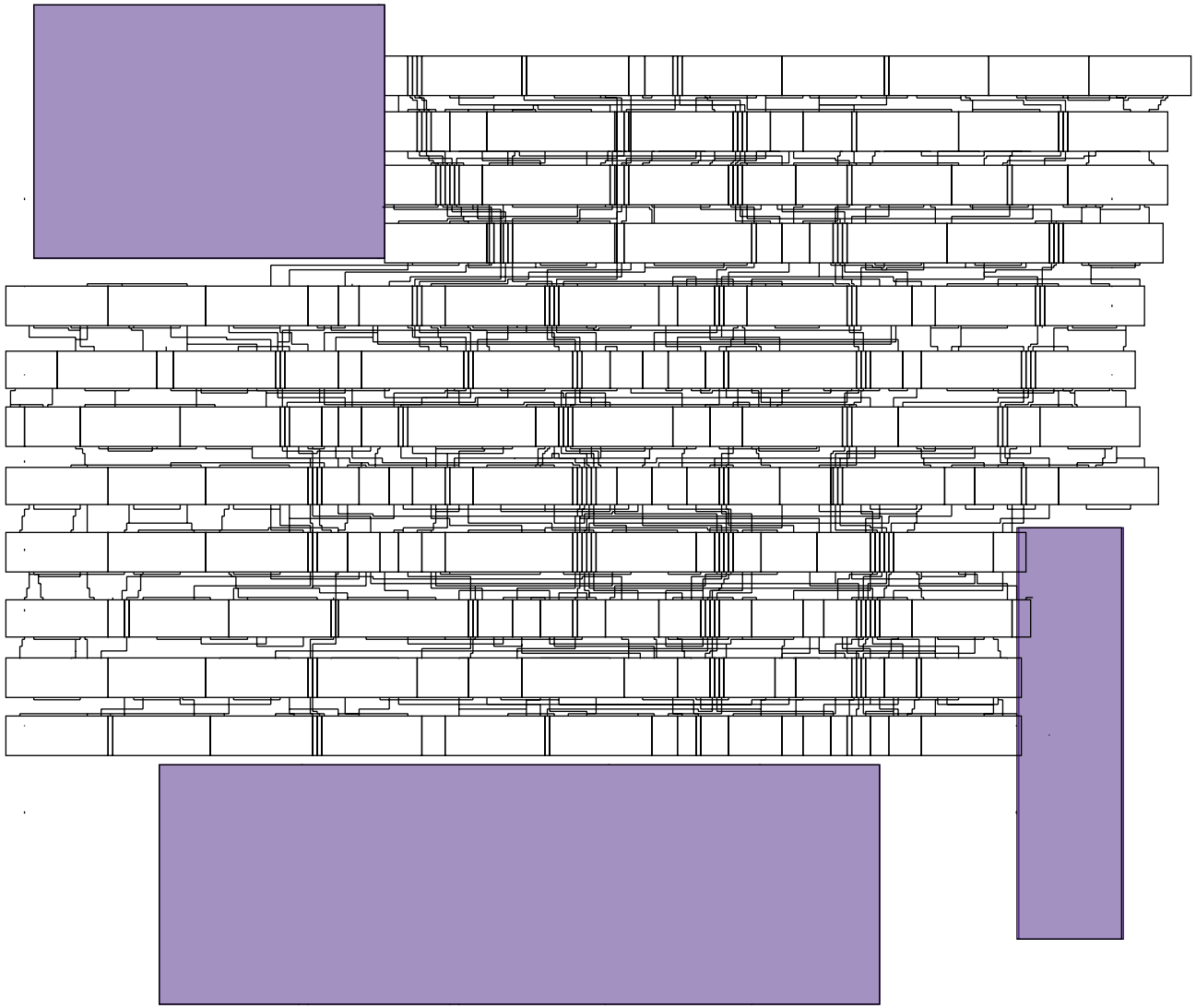


TimberWolfSC : Placement and global routing

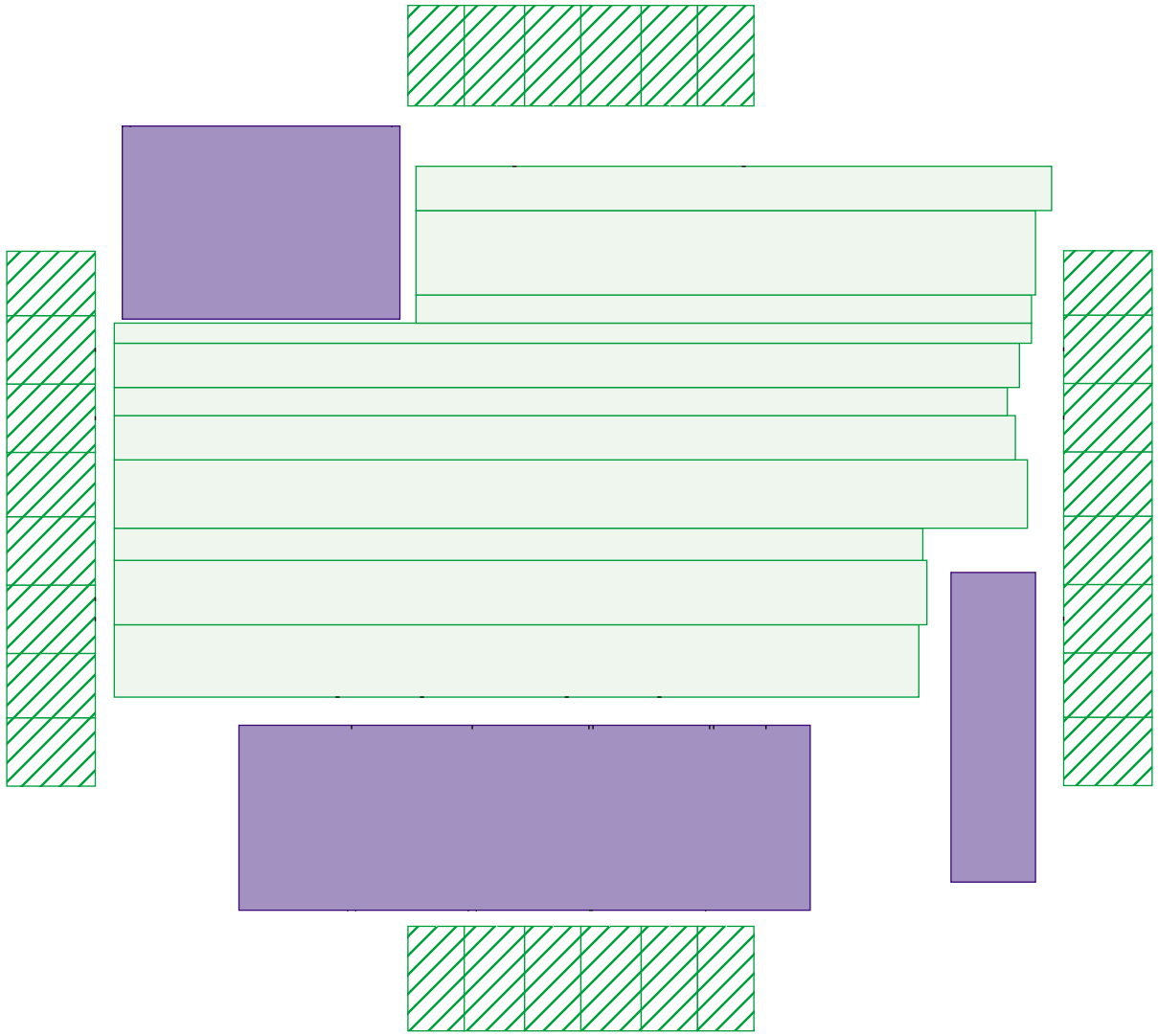
- Placement - determine the position of individual cells given a cost function.
  - Wire length
  - Timing constraints

- Global routing - determine the interconnections of the signal network.
  - The network is decomposed into net segments.
  - The region for each net segment is calculated.

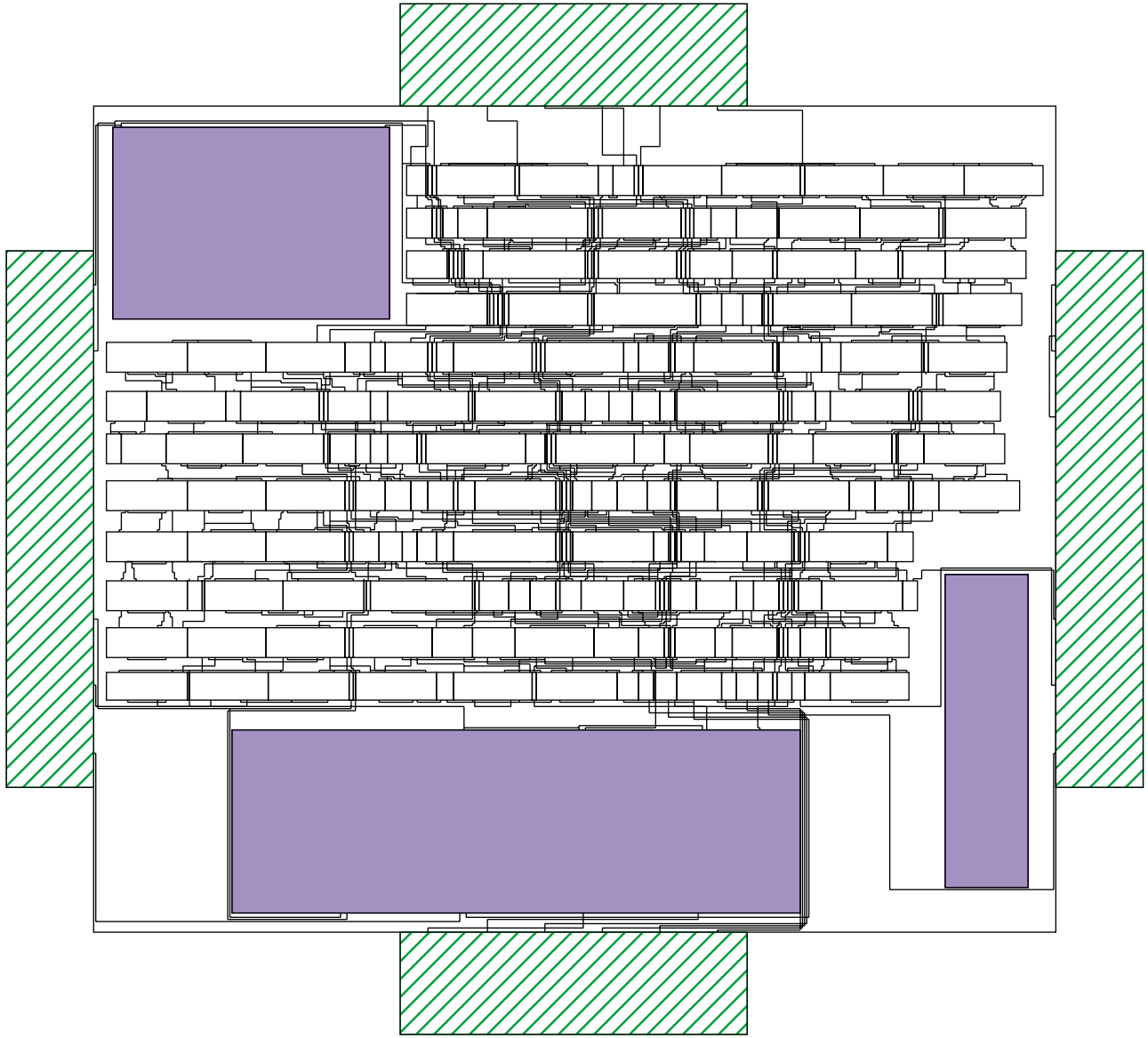
Sc_route : Standard cell detailed-routing



TimberWolfMC : Placement refinement (compaction)



Mc_route : Final detailed-route



Need Efficient Algorithms

complexity	n=20	n=50	n=100	n=200	n=500	n=1000
$1000n^*$	0.02 sec	0.05 sec	0.1 sec	0.2 sec	0.5 sec	1 sec
$1000nlgn^*$	0.09 sec	0.3 sec	0.6 sec	1.5 sec	4.5 sec	10 sec
$100n^{2*}$	0.04 sec	0.25 sec	1 sec	4 sec	25 sec	2 min
$10n^{3*}$	0.02 sec	1 sec	10 sec	1 min	21 min	2.7 hr
n^{lgn}	0.04 sec	1.1 hr	220 days	125 cent	5×10^8cent	
$2^{n/3}$	0.0001 sec	0.1 sec	2.7 hr	3×10^4cent		
2^n	1 sec	35 yr	3×10^4cent			
3^n	58 min	2×10^9cent				

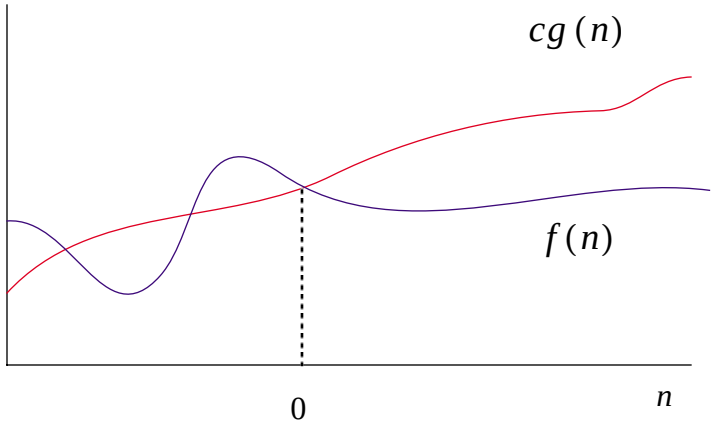
"Big-*O*" notation

- *O*-notation gives an upper bound of a function within a constant factor.

For a given function $g(n)$ we denote by $O(g(n))$ the set of functions

$O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that

$0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$.



Basic Simulated Annealing Algorithm

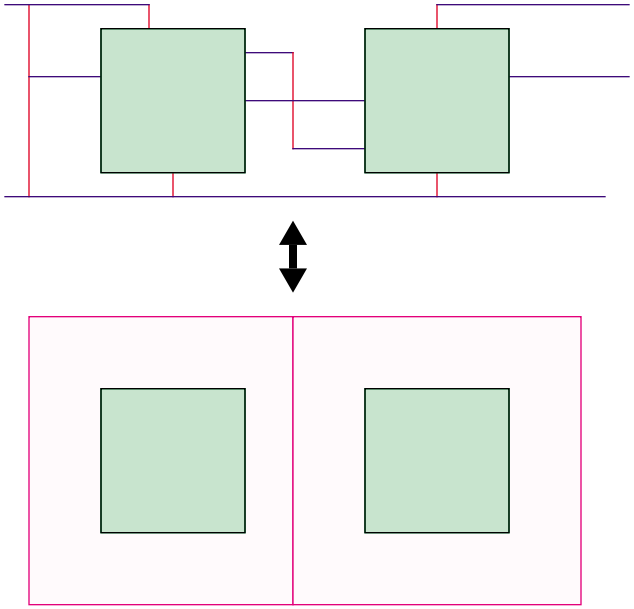
Algorithm simulated_annealing()

```
1   do
2       do
3           j = generate(i)
4           if accept(ΔC, T) then
5               i = j
6       until cost is in equilibrium
7       reduce(T)
8   until cost cannot be reduced any further
```

Algorithm accept(Δ*C*, *T*)

```
1   if ΔC ≤ 0 then /* new cost is less than or equal to the old cost */
2       return(ACCEPT) /* accept the new configuration */
3    else
4       randomly generate a number r between 0 and 1
5       if r < xp(−ΔC / T) then return(ACCEPT)
6       else return(REJECT)
```

New Placement Techniques-Wire Area Estimation



- Routing interconnect between adjacent macro cells is modeled as extra area appended to each of the corresponding cells during simulated annealing placement.
- Wiring area between macros is estimated during placement in order to avoid large perturbations in the topology during detailed-routing and compaction.

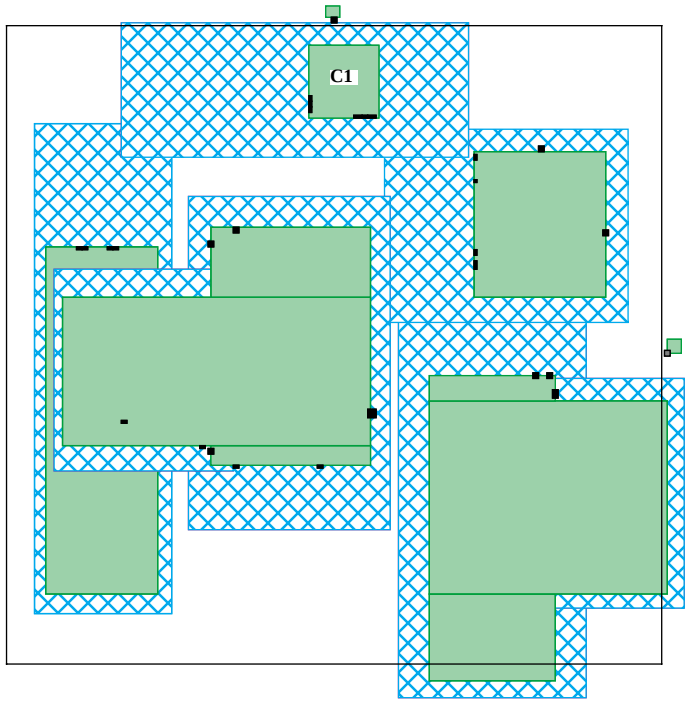
Problems with Existing Wire Estimators

- Existing wire estimators are theoretical models.
- Problematic if the design style violates any of the assumptions of the theoretical model.
- For example, model must be changed if another routing layer is added.

Inaccurate Modeling

Cell area

Routing Estimate



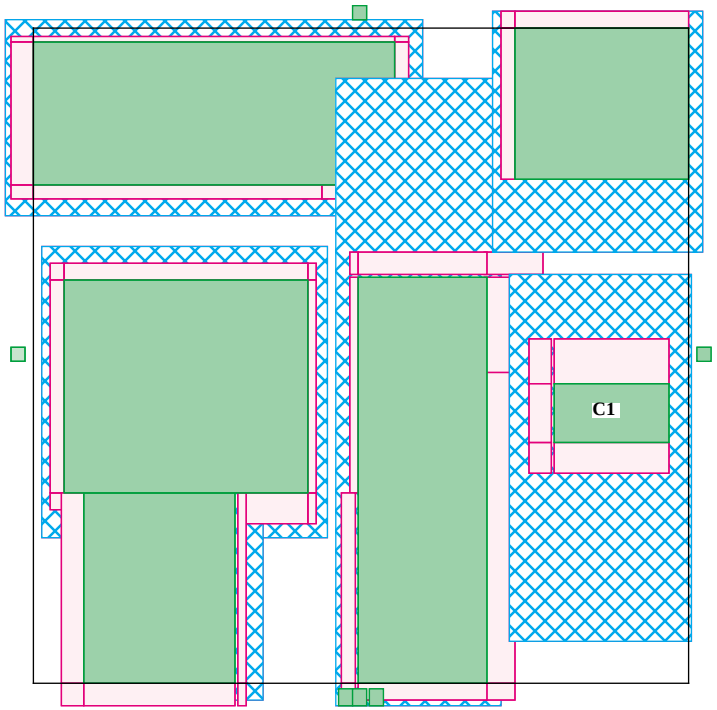
- TimberWolfMC version 1 overestimates the routing area needed for cell C1.

Resulting Placement using Previous Wire Estimator

Cell area

Routing Estimate

Actual Routing

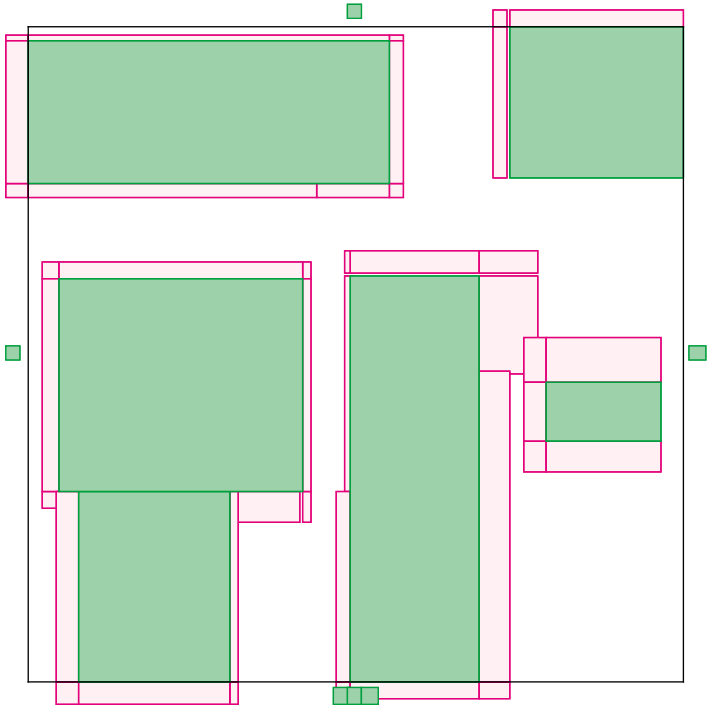


- Placement after global routing using original wire estimator. Notice the gross inaccuracy in the estimation of cell C1.

Inaccurate Wiring Estimates Lead to Poor Area Efficiency

Cell area

Actual Routing



- White space around cells is unused area.

Solution: Statistical Wire Estimator

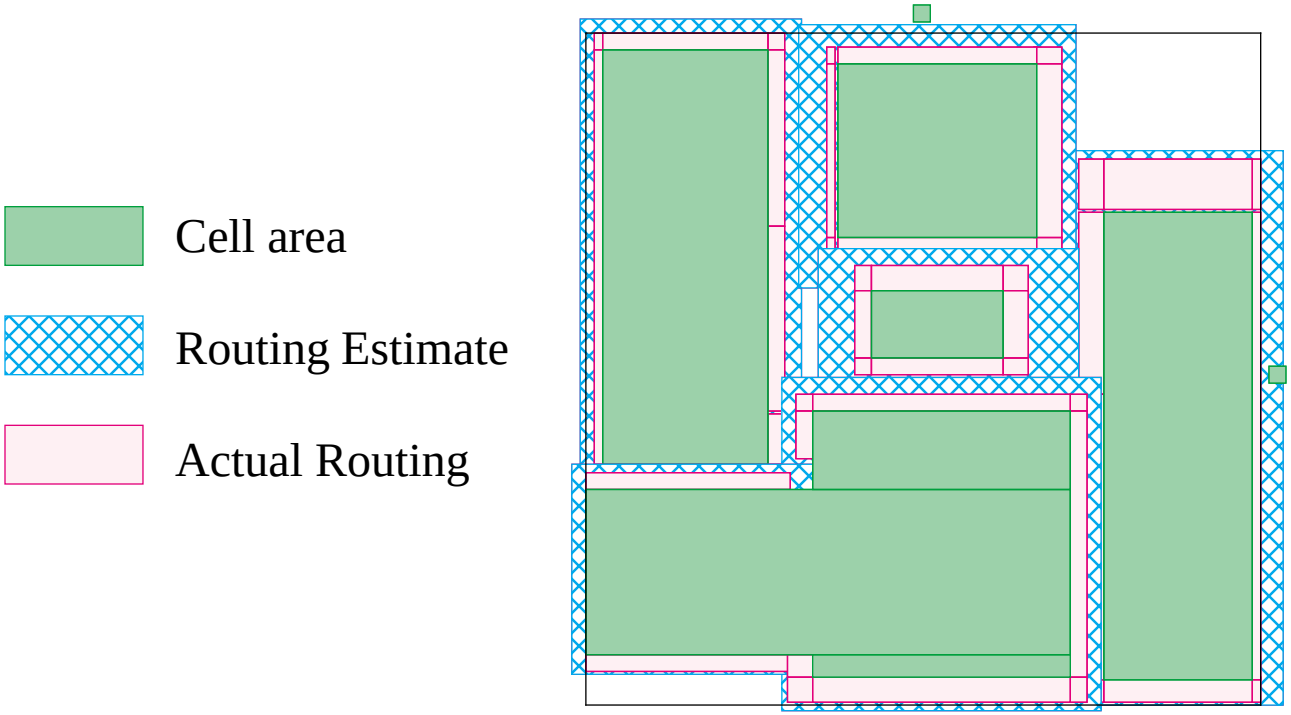
- Use a general statistical model which is adapted for every circuit.
- Define the estimated interconnect area for cell edge *i* to be

$$i = c_0 + c_1 \cdot x + c_2 \cdot x^2 + c_3 \cdot y + c_4 \cdot y^2 + c_5 \cdot p \qquad (1)$$

where $c_0 ... c_5$ are constants, x and y are the normalized chip coordinates [0.0, 1.0], and p is the number of pins in the channel.

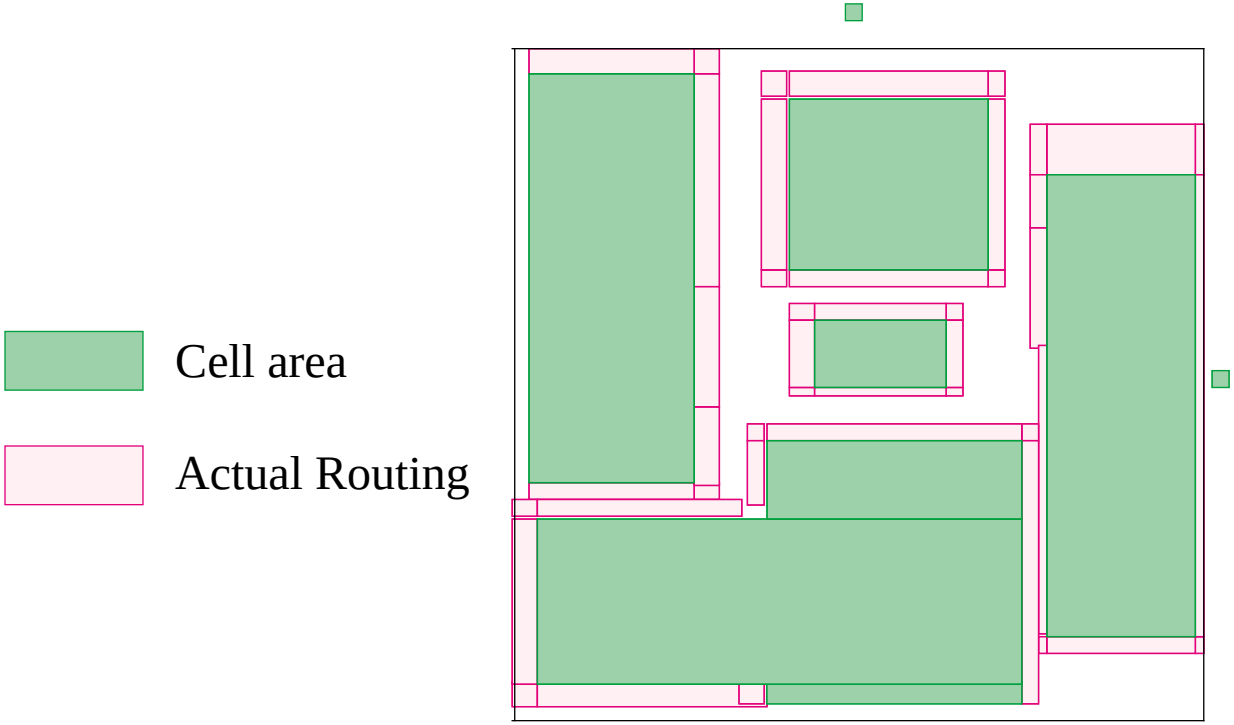
- To obtain the model constants:
  - Place the circuit using a 10*x* annealing schedule and the theoretical estimation model.
  - Perform global routing and/or detail routing to calculate routing areas.
  - Use a least squares method to fit the data to estimator model.
- Subsequent placements are performed using the statistical model.
- Placement algorithm *learns* from the previous executions.
- The statistical model adapts to any design style and routing technology.

Placement Using Statistical Estimator



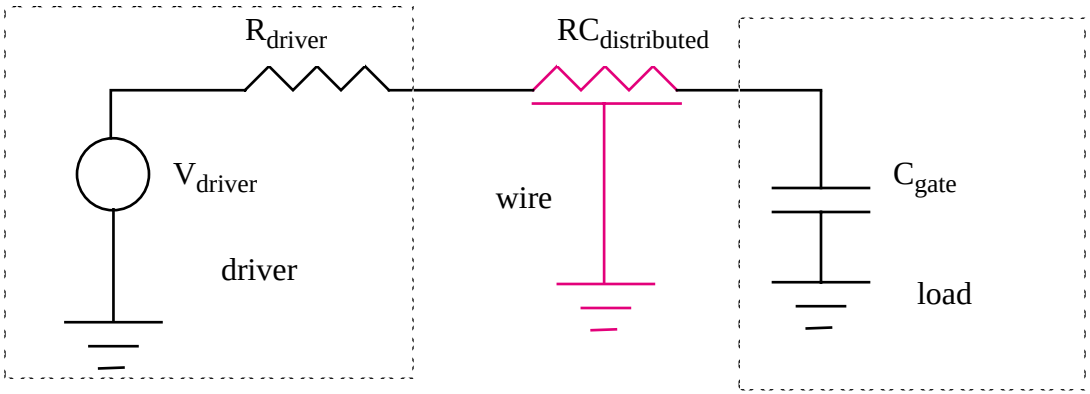
- Placement after global routing using statistical wire estimator.

Accurate Wire Estimation



- This is the minimum area placement for this example.
- The remaining white space does not impact chip area.

New Placement Techniques - Timing Constraints



The parasitic capacitance of a wire segment is

$$C = \frac{\varepsilon A}{d} = \frac{\varepsilon l w}{t} \Rightarrow C \propto l \tag{2}$$

The resistance of a wire segment is given by

$$R = \frac{\rho l}{A} = R_s \cdot \frac{l}{w} \Rightarrow R \propto l \tag{3}$$

Timing Driven Placement

TimberWolf supports the following types of timing constraints:

- critical path using wire length constraints.
- matched critical path using wire length constraints.
- critical path using timing constraints.
- matched critical path using timing constraints.
- critical path analysis using pin pair constraints.
- matched critical path analysis using pin pair constraints.

Critical path using wire length constraints

- Implemented first in TimberWolf version 5.6.
- The penalty assigned for a path p is the amount the length deviates from satisfying the bounds:

$$p = \begin{cases} \text{length}(p) - \text{upperBound}(p) & \text{if} \quad \text{length}(p) > \text{upperBound}(p) \\ \text{lowerBound}(p) - \text{length}(p) & \text{if} \quad \text{length}(p) < \text{lowerBound}(p) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where the length of a path p is the sum of the half perimeter wire length of all the nets n in the path:

$$\text{length}(p) = \sum_{\forall n \in p} S_x(n) + S_y(n) \tag{5}$$

Wire length Constraints

- The total penalty is just the sum over all the specified critical paths:

$$T = \sum_{p=1}^{N_p} P_p \qquad (6)$$

- Does not take drive strength into account.

Matched Wire Length Constraints

- The user specifies a tolerance for the mismatch in path length. In this case, we assign the penalty for a set of paths to be:

$$P_p = \begin{cases} \text{match}(p) - \text{tolerance}(p) & \text{if} \quad \text{match}(p) > \text{tolerance}(p) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where the match is defined as

$$\text{match}(p) = \left| \text{length}(p_1) - \text{length}(p_2) \right| \tag{8}$$

- Useful for analog circuits.

Critical Path Using Timing Constraints

- Time constraints which consider driver strength.
- The arrival time T_a for a path p is the summation of all the net delays $_n$ for the path:

$$_a = \sum_{\forall n \in p} D_n \tag{9}$$

- The delay for a single net n is the sum of the intrinsic gate delay T_n associated with the driver of the net n, and the product of the equivalent driver resistance R_n, and the total load capacitance seen by the driver:

$$D_n = T_n + R_n C_n \tag{10}$$

Parasitics

- The total capacitance for a net has two components: gate input capacitance and parasitic capacitance.

$$C_n = C_{G_n} + C_{p_n} \tag{11}$$

- During placement, we can estimate the parasitic capacitance using the half perimeter bounding box metric:

$$C_{n_p} = C_{L_x} S_x(n) + C_{L_y} S_y(n) \tag{12}$$

Arrival Time Equation

- Substituting Equation 8 and Equation 9 into Equation 10, we get:

$$D_n = T_n + R_n C_G + R_n \left[C_{L_x} S_x(n) + C_{L_y} S_y(n) \right] \quad (13)$$

- We can precompute the terms in the summation which do not depend on wire length by defining:

$$= \sum_{\forall n \in p} [T_n + R_n C_G] \quad (14)$$

- This results in the following simplified equation for the arrival time:

$$T_a = \sum_{\forall n \in p} R_n \left[C_{L_x} S_x(n) + C_{L_y} S_y(n) \right] + K \quad (15)$$

Penalty for Critical Path Constraints

$$= \begin{cases} T_a(p) - T_{ru}(p) & \text{if} \quad T_a(p) > T_{ru}(p \\ T_{rl}(p) - T_a(p) & \text{if} \quad T_a(p) < T_{rl}(p) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where the user has specified an upper (T_{ru}) and lower bound (T_{rl}) on the required arrival times.

Penalty for Matched Critical Path Constraints

We assign the penalty for a set of paths to be:

$$P_p = \begin{cases} \text{match}(p) - \text{tolerance}(p) & \text{if} \quad \text{match}(p) > \text{tolerance}(p) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

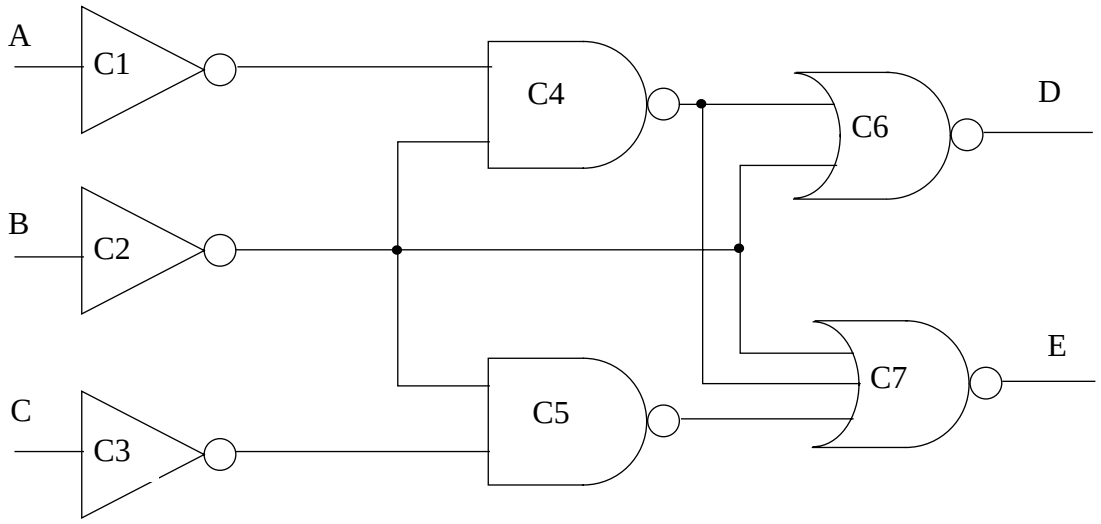
where the match is now defined as difference in arrival times,

$$\text{match}(p) = \left| T_a(p_1) - T_a(p_2) \right| \tag{18}$$

Timing Driven Placement Using Pin Pairs

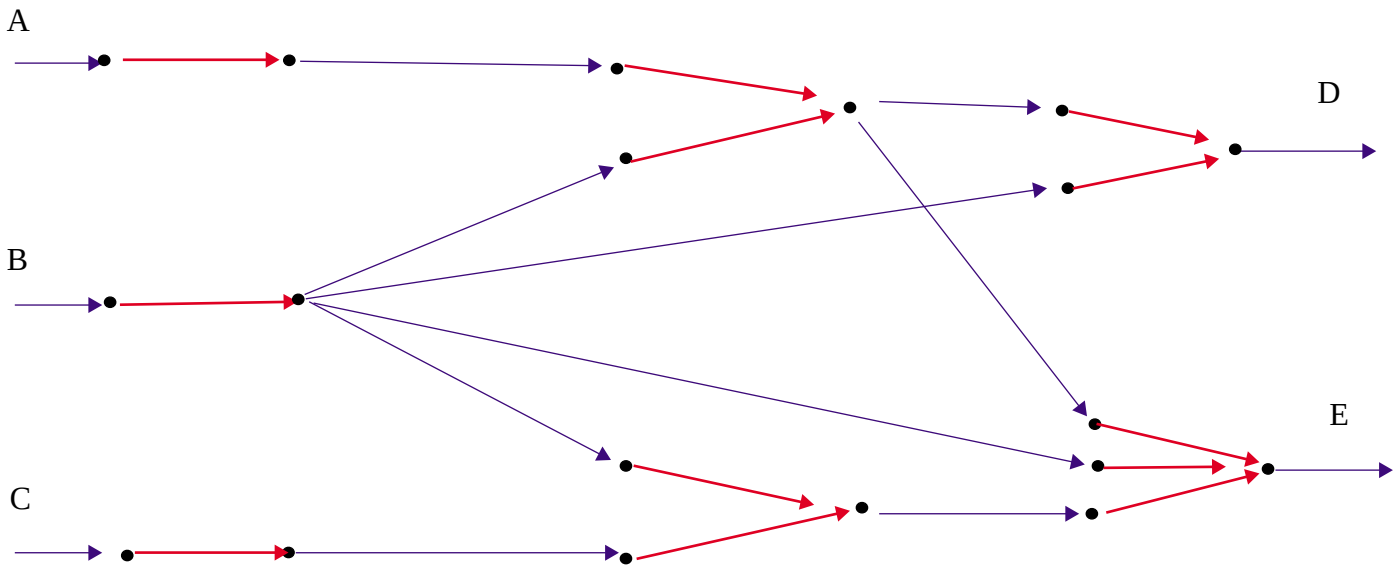
- User only specifies primary input and primary output pin pairs from logic diagram.

No need to enumerate critical paths between pins.



Pin Pair Constraints

- From logic diagram convert to timing graph
- Remove any cycles by breaking feedback paths.
- Use delay modifiers to break any false paths (user specified or from TA).



New simulated annealing algorithm

Algorithm Simulated-Annealing-With-Timing(M)

```
 1   {FP} ← readFalsePaths()          /* read list of false paths from user */
 2   buildTimingGraphs (P, G_n)       /* P is the set of all pin pairs */
 3   do
 4      do
 5         j = generate(i)
 6         if accept(ΔC, T) then
 7            i = j
 8      until cost is in equilibrium
 9      for each d = (i, j) ∈ P do    /* d is a pin pair */
10         if T_lb (i, j) > 0 then    /* a nonzero lower bound exists */
11            {T_a} ← findMShortestPaths (i, j, M)
12         for each edge e ∈ E [V] do /* negate edge weights */
13            w' ← −w
14         {T_a} ← findMShortestPaths (i, j, M)   /* find M longest paths */
15      reduce(T)
16   until cost cannot be reduced any further
```

Pin Pair Constraints

- User parameter M controls the number of paths monitored during simulated annealing.
- Since timing graphs are directed acyclic graphs, we can find the M shortest or M longest paths in $O(Mn\log n)$ time using Dreyfus's algorithm.

Matched Pin Pair Constraints

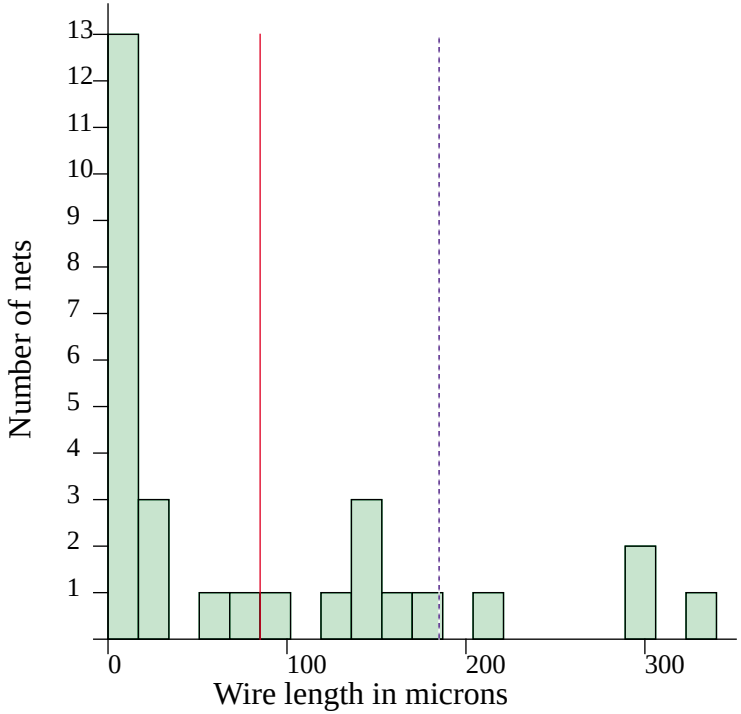
- The user specifies a tolerance for the mismatch in path length. In this case, we assign the penalty for a set of paths to be:

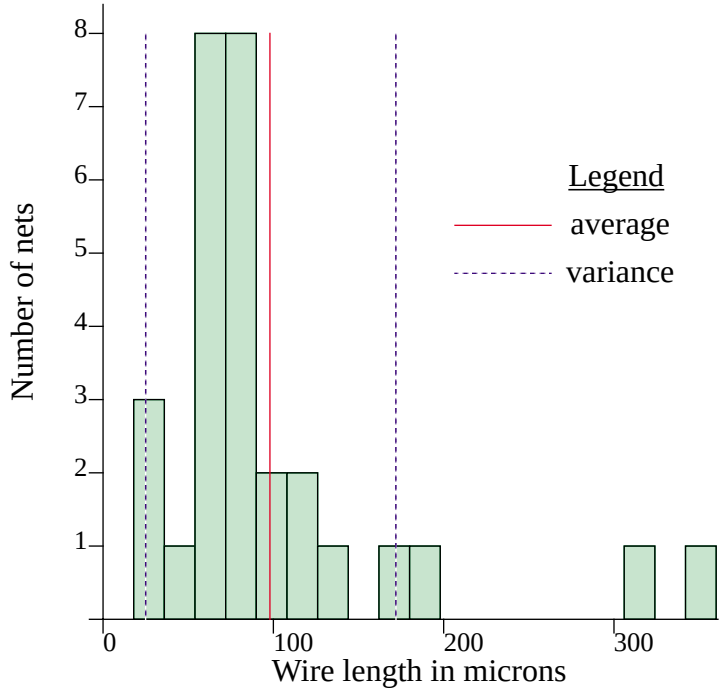
$$_p = \begin{cases} \text{match}(p) - \text{tolerance}(p) & \text{if} \quad \text{match}(p) > \text{tolerance}(p) \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where the match is now defined as difference in arrival times,

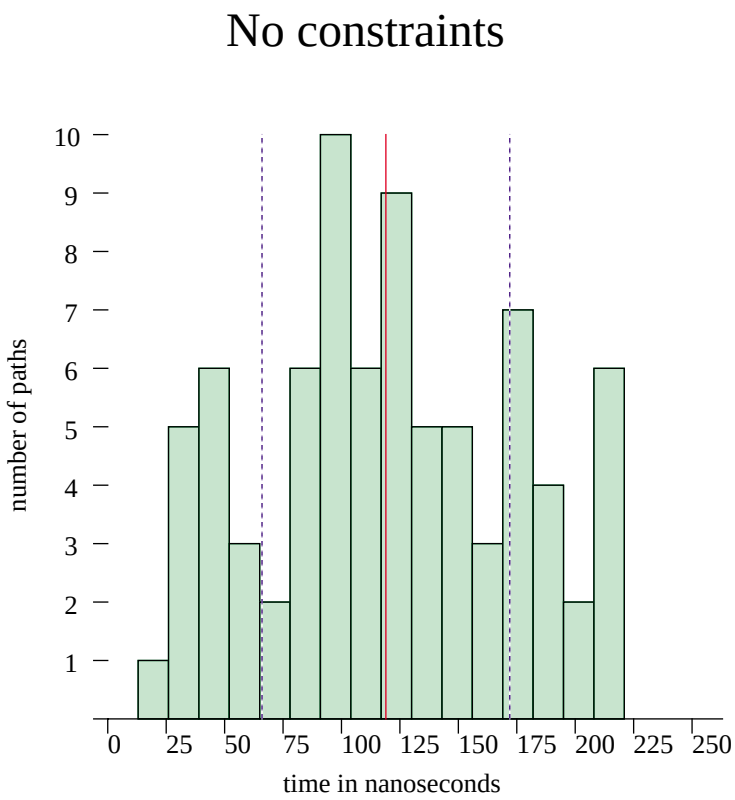
$$\text{match}(p) = \max_{i,j} \left| T_a(p_i) - T_a(p_j) \right| \tag{20}$$

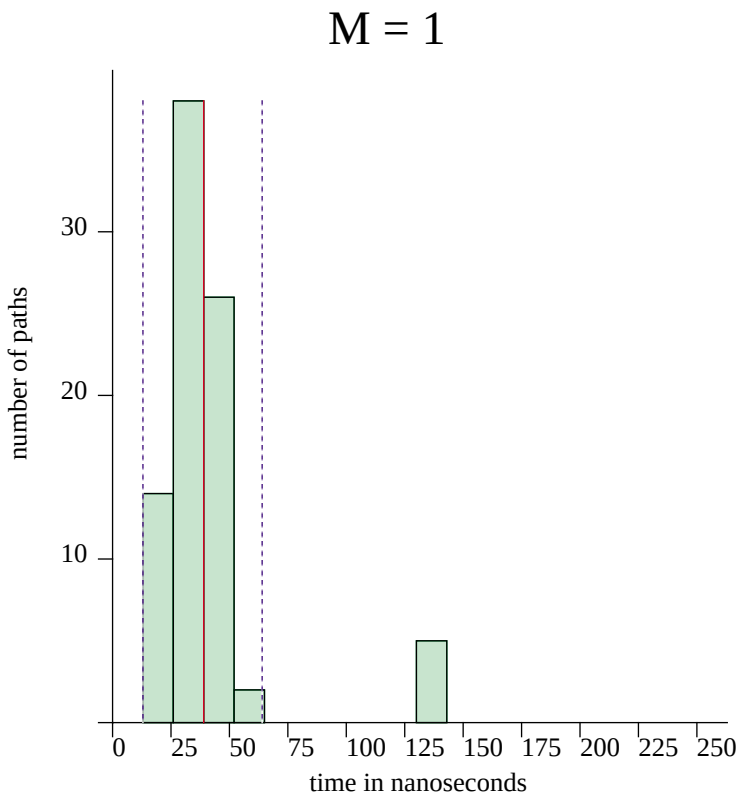
Results: wire length constraints





Results: pin pair constraints





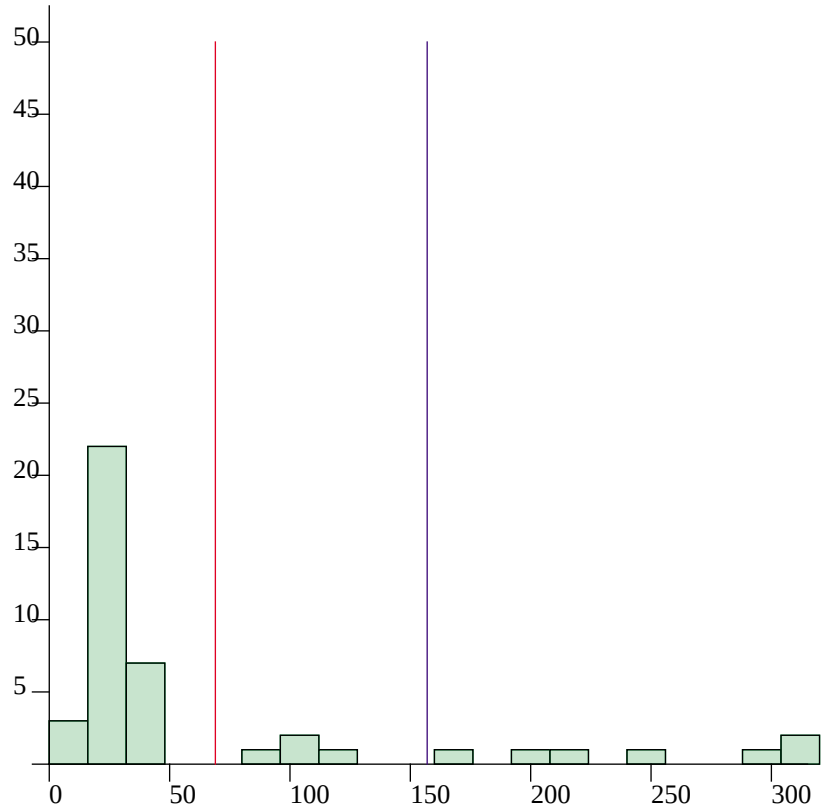
Results : pin pair constraints

run	no constraints	M=1	M=5	M=17
1	198	130	131	129
2	209	134	125	134
3	211	135	119	133
4	214	136	126	124
5	209	137	119	129
average	208	134	124	130

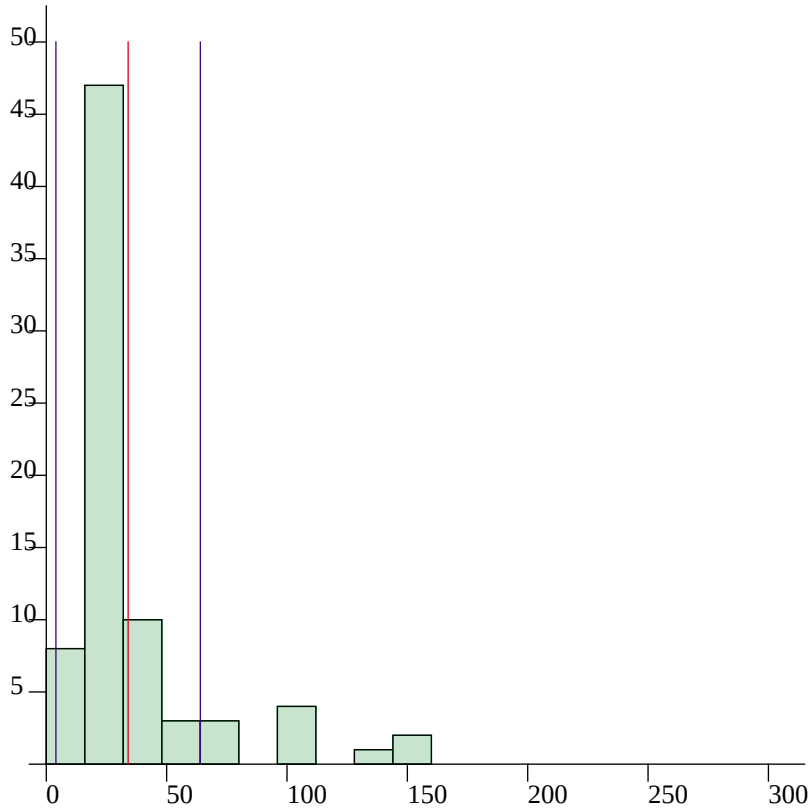
	no constraints	M=1	M=5	M=17
wire length	57276	60691	63073	61209
area (μm^2)	1.56×10^6	1.60×10^6	1.68×10^6	1.58×10^6
run time (secs.)	670	801	966	1201

- The integrated circuit is 34% faster, with an area penalty of only 2.5% and an execution time increase of 16%.

Timing Results : *Struct*



No constraints



M=5

Crosstalk Constraints

- Parasitic capacitance between conductors:

$$C_{AB} = \frac{\varepsilon A}{d} \tag{21}$$

- Lower capacitance
  - Decrease area of overlap.
  - Increase spacing betwen conductors.

Net Classification Scheme

ANAGRAM net classification

NET digital_net1 noisy

NET analog_net1 sensitive

NET ground CLASS neutral

NET digital_net2 CLASS noisy

NET digital_net3 CLASS noisy

General net classification

NET digital_net1 CLASS 1

NET analog_net1 CLASS 2

NET ground CLASS 3

NET digital_net2 CLASS 4

NET digital_net3 CLASS 5

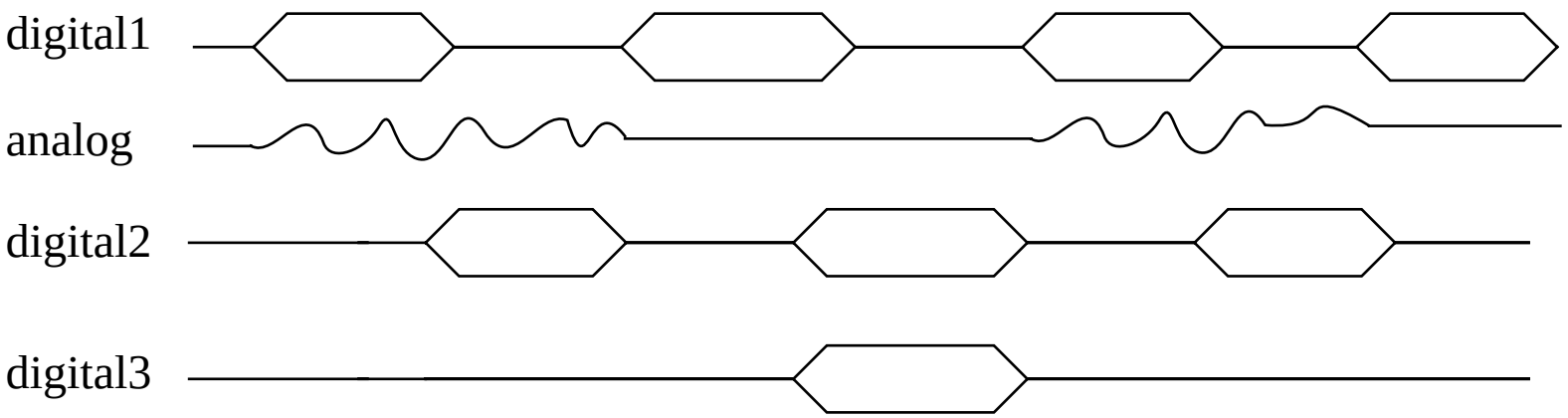
CROSSTALK CLASS 1 CLASS 2 10 10

CLASS 3 SHIELDS 1 FROM 2

CROSSTALK CLASS 2 CLASS 4 10 10

CROSSTALK CLASS 4 CLASS 5 20 20

Justification for the General Net Classification

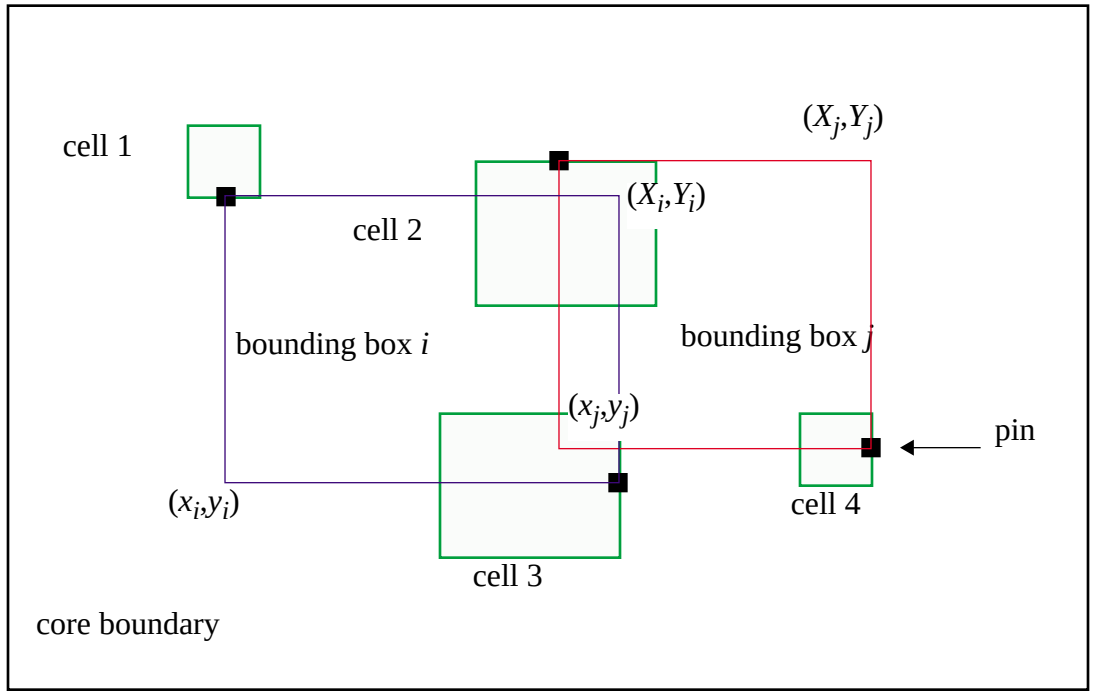


- Each signal has a period of activity.

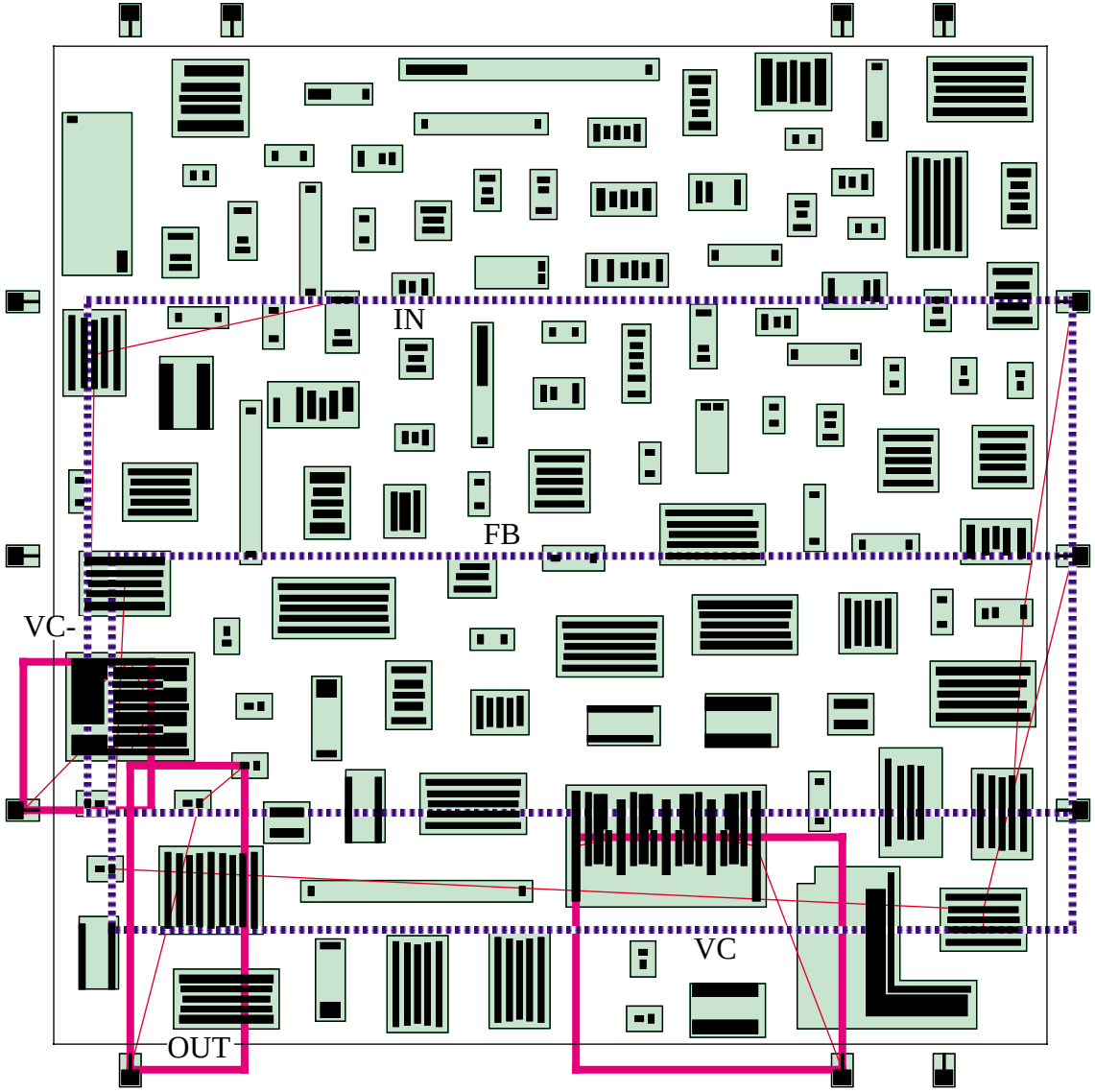
Crosstalk penalty in Simulated Annealing

$$T = \begin{cases} P & \text{if overla} \\ 0 & \text{otherwise} \end{cases} \qquad (22)$$

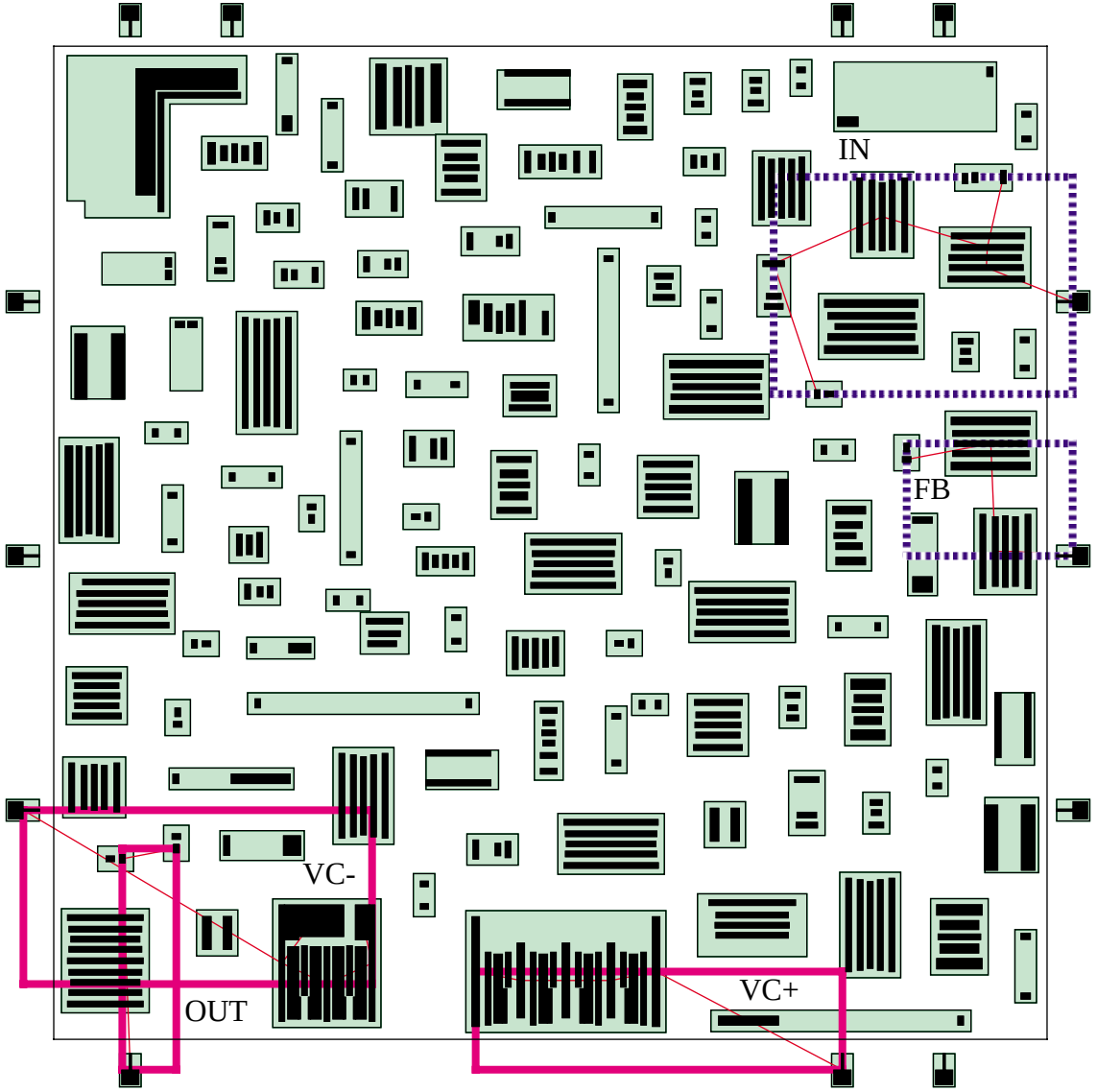
$$P = [\min(X_i, X_j) - \max(x_i, x_j)] + [\min(Y_i, Y_j) - \max(y_i, y_j)] + d_x + d_y \qquad (23)$$



Crosstalk (without constraints)



Crosstalk (with constraints)



New Global Routing Algorithm

- Generalized row-based global router suitable for standard cell, gate-array, sea-of-gates, and FPGAs.
- First row-based global router to *explicitly* minimize chip area.
- Adapts to technologies.
- Uses exact port locations and exact density calculations.
- Handles preplaced feedthrough cells.
- Allows incremental global routing.
- Vertical constraint minimization
- Takes timing into account.

Previous Work

Global Routing

- Maze Routing - Lee [1961].
- Constructive Method - J.T. Lee and M. Marek-Sadowska [1984].
- TimberWolf 3.2 Standard Cell Global Router - Sechen and Sangiovanni-Vincentelli [1986].
- Integer Linear Programming - Raghavan and Thompson [1987].
- Parallel Algorithm - Rose [1988].
- Standard Cell Router - Cong and Preas [1988].
- Field Programmable Gate Arrays - Rose et al. [1990][1991].
- Sea-of-Gates Extension - Lee and Sechen [1991].
- Provably Good Performance-Driven Global Routing - Cong et al. [1992].

Previous Work

Steiner Tree Generation

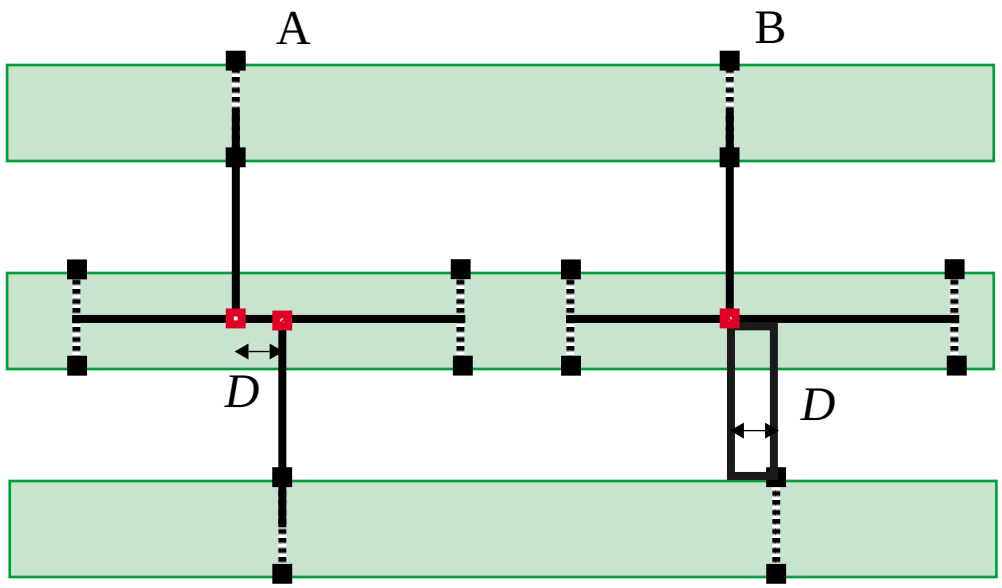
- Fermat [1638]
- Torricelli [1646]
- Hanan [1966]
- Hwang [1976][1979]
- Lee, J. H., Bose, and Hwang [1976]
- Servit [1981]
- Bern [1988]
- Lee, K. and Sechen [1988]
- Richards [1989]
- Ho, Vijayan and Wong [1990]
- Hasan, Vijayan, and Wong [1990]
- Kahng and Robins [1990]
- Chua and Lim [1991]

But the Results are Worse

Circuit	TimberWolfSC /SGGR			TimberWolfSC with iterated Steiner		
	wirelength	feedthrus	tracks	wirelength	feedthrus	tracks
primary1	944360	717.8	163.5	942816	749	163.4
primary2	4251730	4383	346.4	3811600	2982	359.6

- Data is for an average of 8 runs.

Steiner Points Aren't Always Good



A) Desired Steiner tree if Steiner point separation *D* is greater than average feed separation.

B) Otherwise, only one Steiner point should be inserted.

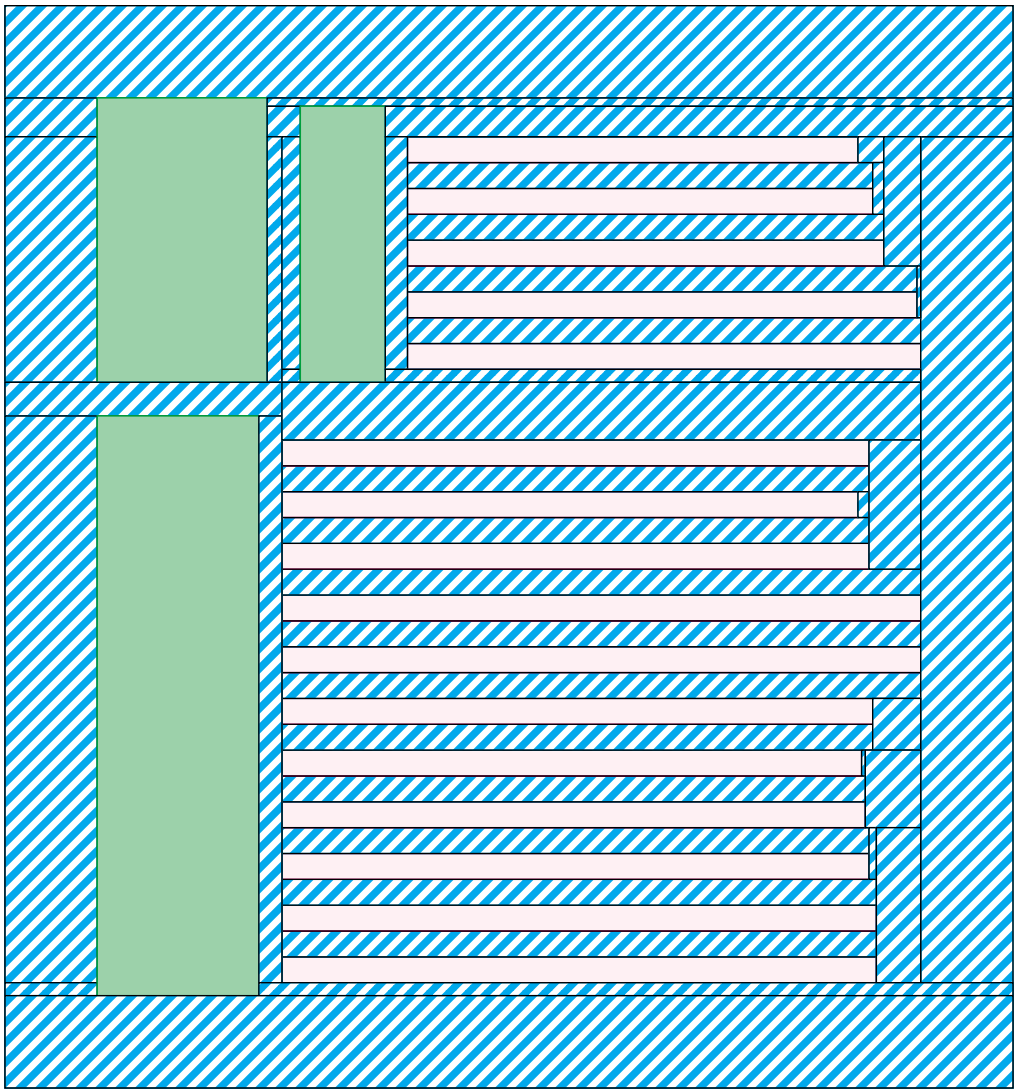
- Feedthrough resources must be taken into account when building Steiner trees.

Overview

Algorithm Global-Router(placement, netlist)

1 Region-Generation()
2 Area-Minimization()
3 Assign-Multi-Row-Feedthroughs()
4 Assign-Single-Row-Feedthroughs()
5 Remove-Cell-Overlap()
6 Cell-Swap-Optimization()
7 Switchable-Segment-Optimization()
8 Maze-Route()
9 Vertical-Constraint-Minimization()
10 Route-Verification()

Region Generation

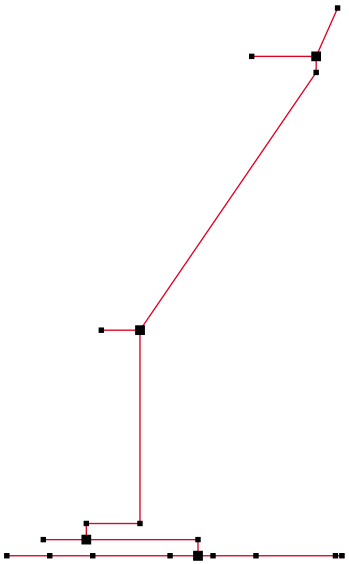


Area Minimization

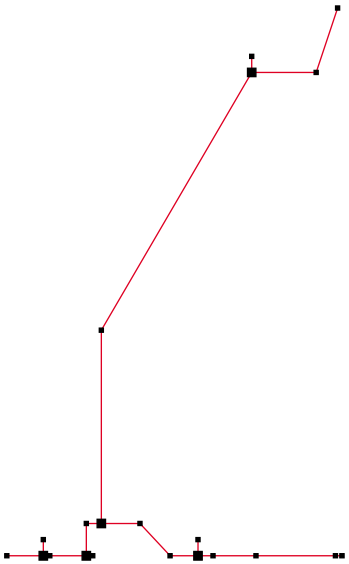
Algorithm Area-Minimization()

1 **for** $i \leftarrow 1$ **to** *numnets*
2 **do** T_i = Build-Steiner-Tree(*i*, MINIMUM_WIRELENGTH)
3 **compute** $H = \sum_{r=1}^{numregions} trackpitch \cdot tracks_r + \sum_{b=1}^{numrows} height_b$
4 **compute** $W = \max_{b \in \{1, numrows\}} (length_b + feedwidth \cdot feeds_b)$
5 **compute** P_T
6 $oldcost \leftarrow W \cdot H$
7 **do**
8 $n \leftarrow Random(1, numnets)$
9 pick segment *s* of net *n*
10 **if** *s* crosses maximum density **then**
11 *cost* ← Flip-Segment()
12 **else**
13 **if** feed limited **then**
14 T_n ← Build-Steiner-Tree(*n*, MINIMUM_FEEDS)
15 **else if** density limited **then**
16 T_n ← Build-Steiner-Tree(*n*, MINIMUM_DENSITY)
17 **else if** Instance-Versions exists for net **then**
18 $\{T_i\}$ ← Swap-Instance-Versions(*n*)
19 cost ← Calculate-New-Cost()
20 **if** $cost < oldcost$ **then** Accept-Move()
21 **while** *cost* improves

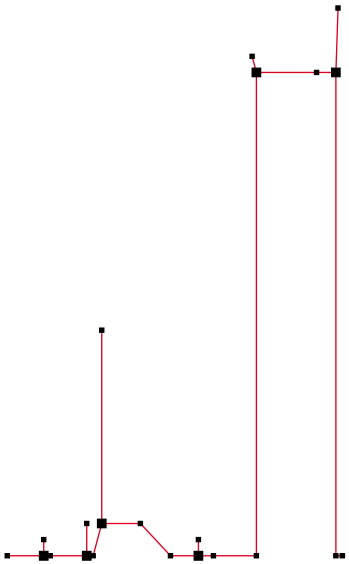
Steiner Tree Construction



Minimize Feeds

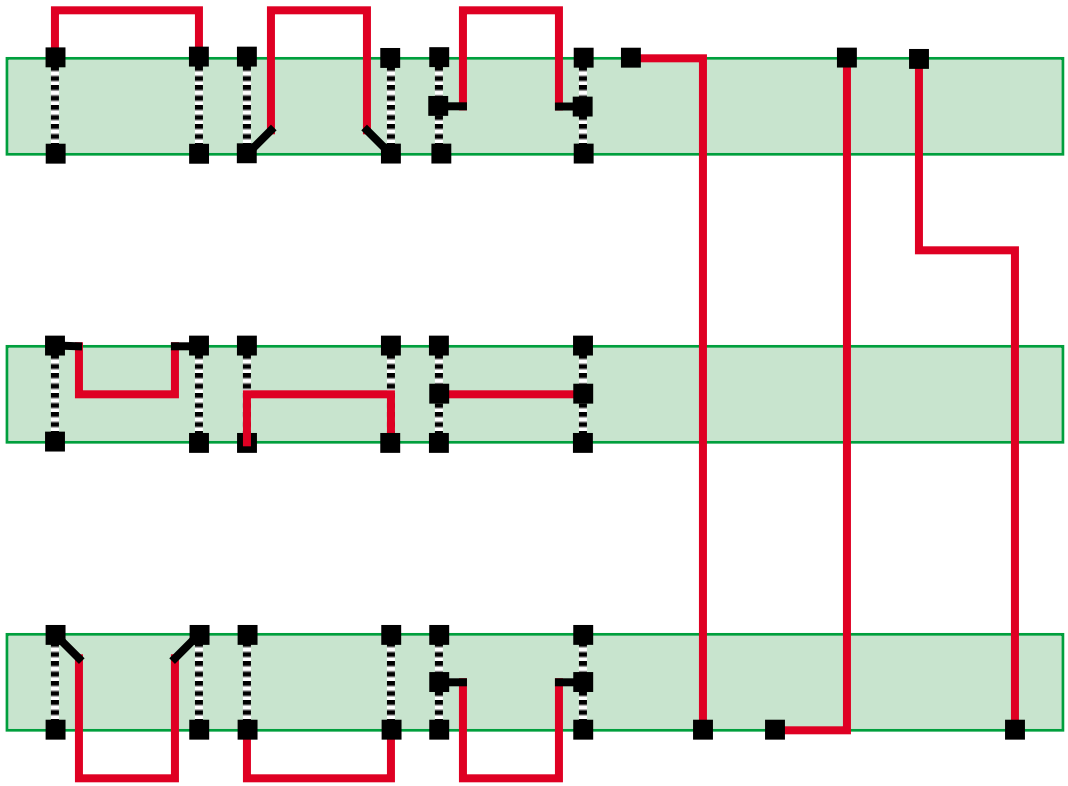


Minimize Wirelength

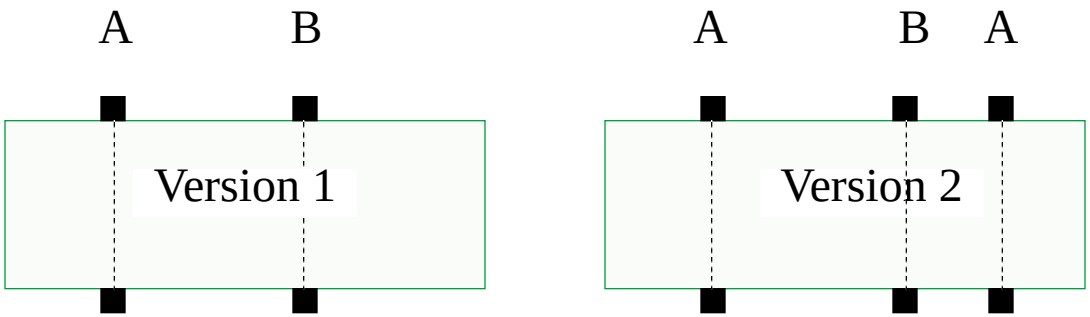


Minimize Density

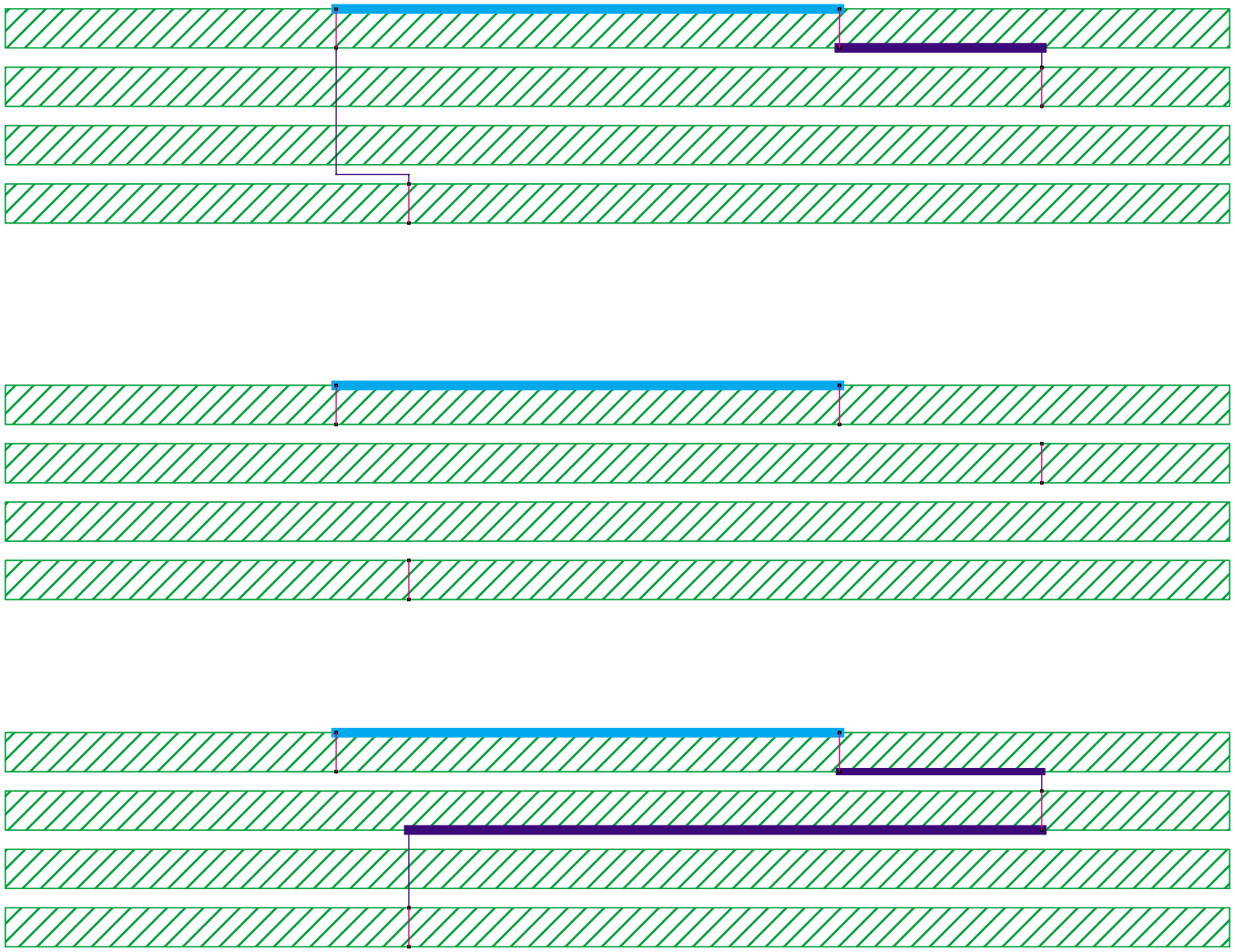
Switchable Segment Optimization



Instance Versions



Steiner Tree Reconstruction



Feedthrough Assignment

First Phase

- Assign feedthroughs over macro cells and/or FPGA freeways.

Second Phase

- Assign feedthroughs over rows if needed.

Optimal with respect to wirelength and congestion.

Phase I Assignment

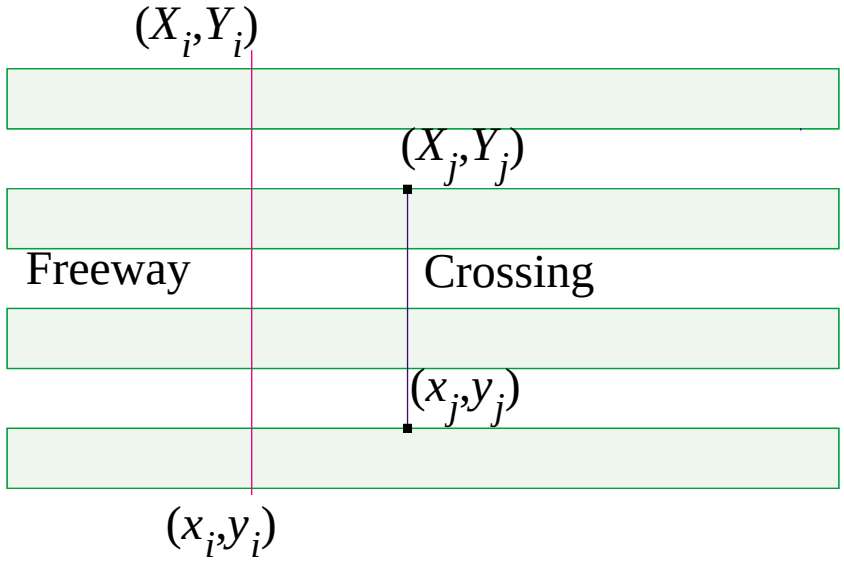
Cost function

$C_{ij} = |X_i - X_j| + (Y_i - y_i) - [\min(Y_i, Y_j) - \max(y_i,y_j)] + P_1$

where $F_i(x) = [x_i,X_i]$, $F_i(y) = [y_i,Y_i]$

$C_j(x) = [x_j,X_j]$, $C_j(y) = [y_j,Y_j]$

$P_1 = K \cdot \Delta density$



Multiple Row Freeway Assignment

Algorithm Feedthrough Assignment Phase I()

```
 1   work_to_do ← TRUE
 2   do
 3      {Region_i} ← Find-Available-Macrofeed()
 4      num_free ← Find-Number-Macrofeeds({Region_i})
 5      num_cross ← Find-MultiRow-Crossings({Region_i})
 6      if num_cross = 0 then break
 7      if num_cross > 0 and num_free = 0 then error break
 8      if num_cross > num_free then
 9           Relax-Crossings({Region_i})
10      Linear-Assignment(C)
11      Update-Steiner-Trees()
12   while work_to_do
```

Phase II Assignment

Cost function

$C_{ij} = |X_i - X_j| + P_1 + P_2 + P_3$

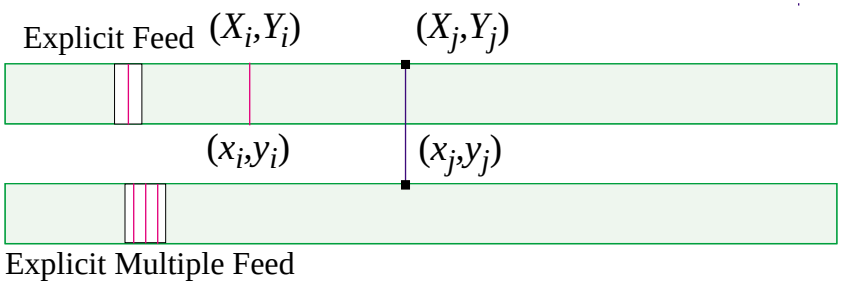
where

$$P_1 = \begin{cases} rowlength & \text{if } explicit\ fe \\ 0 & otherwise \end{cases}$$

$$P_2 = \begin{cases} K & \text{if } unused\ multife \\ 0 & otherwise \end{cases}$$

$$P_3 = K \cdot \Delta density$$

$$rowlength \ll K$$

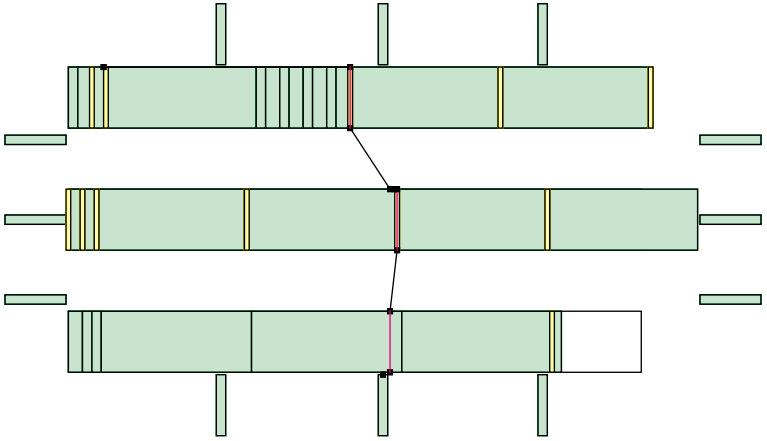


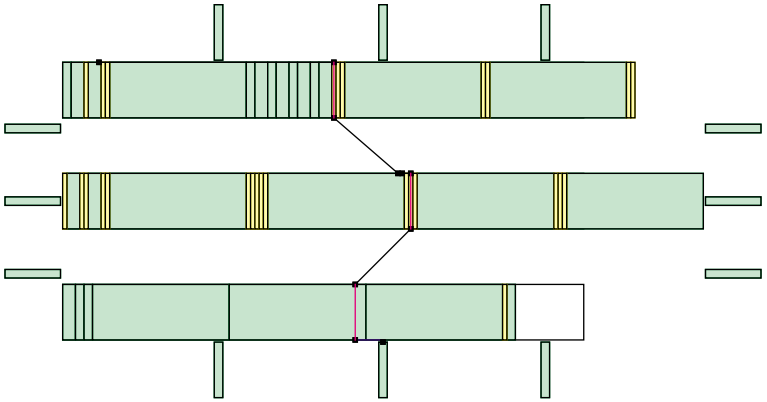
Feedthrough Assignment

Algorithm Feedthrough Assignment Phase II()

```
1   for r ← 1 to numrows do
2       avail ← Find-Available-Feeds(r)
3       need ← Find-Net-Crossings(r)
4       if avail < need then
5           if fixed-width then
6               Relax-Steiner-Trees(r, MINIMUM_FEEDS)
7           Add-Extra-Feeds-Between-Cells()
8       Linear-Assignment(C)
9       Update-Steiner-Trees()
```

Cell Overlap Removal





Cell Swap Optimization

Pairwise interchange of neighboring cells.

Orientation optimization.

$$C = \sum_{n=1}^{nets} \left(S_{x_n} + S_{y_n} \right) + P_T$$

where

S_{x_n}, S_{y_n} *steiner wirelength*

P_T *timing penalty*

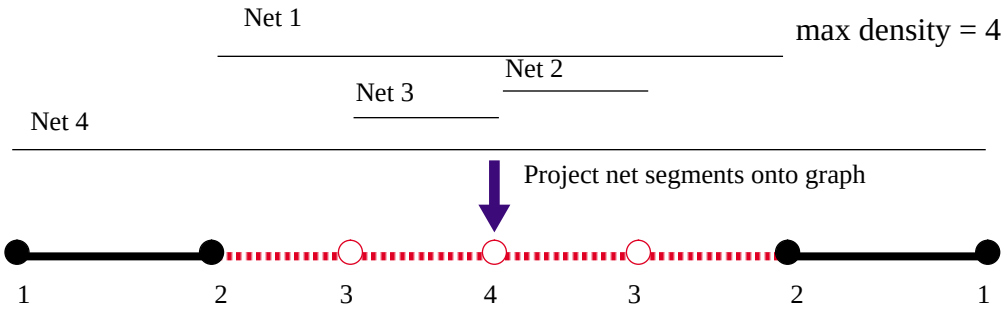
Has major impact for designs that require many explicit feedthrus.

Switchable Segment Optimization

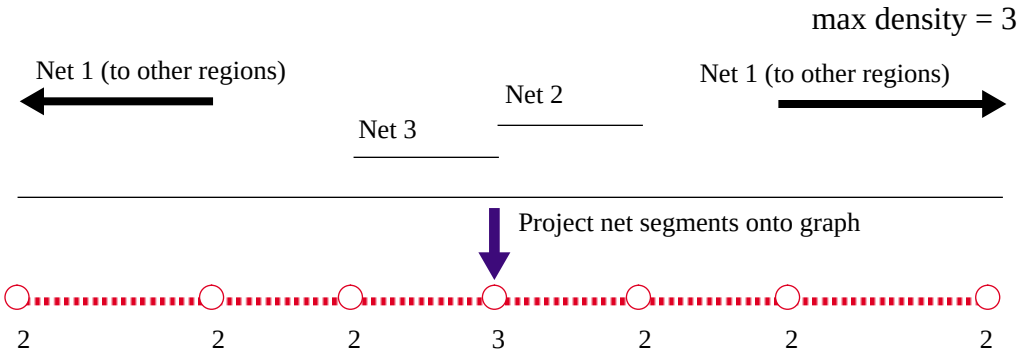
- Run area minimization algorithm again.
- Since feed positions are known $\Rightarrow$ chip width is fixed.

Optimize track density.

Maze Routing



Now if we reroute net 1 using other regions:



Legend:

Noncritical Edge -

Critical Edge -

Noncritical Node

Critical Node

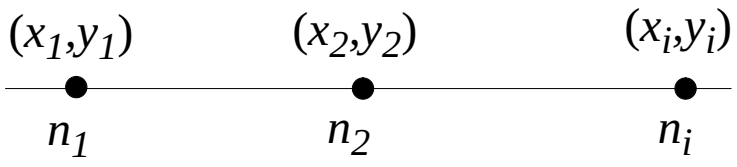
Maze Routing Critical Edges

We define

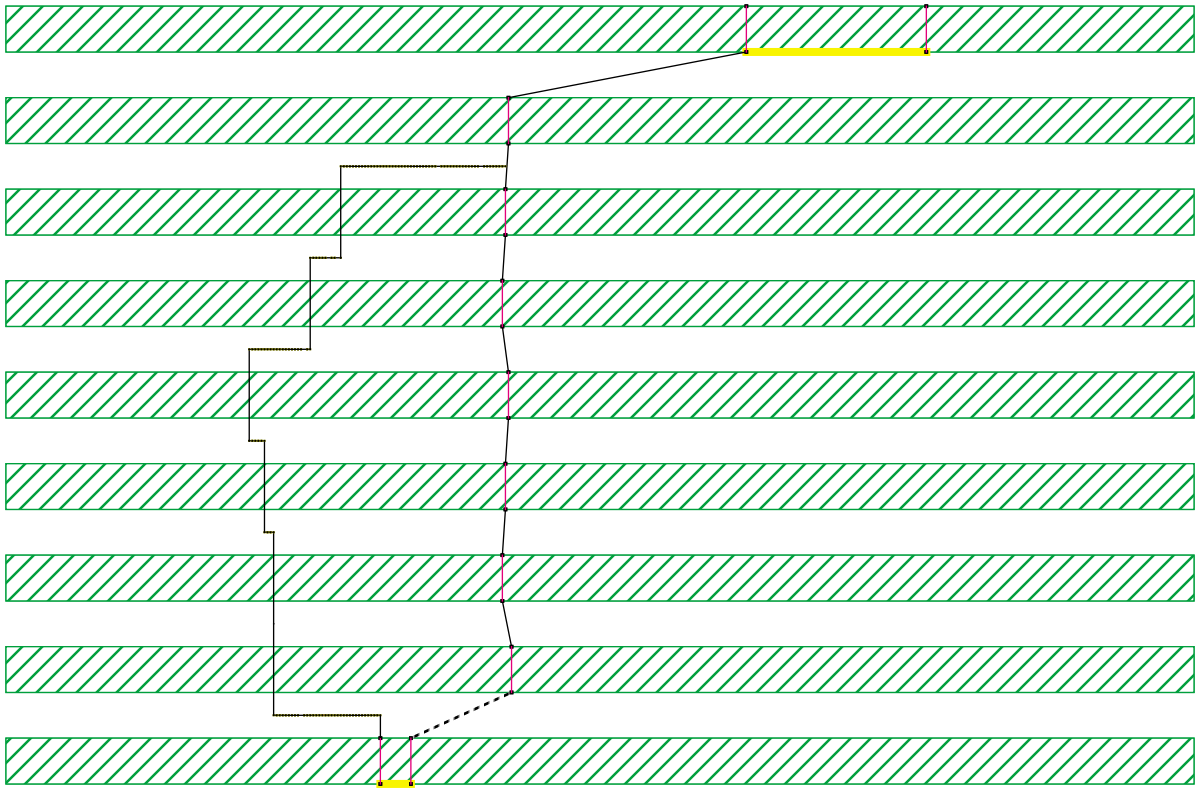
$$critical \equiv \begin{cases} 0 \text{ if } density(node) < maxdensity(region) - 1 \\ \\ 1 \qquad\qquad\qquad\qquad otherwise \end{cases}$$

Dynamically, we update the edge cost weights

$$C = \begin{cases} \infty \quad if \quad critical(n_1) \quad \text{or} \quad critical(n \\ |x_1 - x_2| + |y_1 - y_2| \quad \text{otherwise} \end{cases}$$



Maze Route Example



Vertical Constraint Minimization

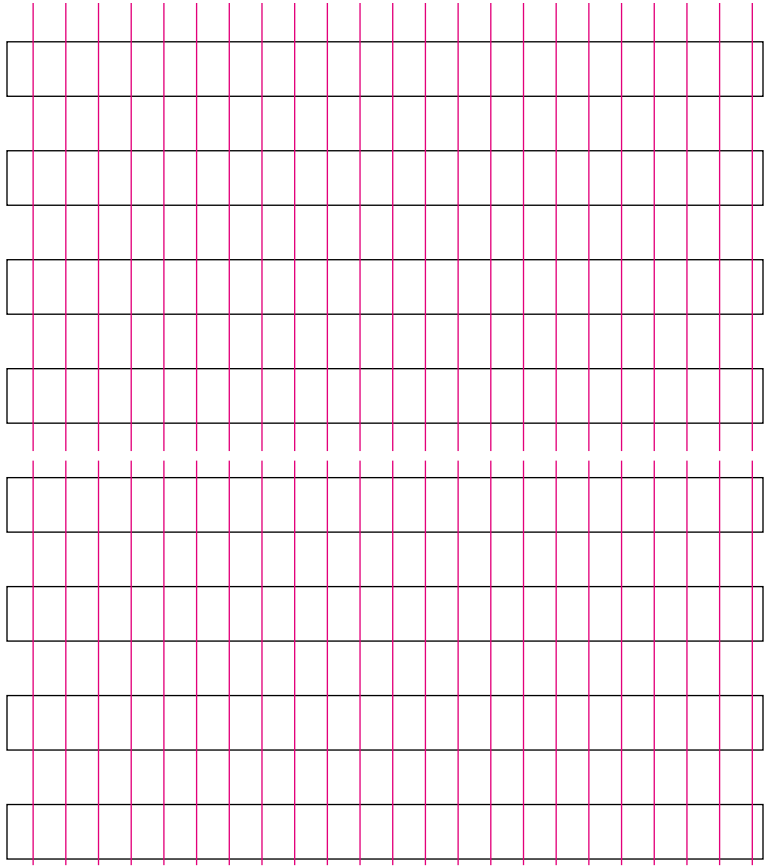
```
lgorithm Vertical-Constraint-Minimization()
1   C ← { }                                          /* C is the set of cycles */
2   for r ← 1 to numregions do
3       G_v ← Build-Vertical-Constraint-Graph (r)
4       C ← C ∪ FindCycles (G_v, r)                  /* find cycles in the vertical constraint graph */
5   oldcost ← Initialize-Cost()                      /* calculate initial track density */
6   while C ≠ { } and cost improves do
7       n ← Random (1, numnets)
8       pick segment s of net n
9       if s ∈ C then                                /* check to see if segment s occurs in any cycle */
0           cost← Flip-Segment()
1           broken ← Check-Cycle (C, s)              /* does the flip break the cycle */
2           if cost ≤ oldcost and broken = TRUE then
3               Accept-Move()
4               oldcost ← cost
5               C = C – {C_s}
```

Route Verification

- Depth-first search of Steiner tree checks to see if all pins are connected.
- Depth-first can detect if cycles exist.

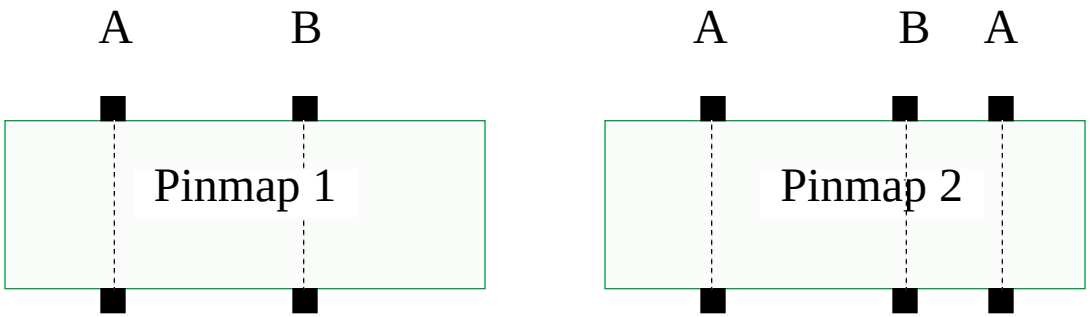
Row-based FPGA Extensions

Freeway map

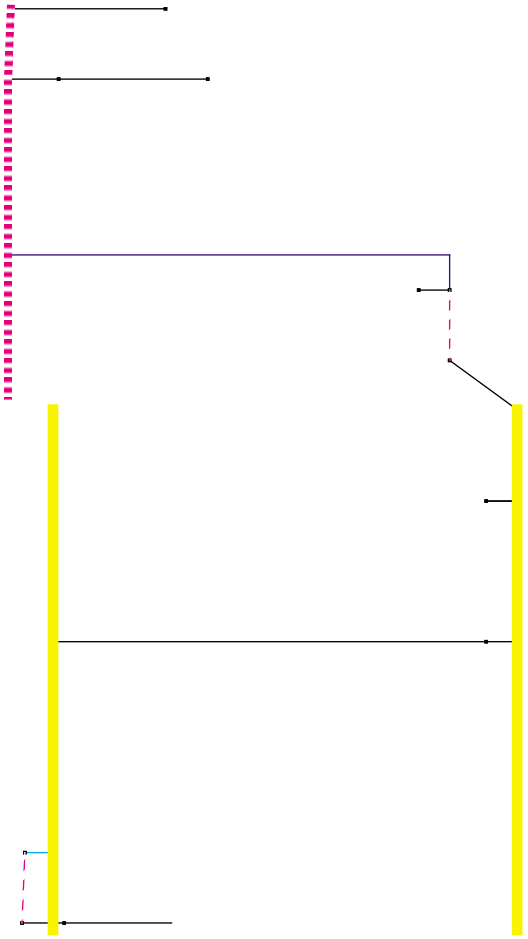


Row-based FPGA Extensions

- FPGA pin-maps are handles using instance version moves in area minimization.



Row-based FPGA Extensions



Algorithmic Complexity

Step	Time Complexity	Space Complexity
Region-Generation	$O(R^2)$	$O(R)$
Area-Minimization	$O(NS)$	$O(P)$
Assignment I	$O(F^3)$	$O(F)$
Assignment II	$O(F^3)$	$O(F)$
Remove-Cell-Overlap	$O(ClogC)$	$O(C)$
Cell-Swap-Optimization	$O(C)$	$O(C)$
Switchable-Segment-Opt	$O(N\mathrm{S})$	$O(P)$
Maze-Route	$O(P[logV + E])$	$O(E+V)$
Vertical-Constraint-Min	$O(N\mathrm{S})$	$O(P)$
Route-Verification	$O(V+E)$	$O(P)$

Legend: C = number of cells, E = number of edges in maze graph, F = number of feedthrus in a single row/region, N = number of nets, P = max. number of pins of a single net, R = number of regions, S = number of net segments, V = number of vertices in maze graph.

Results

MCNC Benchmark Results

Global Router Comparison using the same placement.

Circuit	TimberWolfSC version 6.0			TimberWolfSC version 7.0		
	width	tracks	area μm^2	width	tracks	area μm^2
sp1	5210	163	2.18×10^7	5200	160	2.16×10^7
sp2	11730	401	8.76×10^7	11600	374	8.34×10^7
guts	8344	1817	5.76×10^7	8300	1734	5.59×10^7

Implicit feeds were removed from the circuits.

Results

- Results of MCNC benchmarks. Track count for implicit feeds case.

Circuit	TimberWolfSC 6.0			SGGR			TimberWolfSC 7.0		
	avg	min	max	avg	min	max	avg	min	max
small	60	51	65	55	53	59	49	47	51
primary1	150	142	167	141	140	141	141	140	143
primary2	386	368	407	346	342	352	340	338	344
industry3	1631	1576	1756	1485	1477	1490			

Results

- Results of MCNC benchmarks. Wire length comparison for implicit feeds case.

Circuit	SGGR			TimberWolfSC 7.0		
	avg	min	max	avg	min	max
small	109719	109082	111474	95501	94280	96810
primary1	801217	794165	811950	736283	735290	737600
primary2	4257644	4234530	4276180	4007742	3987810	4017180

Summary - What's New

- New simulated annealing algorithm which is based on a theoretically derived annealing schedule.
- New method for statistical wiring estimation.
- New placement refinement method was developed for rectilinear cells which spaces the cells at density avoiding the need for post-routing compaction.
- A new algorithm which uses previously generated constraints to maintain the topology during placement refinement.
- A new detailed routing method has been developed which avoids the classically difficult problem of defining channels for detailed routing, and in addition, avoids the equally difficult problem of defining a routing order for the defined channels.
- A new fully automatic placement and routing system has been developed for mixed macro/standard cell designs.
- A new general net classification scheme for eliminating crosstalk between signals.

What's New (Continued)

- Six new algorithms for controlling time delay in an integrated circuit.
  - A novel pin pair algorithm controls the delay without the need for user path specification.

- A new generalized row-based global router suitable for standard cell, gate-array, sea-of-gates, and FPGAs.
  - It is the first row based global router to explicitly minimize chip area.
  - This global router automatically adapts to technologies.
  - Optimal feedthrough placement is accomplished using linear assignment.
  - Throughout the algorithm, timing constraints are taken into account.
  - A unique vertical constraint minimization step eases the task of LEA channel routers.