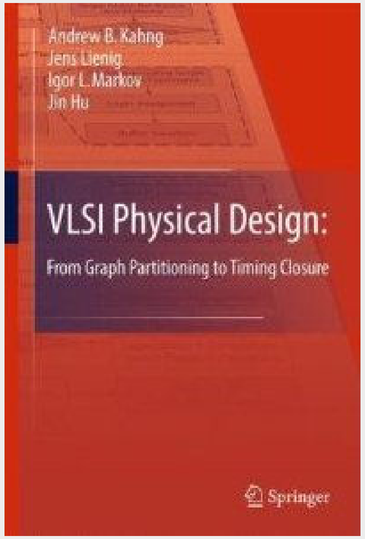
VLSI Physical Design: From Graph Partitioning to Timing Closure

Chapter 7 – Specialized Routing

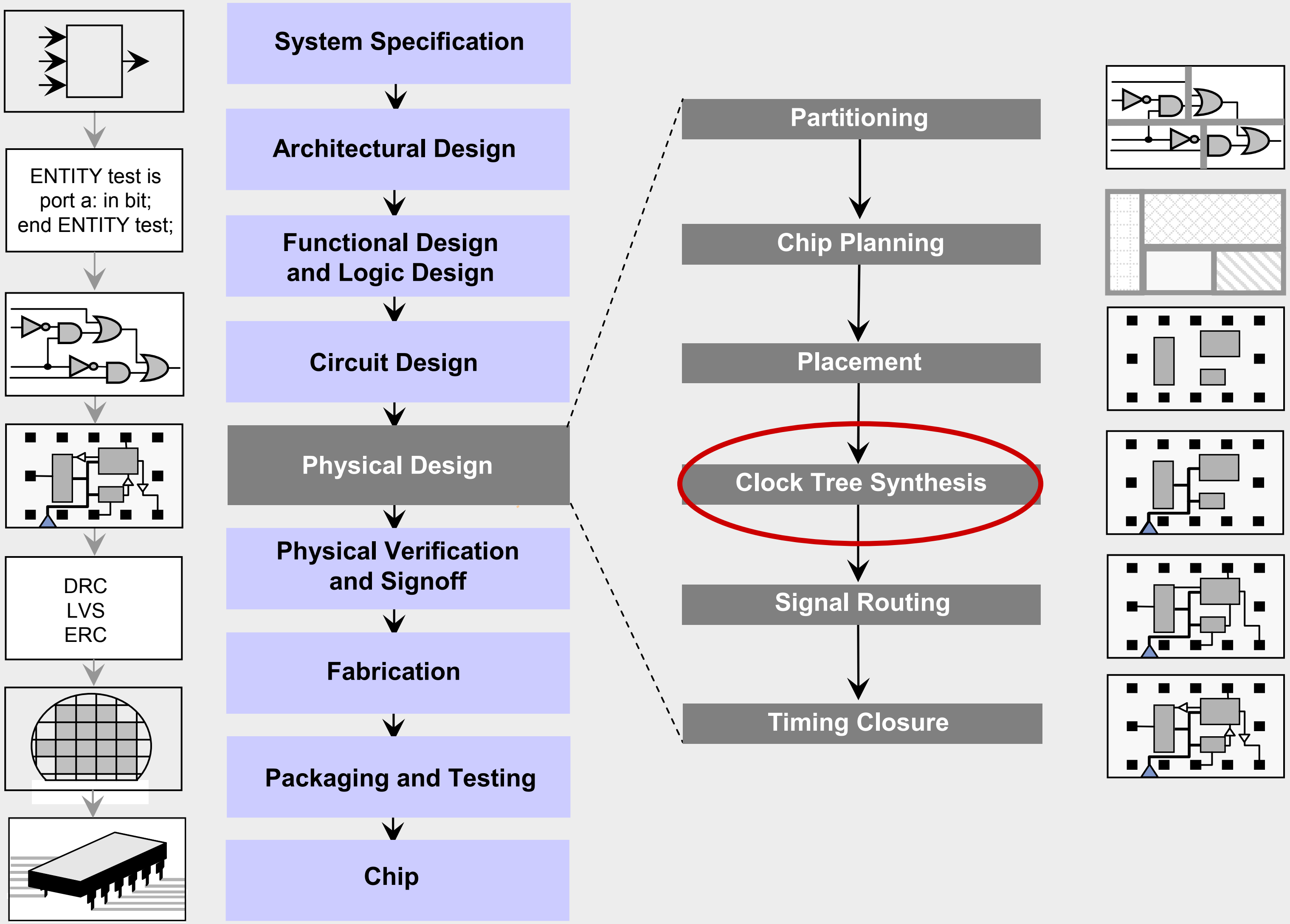


Original Authors:

Andrew B. Kahng, Jens Lienig, Igor L. Markov, Jin Hu

Chapter 7 – Specialized Routing

7 Specialized Routing



System Specification

Architectural Design

Functional Design and Logic Design

Circuit Design

Physical Design

Physical Verification and Signoff

Fabrication

Packaging and Testing

Chip

Partitioning

Chip Planning

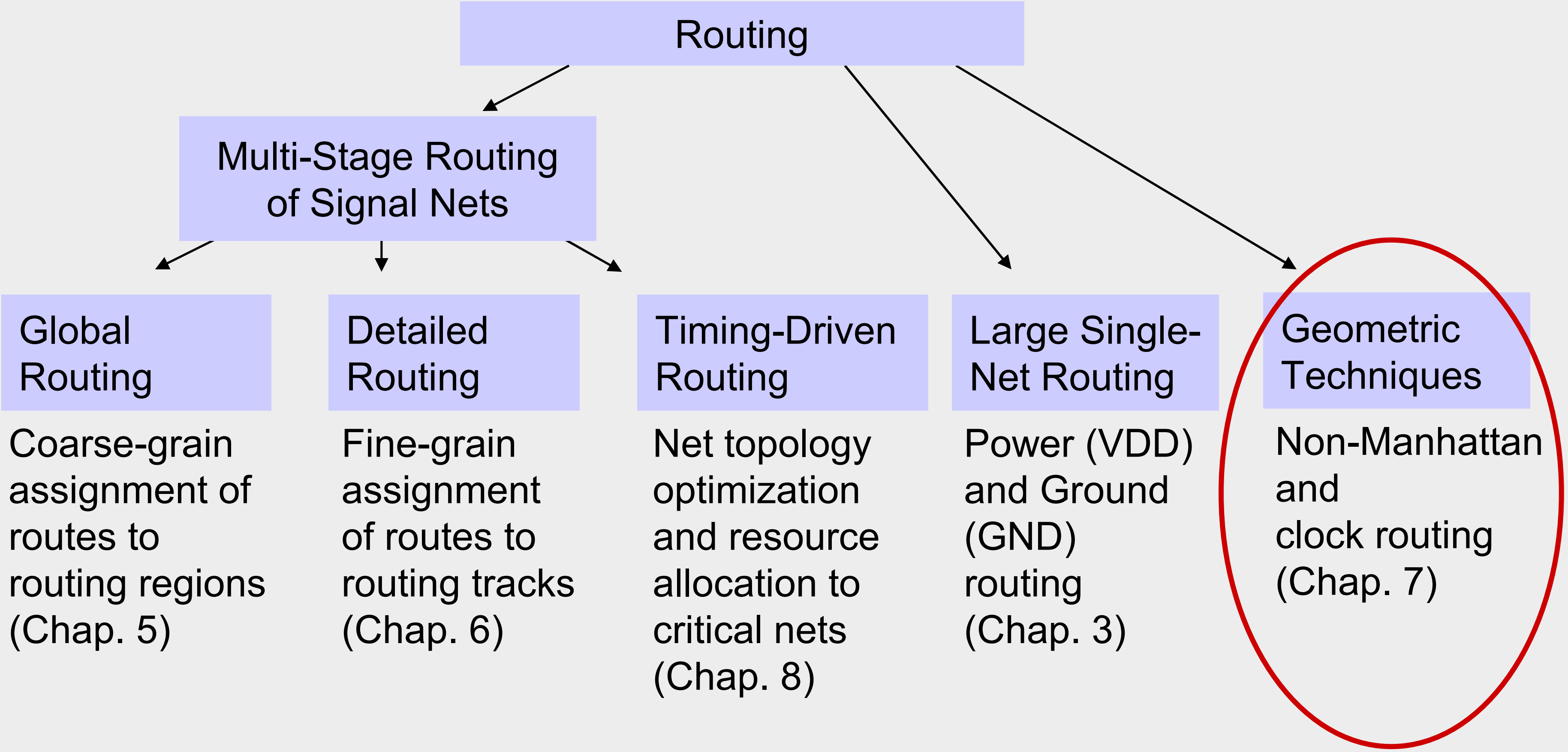
Placement

Clock Tree Synthesis

Signal Routing

Timing Closure

7 Specialized Routing



Routing

Multi-Stage Routing of Signal Nets

Global Routing

Coarse-grain assignment of routes to routing regions (Chap. 5)

Detailed Routing

Fine-grain assignment of routes to routing tracks (Chap. 6)

Timing-Driven Routing

Net topology optimization and resource allocation to critical nets (Chap. 8)

Large Single-Net Routing

Power (VDD) and Ground (GND) routing (Chap. 3)

Geometric Techniques

Non-Manhattan and clock routing (Chap. 7)

7 Specialized Routing

- Area routing directly constructs metal routes for signal connections (no global and detailed routing, Secs. 7.1-7.2)
- Non-Manhattan routing is presented in Sec. 7.3
- Clock signals and other nets that require special treatment are discussed in Secs. 7.4-7.5

7.1 Introduction to Area Routing

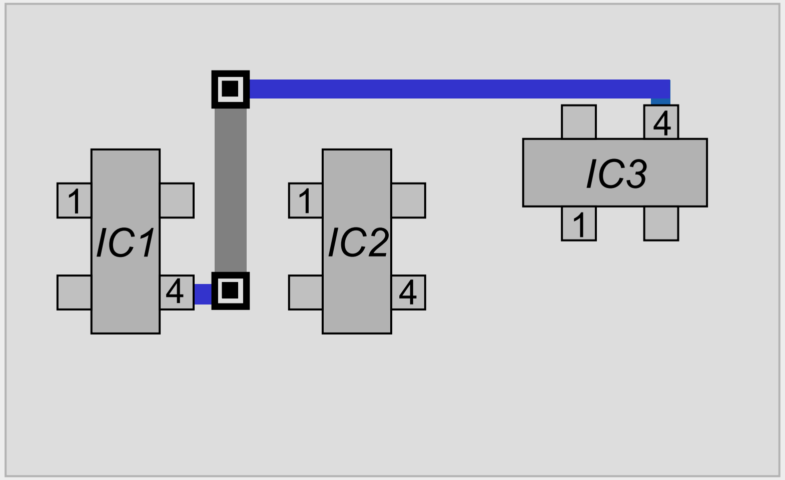
- The goal of area routing is to route all nets in the design
  - without global routing
  - within the given layout space
  - while meeting all geometric and electrical design rules

- Area routing performs the following optimizations
  - minimizing the total routed length and number of vias of all nets
  - minimizing the total area of wiring and the number of routing layers
  - minimizing the circuit delay and ensuring an even wire density
  - avoiding harmful capacitive coupling between neighboring routes

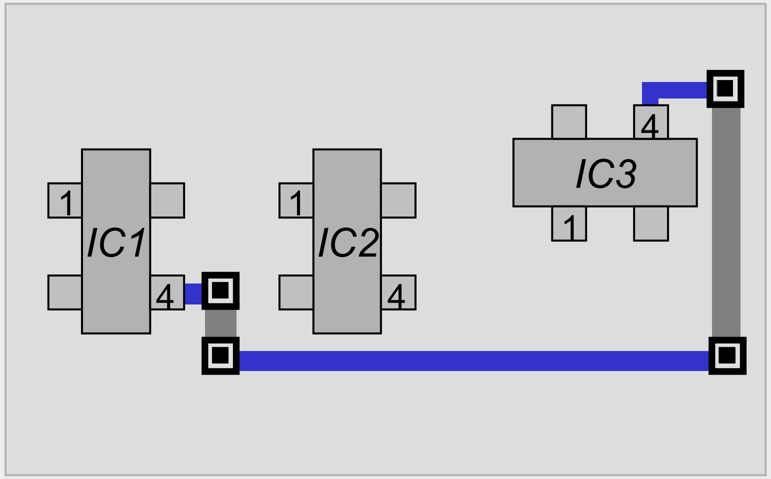
- Subject to
  - technology constraints (number of routing layers, minimal wire width, etc.)
  - electrical constraints (signal integrity, coupling, etc.)
  - geometry constraints (preferred routing directions, wire pitch, etc.)

7.1 Introduction to Area Routing

Minimal wirelength:



Alternative routing path:



▬ Metal1

▮ Metal2

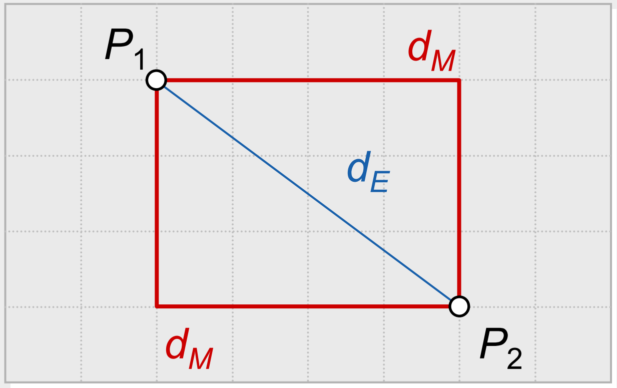
▣ Via

7.1 Introduction to Area Routing

Distance metric between two points $P_1\,(x_1,y_1)$ and $P_2\,(x_2,y_2)$

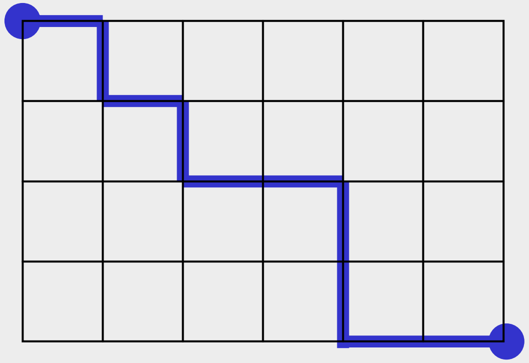
Euclidean distance $$d_E(P_1,P_2)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}=\sqrt{(\Delta x)^2+(\Delta y)^2}$$

Manhattan distance $$d_M(P_1,P_2)=|x_2-x_1|+|y_2-y_1|=|\Delta x|+|\Delta y|$$



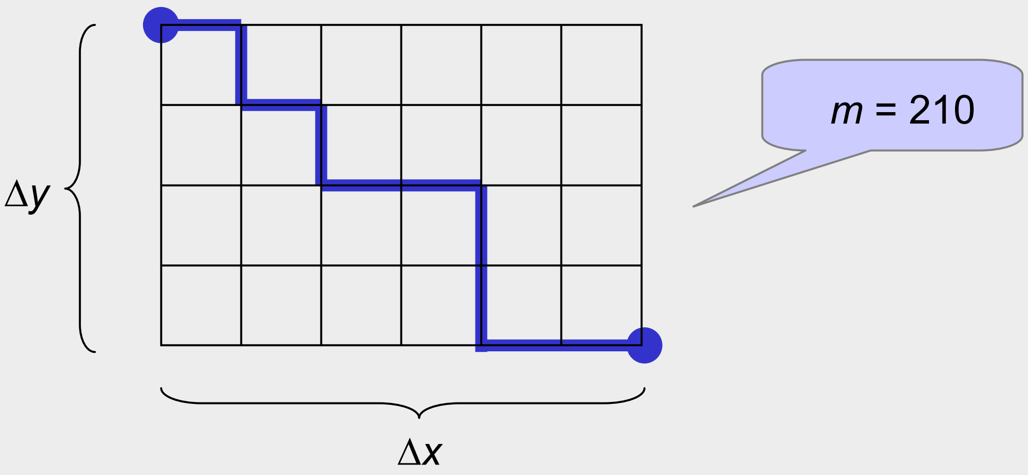
7.1 Introduction to Area Routing

- Multiple Manhattan shortest paths between two points



7.1 Introduction to Area Routing

- Multiple Manhattan shortest paths between two points

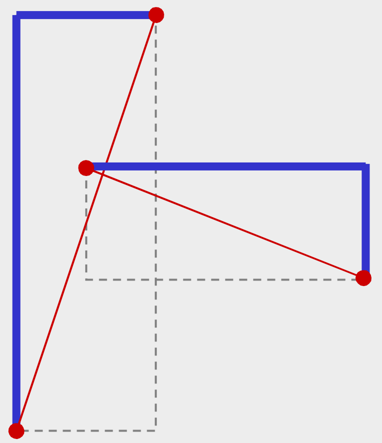


With no obstacles, the number of Manhattan shortest paths in an $\Delta x \times \Delta y$ region is

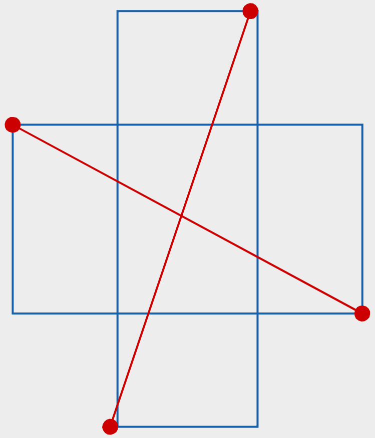
$$m = \binom{\Delta x + \Delta y}{\Delta x} = \binom{\Delta x + \Delta y}{\Delta y} = \frac{(\Delta x + \Delta y)!}{\Delta x! \Delta y!}$$

7.1 Introduction to Area Routing

- Two pairs of points may admit non-intersecting Manhattan shortest paths, while their Euclidean shortest paths intersect (but not vice versa).



- If all pairs of Manhattan shortest paths between two pairs of points intersect, then so do Euclidean shortest paths.



7.1 Introduction to Area Routing

- The Manhattan distance d_M is (slightly) larger than the Euclidean distance d_E:

$$\frac{d_\mathrm{M}}{d_\mathrm{E}} = \begin{cases} 1.41 & \text{worst case: a square where } \Delta x = \Delta y \\ 1.27 & \text{on average, without obstacles} \\ 1.15 & \text{on average, with obstacles} \end{cases}$$

7.2 Net Ordering in Area Routing

7.1 Introduction to Area Routing

→ 7.2 Net Ordering in Area Routing

7.3 Non-Manhattan Routing

7.3.1 Octilinear Steiner Trees

7.3.2 Octilinear Maze Search

7.4 Basic Concepts in Clock Networks

7.4.1 Terminology

7.4.2 Problem Formulations for Clock-Tree Routing

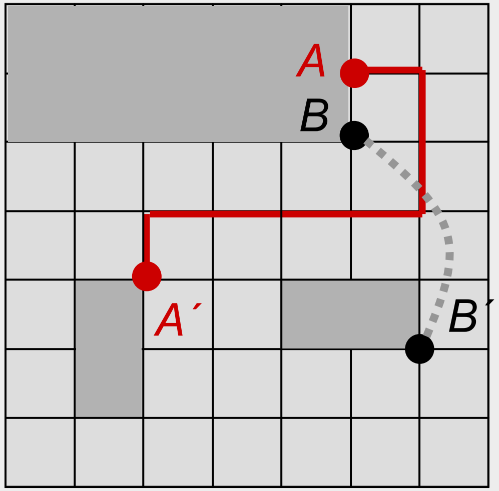
7.5 Modern Clock Tree Synthesis

7.5.1 Constructing Trees with Zero Global Skew

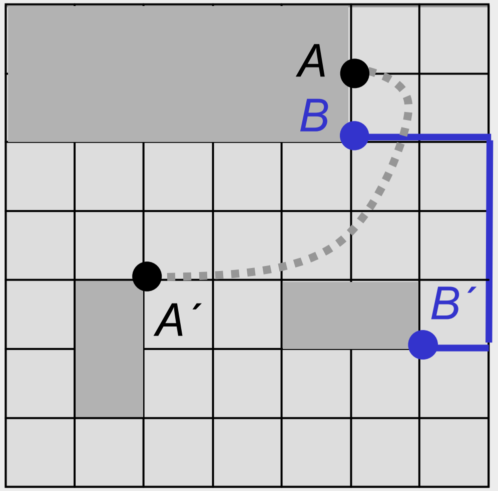
7.5.2 Clock Tree Buffering in the Presence of Variation

7.2 Net Ordering in Area Routing

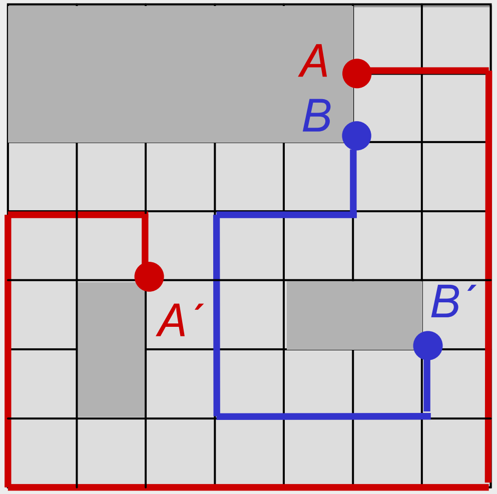
Effect of net ordering on routability



Optimal routing of net *A*



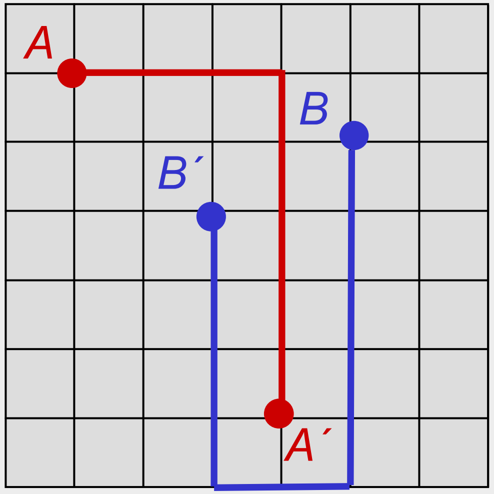
Optimal routing of net *B*



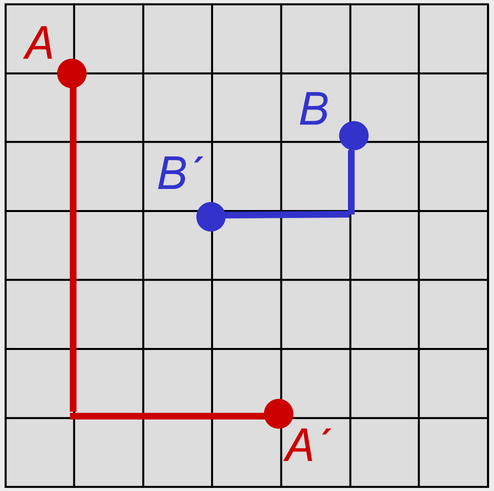
Nets *A* and *B* can be routed only with detours

7.2 Net Ordering in Area Routing

Effect of net ordering on total wirelength



Routing net *A* first



Routing net *B* first

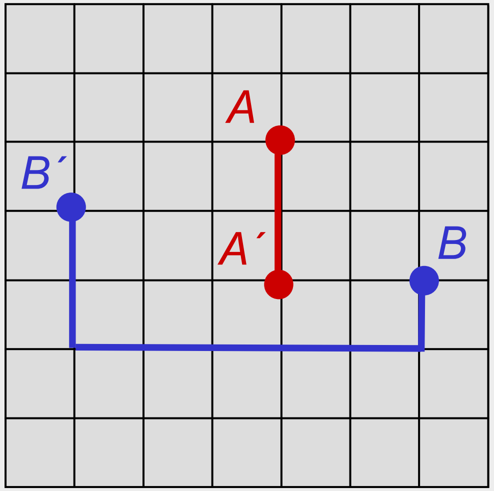
7.2 Net Ordering in Area Routing

- For *n* nets, there are *n*! possible net orderings

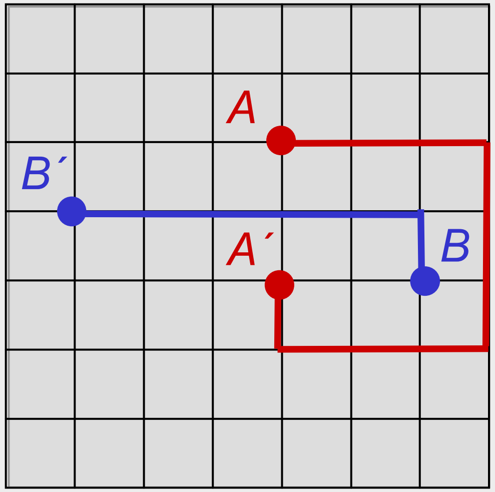
⇒ Constructive heuristics are used

Net Ordering in Area Routing

- **Rule 1:** For two nets *i* and *j*, if *aspect ratio* (*i*) > *aspect ratio* (*j*), then *i* is routed before *j*



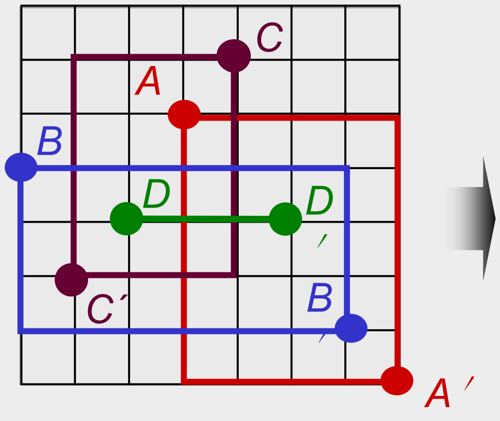
Net *A* has a higher aspect ratio of its bounding box; routing *A* first results in shorter total wirlength



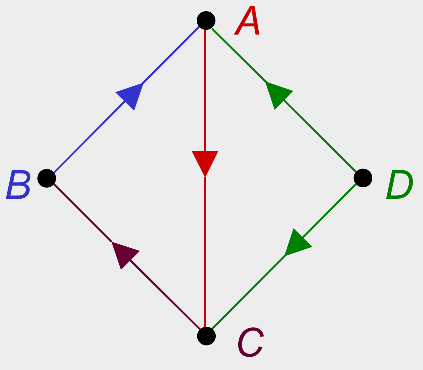
Routing net *B* first results in longer total wirelength

7.2 Net Ordering in Area Routing

- **Rule 2:** For two nets *i* and *j*, if the pins of *i* are contained within *MBB*(*j*), then *i* is routed before *j*

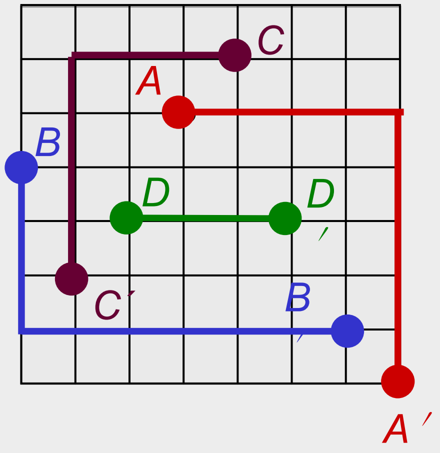


Constraint Graph



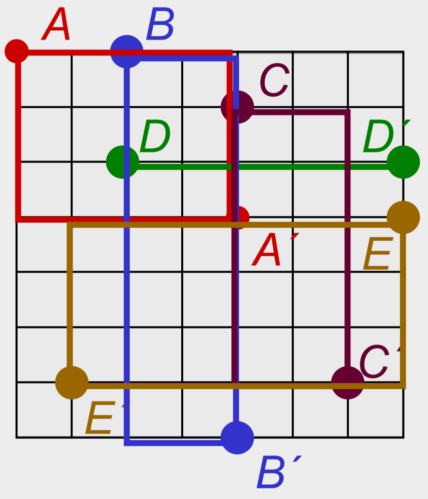
Net Ordering

Ordering *D*-*A*-*C*-*B*
or *D*-*C*-*B*-*A*
(not *D*-*B*-*A*-*C*)

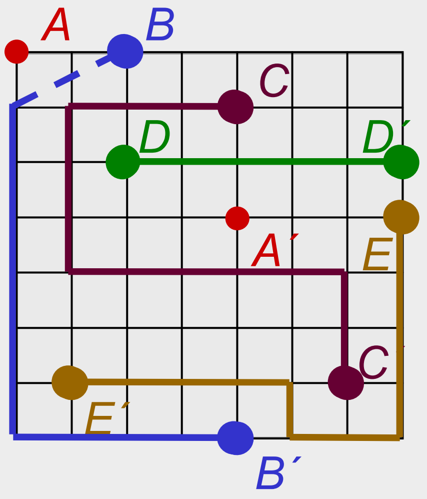


7.2 Net Ordering in Area Routing

- **Rule 3:** Let $\Pi(net)$ be the number of pins within *MBB*(*net*) for net *net*. For two nets *i* and *j*, if $\Pi(i) < \Pi(j)$, then *i* is routed before *j*.
  - For each net, consider the pins of other nets within its bounding box
  - The net with the smallest number of such pins is routed first
  - Ties are broken based on the number of pins that are contained within the bounding box and on its edge



	Pins Inside (Edge)	$\pi(net)$
MBB (*A*)	*D* (*B*,*C*,*D*)	3
B	- (*A*,*C*,*D*)	3
C	- (*A*)	1
D	- (-)	0
E	- (*A*,*C*)	2



7.3 Non-Manhattan Routing

7.1 Introduction to Area Routing

7.2 Net Ordering in Area Routing

→ 7.3 Non-Manhattan Routing

- 7.3.1 Octilinear Steiner Trees
- 7.3.2 Octilinear Maze Search

7.4 Basic Concepts in Clock Networks

- 7.4.1 Terminology
- 7.4.2 Problem Formulations for Clock-Tree Routing

7.5 Modern Clock Tree Synthesis

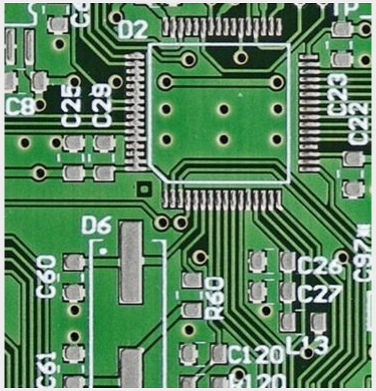
- 7.5.1 Constructing Trees with Zero Global Skew
- 7.5.2 Clock Tree Buffering in the Presence of Variation

7.3 Non-Manhattan Routing

- Allow 45- or 60-degree segments in addition to horizontal and vertical segments

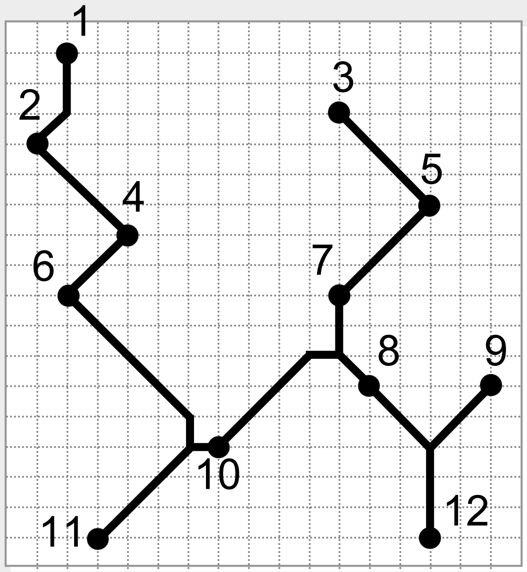
- λ-geometry, where λ represents the number of possible routing directions and the angles π / λ at which they can be oriented
  - $\lambda = 2$ (90 degrees): Manhattan routing (four routing directions)
  - $\lambda = 3$ (60 degrees): Y-routing (six routing directions)
  - $\lambda = 4$ (45 degrees): X-routing (eight routing directions)

- Non-Manhattan routing is primarily employed on printed circuit boards (PCBs)



7.3.1 Octilinear Steiner Trees

- Route planning using octilinear Steiner minimum trees (OSMT)
- Generalize rectilinear Steiner trees by allowing segments that extend in eight directions
- More freedom when placing Steiner points



7.3.1 Octilinear Steiner Trees

Octilinear Steiner Tree Algorithm

Input: set of all pins *P* and their coordinates

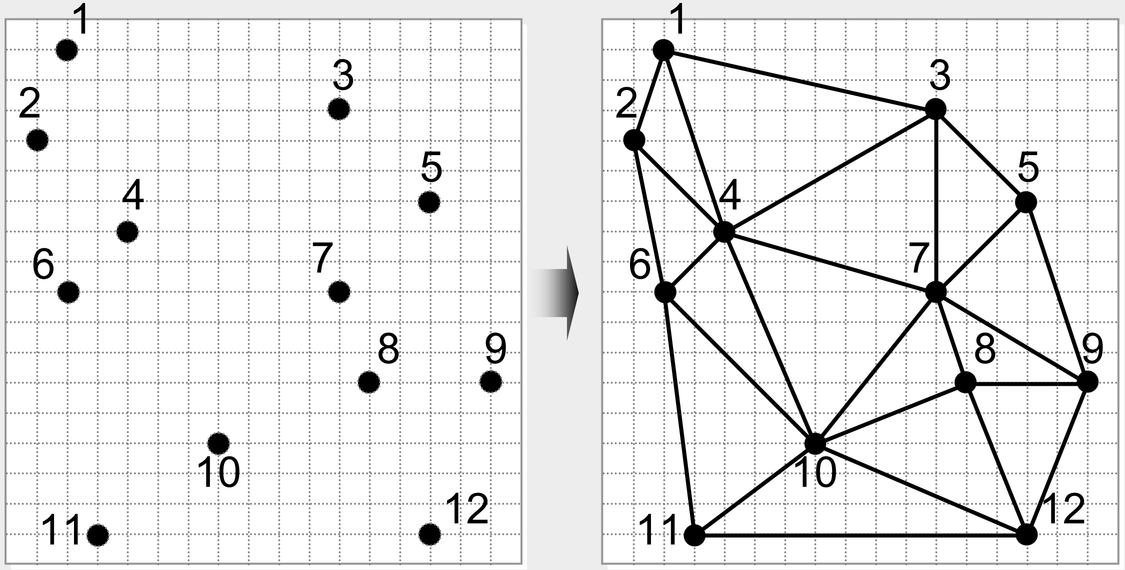
Output: heuristic octilinear minimum Steiner tree *OST*

```
OST = Ø
T = set of all three-pin nets of P found by Delaunay triangulation
sortedT = SORT(T,minimum octilinear distance)
for (i = 1 to |sortedT|)
    subT = ROUTE(sortedT[i])          // route minimum tree over subT
    ADD(OST,subT)                     // add route to existing tree
    IMPROVE(OST,subT)                 // locally improve OST based on subT
```

T.-Y.; Chang, et. al.: Multilevel Full-Chip Routing for the X-Based Architecture

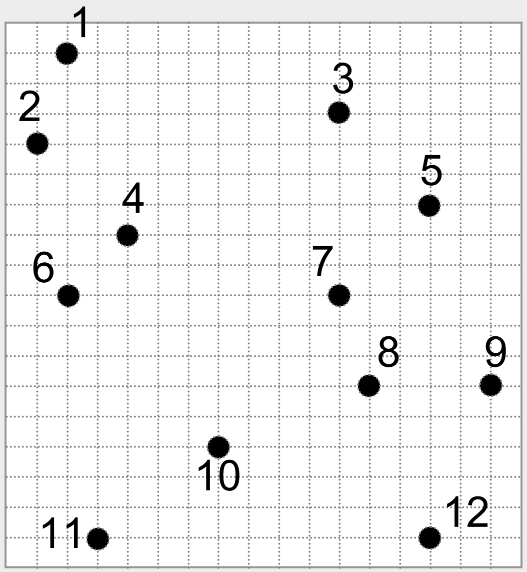
7.3.1 Octilinear Steiner Trees

(1) Triangulate

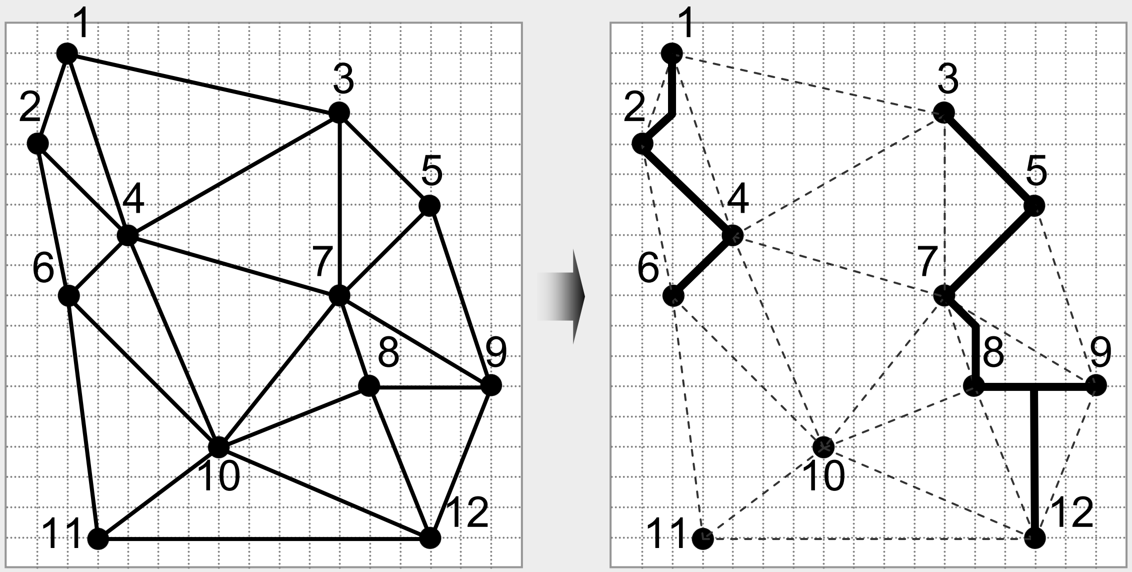


7.3.1 Octilinear Steiner Trees

(1) Triangulate

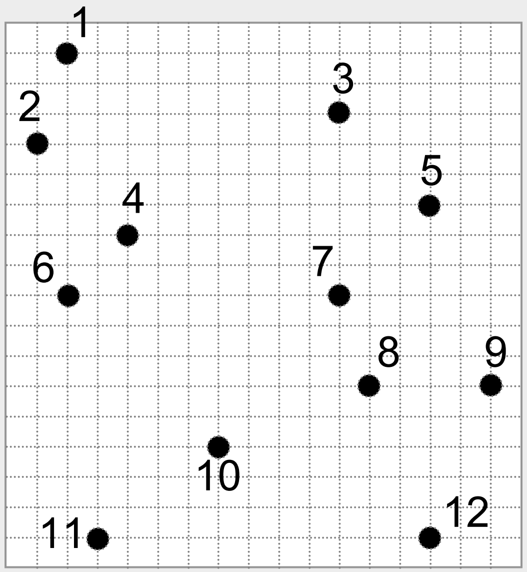


(2) Add route to existing tree

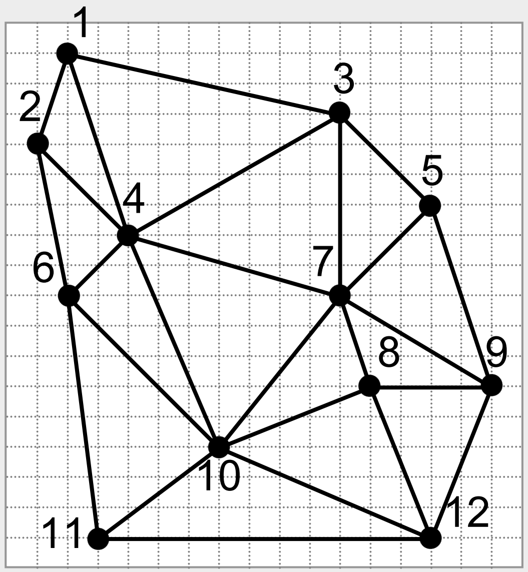


7.3.1 Octilinear Steiner Trees

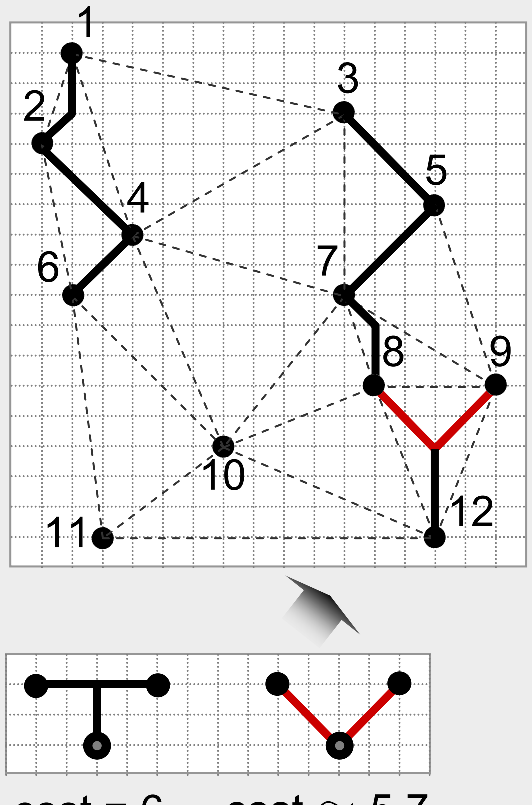
(1) Triangulate



(2) Add route to existing tree



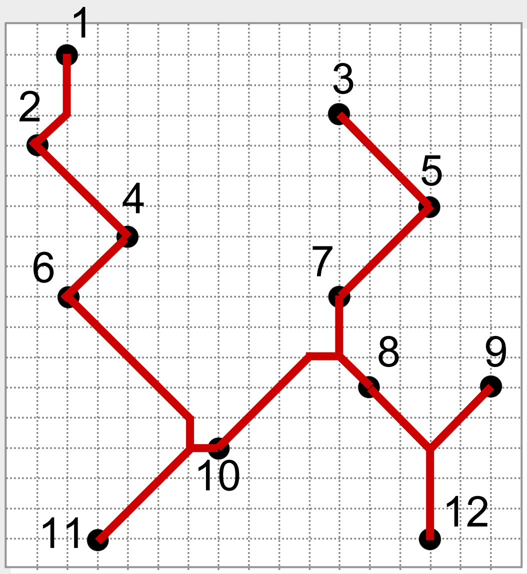
(3) Locally improve OST



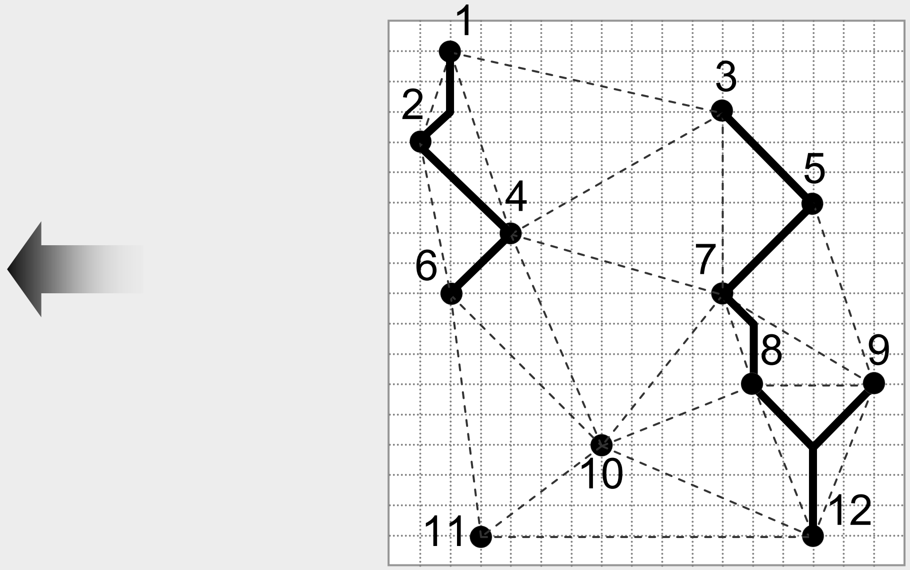
cost = 6 cost ≈ 5.7

7.3.1 Octilinear Steiner Trees

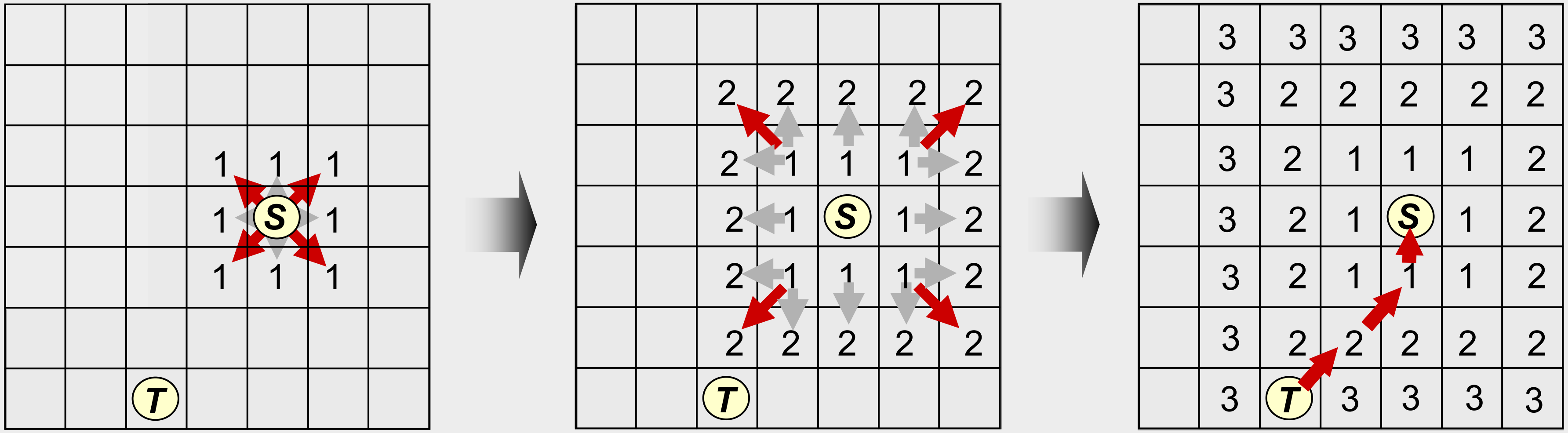
Final OST after merging all subtrees



(3) Locally improve OST



7.3.2 Octilinear Maze Search

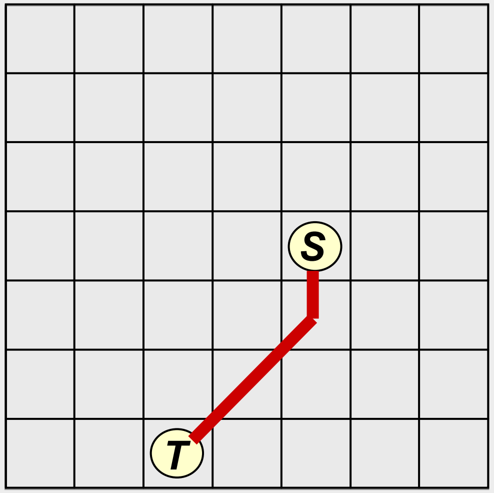


Expansion (1)

Expansion (2)

Backtracing

7.3.2 Octilinear Maze Search



7.4 Basic Concepts in Clock Networks

7.1 Introduction to Area Routing

7.2 Net Ordering in Area Routing

7.3 Non-Manhattan Routing

7.3.1 Octilinear Steiner Trees

7.3.2 Octilinear Maze Search

➡ 7.4 Basic Concepts in Clock Networks

7.4.1 Terminology

7.4.2 Problem Formulations for Clock-Tree Routing

7.5 Modern Clock Tree Synthesis

7.5.1 Constructing Trees with Zero Global Skew

7.5.2 Clock Tree Buffering in the Presence of Variation

7.4.1 Terminology

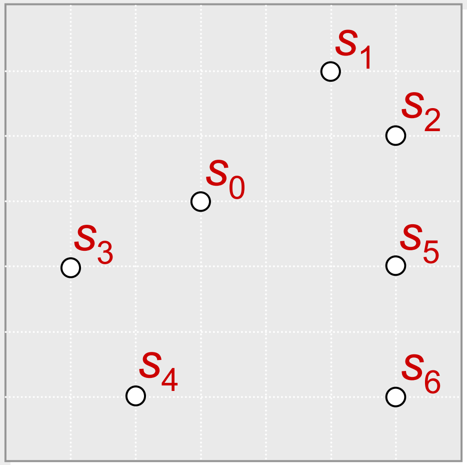
- A clock routing instance (clock net) is represented by n+1 terminals, where s_0 is designated as the source, and $S = \{s_1, s_2, \ldots, s_n\}$ is designated as sinks
  - Let s_i, $0 \leqslant i \leqslant n$, denote both a terminal and its location

- A clock routing solution consists of a set of wire segments that connect all terminals of the clock net, so that a signal generated at the source propagates to all of the sinks
  - Two aspects of clock routing solution: topology and geometric embedding

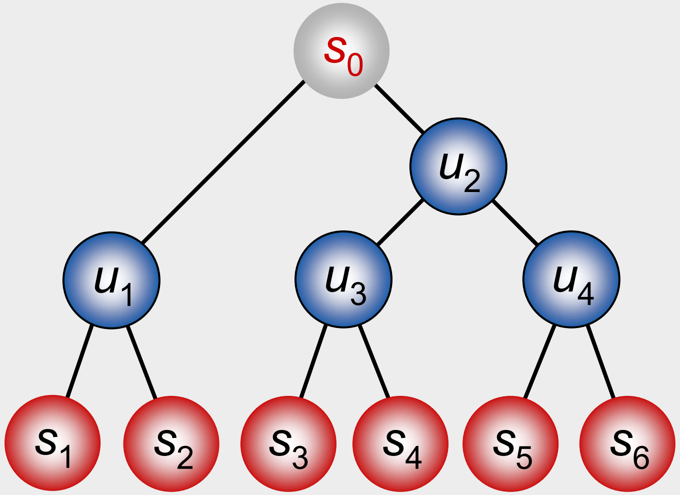
- The clock-tree topology (clock tree) is a rooted binary tree G with n leaves corresponding to the set of sinks
  - Internal nodes = Steiner points

7.4.1 Terminology

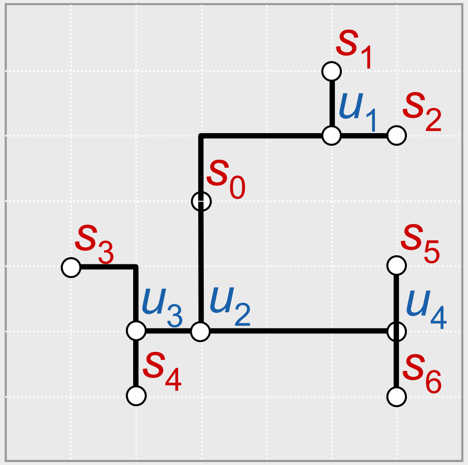
Clock routing problem instance



Connection topology



Embedding



7.4.1 Terminology

- Clock skew: (maximum) difference in clock signal arrival times between sinks

$$skew(T) = \max_{s_i, s_j \in S} |\, t(s_0, s_i) - t(s_0, s_j) \,|$$

- Local skew: maximum difference in arrival times of the clock signal at the clock pins of two or more related sinks
  - Sinks within distance *d* > 0
  - Flip-flops or latches connected by a directed signal path

- Global skew: maximum difference in arrival times of the clock signal at the clock pins of any two (related or unrelated) sinks
  - Difference between shortest and longest source-sink path delays in the clock distribution network
  - The term “skew” typically refers to “global skew”

7.4.2 Problem Formulations for Clock-Tree Routing

- Zero skew: zero-skew tree (ZST)
  - ZST problem

- Bounded skew: true ZST may not be necessary in practice
  - Signoff timing analysis is sufficient with a non-zero skew bound
  - In addition to final (signoff) timing, this relaxation can be useful with intermediate delay models when it facilitates reductions in the length of the tree
  - Bounded-Skew Tree (BST) problem

- Useful skew: correct chip timing only requires control of the local skews between pairs of interconnected flip-flops or latches
  - Useful skew formulation is based on analysis of local skew constraints

7.5 Modern Clock Tree Synthesis

7.1 Introduction to Area Routing

7.2 Net Ordering in Area Routing

7.3 Non-Manhattan Routing

- 7.3.1 Octilinear Steiner Trees
- 7.3.2 Octilinear Maze Search

7.4 Basic Concepts in Clock Networks

- 7.4.1 Terminology
- 7.4.2 Problem Formulations for Clock-Tree Routing

7.5 Modern Clock Tree Synthesis

- 7.5.1 Constructing Trees with Zero Global Skew
- 7.5.2 Clock Tree Buffering in the Presence of Variation

7.5 Modern Clock Tree Synthesis

- A clock tree should have low skew, while delivering the same signal to every sequential gate

- Clock tree synthesis is performed in two steps:

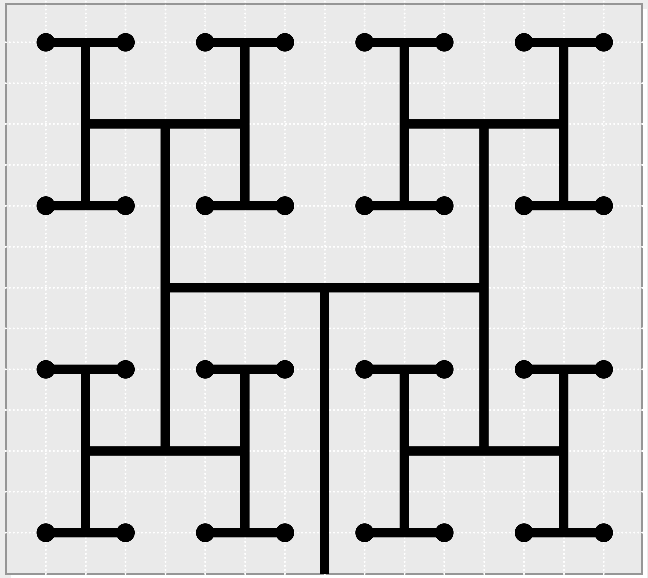
(1) Initial tree construction (Sec. 7.5.1) with one of these scenarios

- Construct a regular clock tree, largely independent of sink locations
- Simultaneously determine a topology and an embedding
- Construct only the embedding, given a clock-tree topology as input

(2) Clock buffer insertion and several subsequent skew optimizations (Sec. 7.5.2)

7.5.1 Constructing Trees with Zero Global Skew

H-tree



- Exact zero skew due to the symmetry of the H-tree
- Used for top-level clock distribution, not for the entire clock tree
  - Blockages can spoil the symmetry of an H-tree
  - Non-uniform sink locations and varying sink capacitances also complicate the design of H-trees

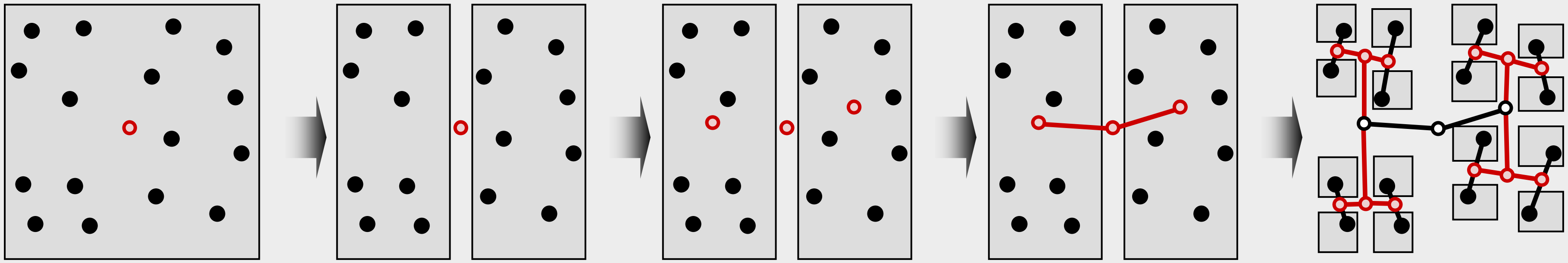
7.5.1 Constructing Trees with Zero Global Skew

Method of Means and Medians (MMM)

- Can deal with arbitrary locations of clock sinks
- Basic idea:
  - Recursively partition the set of terminals into two subsets of equal size (median)
  - Connect the center of gravity (COG) of the set to the centers of gravity of the two subsets (the mean)

7.5.1 Constructing Trees with Zero Global Skew

Method of Means and Medians (MMM)



Find the center of gravity○

Partition *S* by the median

Find the center of gravity for the left and right subsets of *S*

Connect the center of gravity of *S* with the centers of gravity of the left and right subsets

Final result after recursively performing MMM on each subset

7.5.1 Constructing Trees with Zero Global Skew

Method of Means and Medians (MMM)

Input: set of sinks *S*, empty tree *T*

Output: clock tree *T*

```
if (|$S$| $\leqslant$ 1)
    return
($x_0$,$y_0$) = ($x_c$($S$),$y_c$($S$))             // center of mass for $S$
($S_A$,$S_B$) = PARTITION($S$)             // median to determine $S_A$ and $S_B$
($x_A$,$y_A$) = ($x_c$($S_A$),$y_c$($S_A$))         // center of mass for $S_A$
($x_B$,$y_B$) = ($x_c$($S_B$),$y_c$($S_B$))         // center of mass for $S_B$
ROUTE($T$,$x_0$,$y_0$,$x_A$,$y_A$)             // connect center of mass of $S$ to
ROUTE($T$,$x_0$,$y_0$,$x_B$,$y_B$)             //   center of mass of $S_A$ and $S_B$
BASIC_MMM($S_A$,$T$)                 // recursively route $S_A$
BASIC_MMM($S_B$,$T$)                 // recursively route $S_B$
```

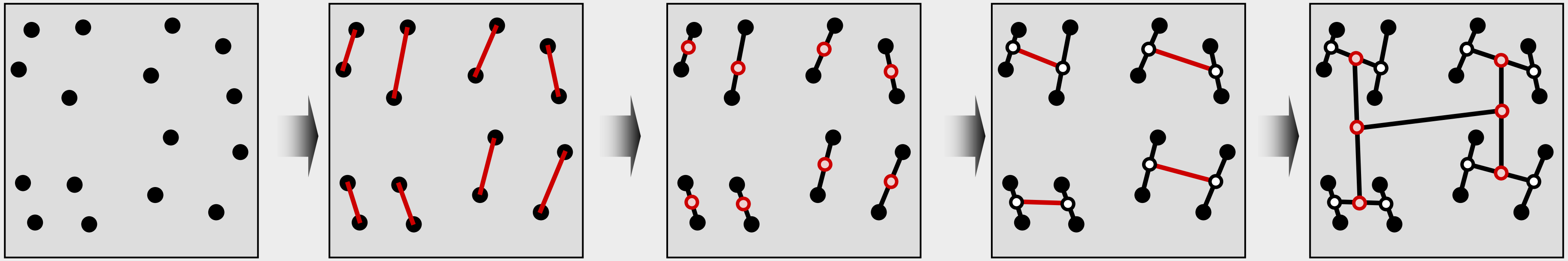
7.5.1 Constructing Trees with Zero Global Skew

Recursive Geometric Matching (RGM)

- RGM proceeds in a bottom-up fashion
  - Compare to MMM, which is a top-down algorithm
- Basic idea:
  - Recursively determine a minimum-cost geometric matching of *n* sinks
  - Find a set of *n* / 2 line segments that match *n* endpoints and minimize total length (subject to the matching constraint)
  - After each matching step, a balance or tapping point is found on each matching segment to preserve zero skew to the associated sinks
  - The set of *n* / 2 tapping points then forms the input to the next matching step

7.5.1 Constructing Trees with Zero Global Skew

Recursive Geometric Matching (RGM)



Set of *n* sinks *S*

Min-cost geometric matching

Find balance or tapping points (point that achieves zero skew in the subtree, not always midpoint)

Min-cost geometric matching

Final result after recursively performing RGM on each subset

7.5.1 Constructing Trees with Zero Global Skew

Recursive Geometric Matching (RGM)

Input: set of sinks *S*, empty tree *T*

Output: clock tree *T*

```
if ($|S| \leqslant 1$)
    return
$M$ = min-cost geometric matching over $S$
$S'$ = Ø
foreach ($<P_i,P_j> \in M$)
    $TP_i$ = subtree of $T$ rooted at $P_i$
    $TP_j$ = subtree of $T$ rooted at $P_j$
    $tp$ = tapping point on $(P_i,P_j)$        // point that minimizes the skew of
                                               //   the tree $T_{tp} = T_{Pi} \cup T_{Pj} \cup (P_i,P_j)$
    ADD($S'$,$tp$)                             // add $tp$ to $S'$
    ADD($T$,$(P_i,P_j)$)                       // add matching segment $(P_i,P_j)$ to $T$
if ($|S|$ % 2 == 1)                            // if $|S|$ is odd, add unmatched node
    ADD($S'$, unmatched node)
RGM($S'$,$T$)                                  // recursively call RGM
```

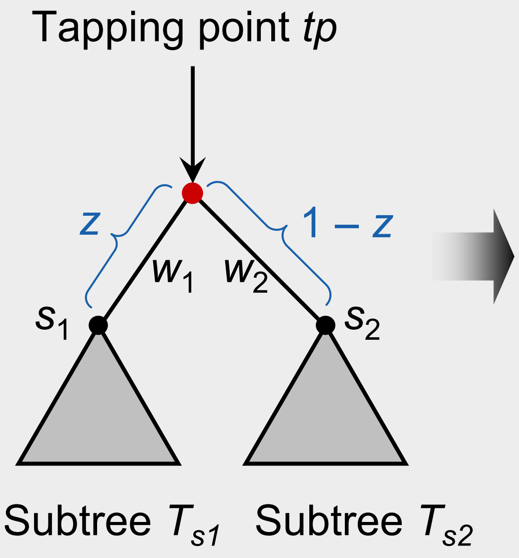
7.5.1 Constructing Trees with Zero Global Skew

Exact Zero Skew

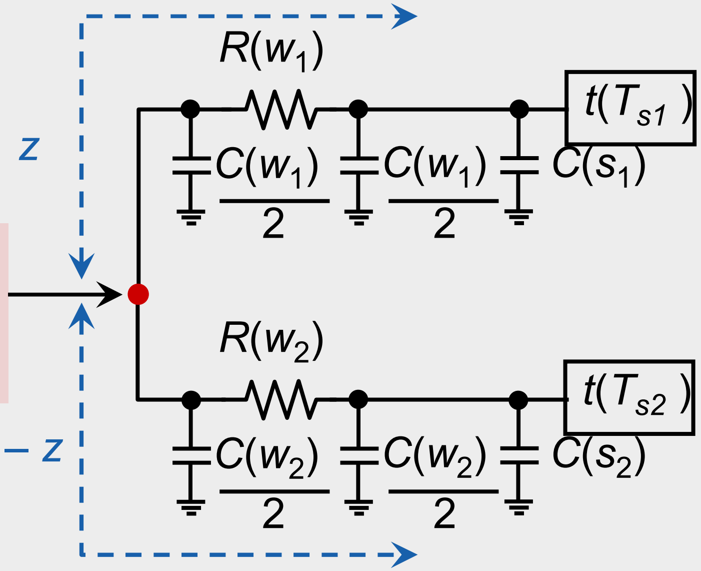
- Adopts a bottom-up process of matching subtree roots and merging the corresponding subtrees, similar to RGM
- Two important improvements:
  - Finds exact zero-skew tapping points with respect to the Elmore delay model rather than the linear delay model
  - Maintains exact delay balance even when two subtrees with very different source-sink delays are matched (by wire elongation)

7.5.1 Constructing Trees with Zero Global Skew

Exact Zero Skew



Tapping point *tp*, where Elmore delay to sinks is equalized



7.5.1 Constructing Trees with Zero Global Skew

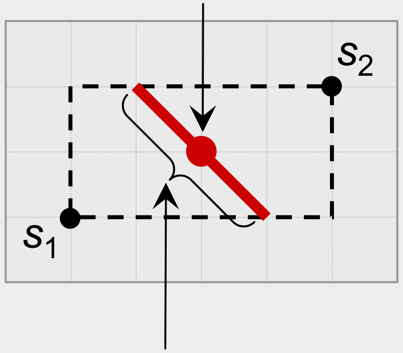
Deferred-Merge Embedding (DME)

- Defers the choice of merging (tapping) points for subtrees of the clock tree
- Needs a tree topology as input
- Weakness in earlier algorithms:
  - Determine locations of internal nodes of the clock tree too early; once a centroid is found, it is never changed
- Basic idea:
  - Two sinks in general position will have an infinite number of midpoints, creating a tilted line segment – Manhattan arc
  - Manhattan arc: same minimum wirelength and exact zero skew
  - Selection of embedding points for internal nodes on Manhattan arc will be delayed for as long as possible

7.5.1 Constructing Trees with Zero Global Skew

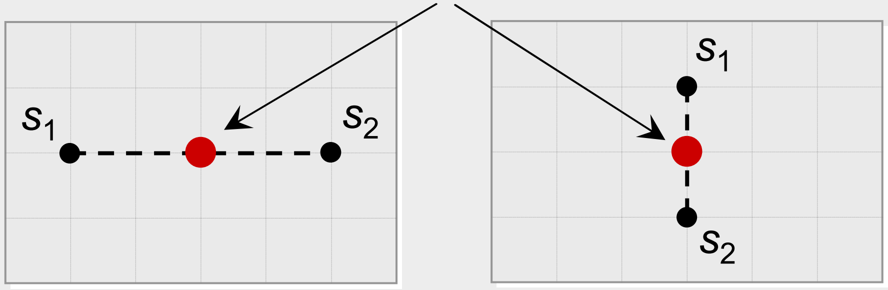
Deferred-Merge Embedding (DME)

Euclidean midpoint



Locus of all Manhattan midpoints is a Manhattan arc in the Manhattan geometry

Euclidean midpoint



Sinks are aligned, hence, Manhattan arc has zero length

7.5.1 Constructing Trees with Zero Global Skew

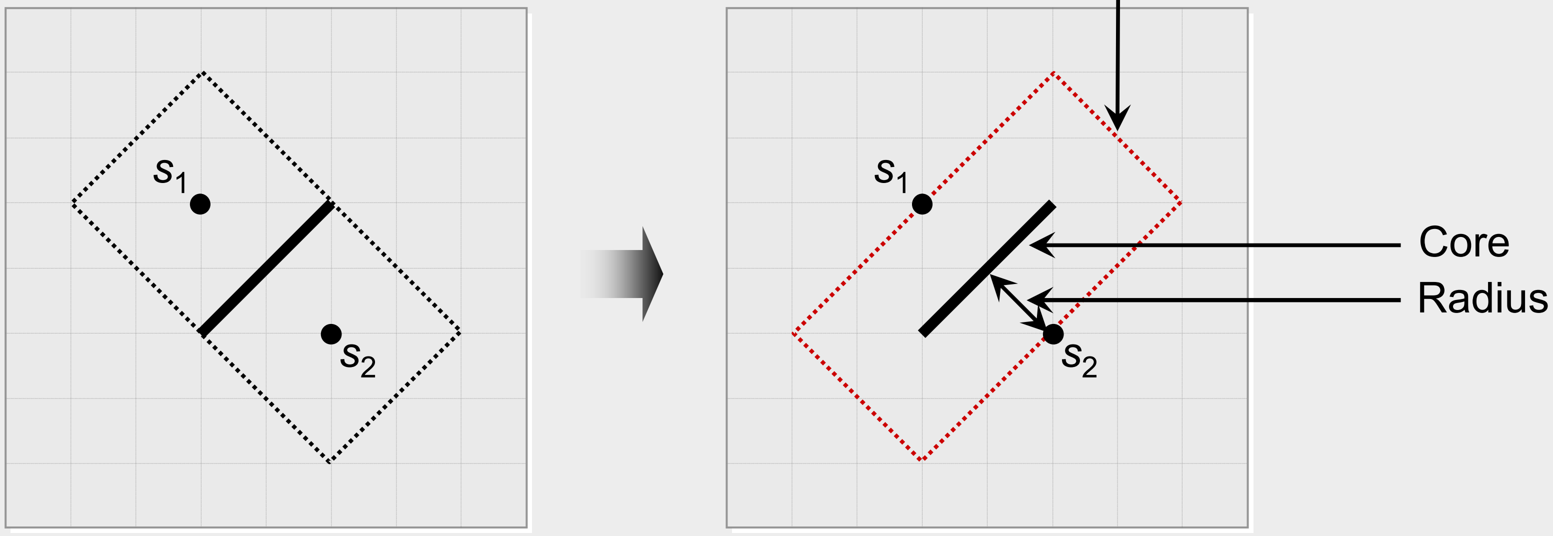
Deferred-Merge Embedding (DME)

- Embeds internal nodes of the given topology *G* via a two-phase process
- First phase is bottom-up
  - Determines all possible locations of internal nodes of *G* consistent with a minimum-cost ZST *T*
  - Output: “tree of line segments”, with each line segment being the locus of possible placements of an internal node of *T*
- Second phase is top-down
  - Chooses the exact locations of all internal nodes in *T*
  - Output: fully embedded, minimum-cost ZST with topology *G*

7.5.1 Constructing Trees with Zero Global Skew

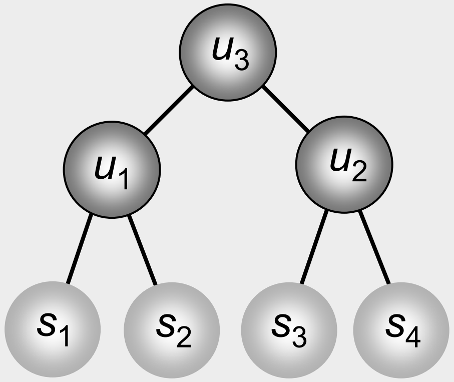
Deferred-Merge Embedding (DME)

Tilted Rectangular Region (TRR) for the Manhattan arc of s_1 and s_2 with a radius of two units

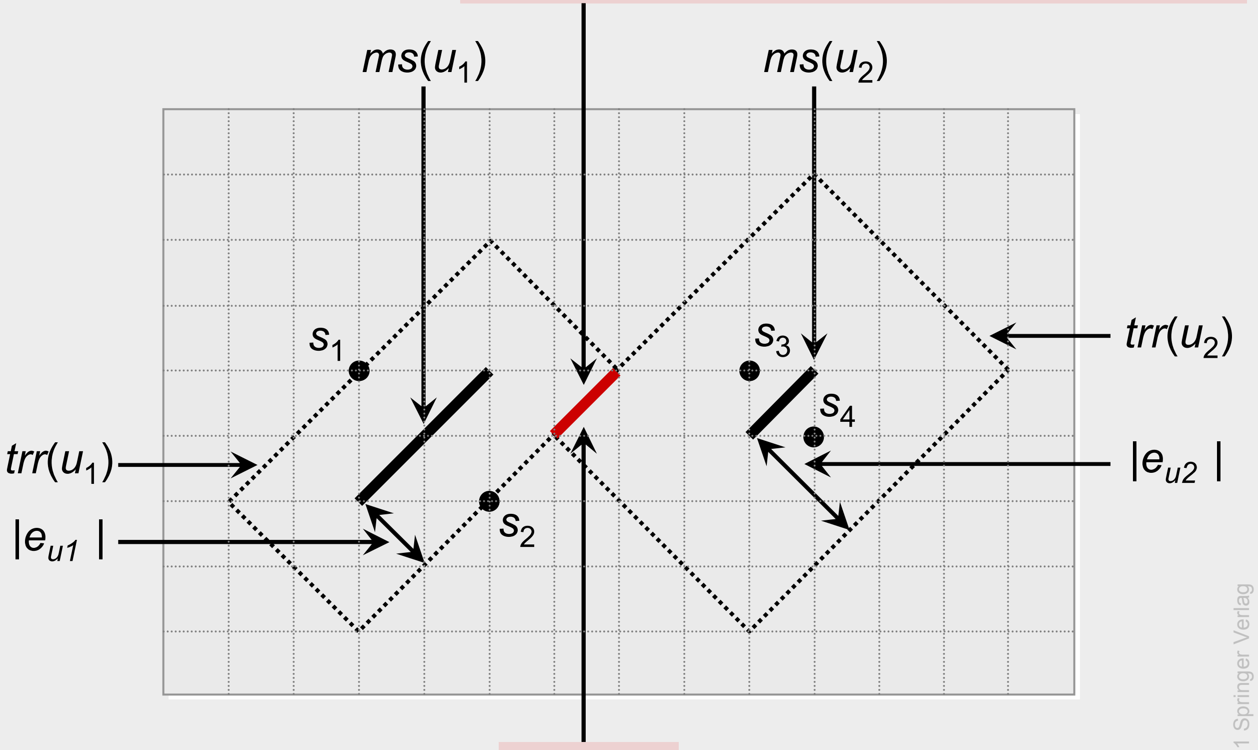


7.5.1 Constructing Trees with Zero Global Skew

Deferred-Merge Embedding (DME)



Merging segment for node u_3 (the parent of nodes u_1 and u_2) is the locus of feasible locations of u_3 with zero skew and minimum wirelength

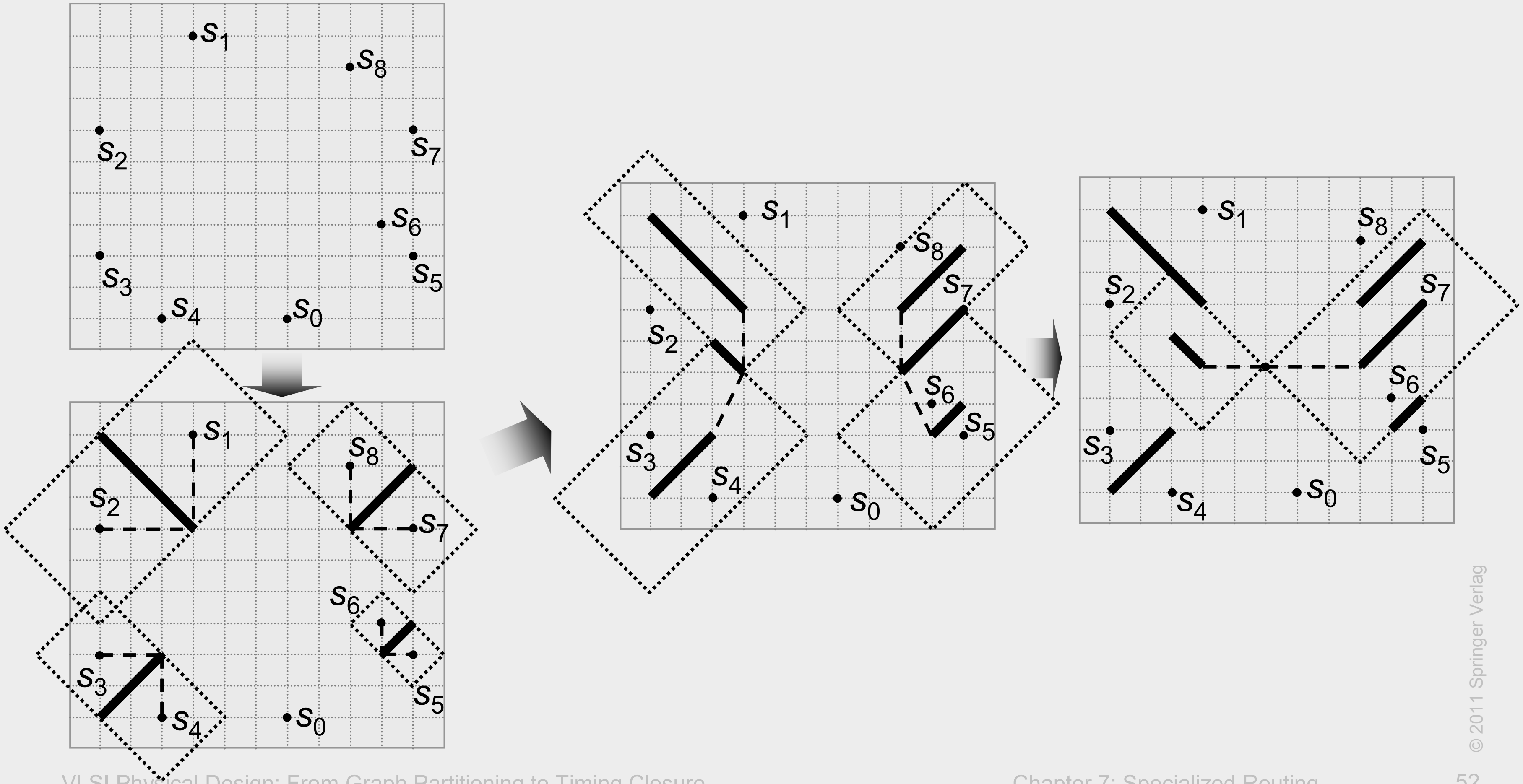


$ms(u_3)$

7.5.1 Constructing Trees with Zero Global Skew

Deferred-Merge Embedding (DME)

Build Tree of Segments Algorithm (DME Bottom-Up Phase)



7.5.1 Constructing Trees with Zero Global Skew

Deferred-Merge Embedding (DME)

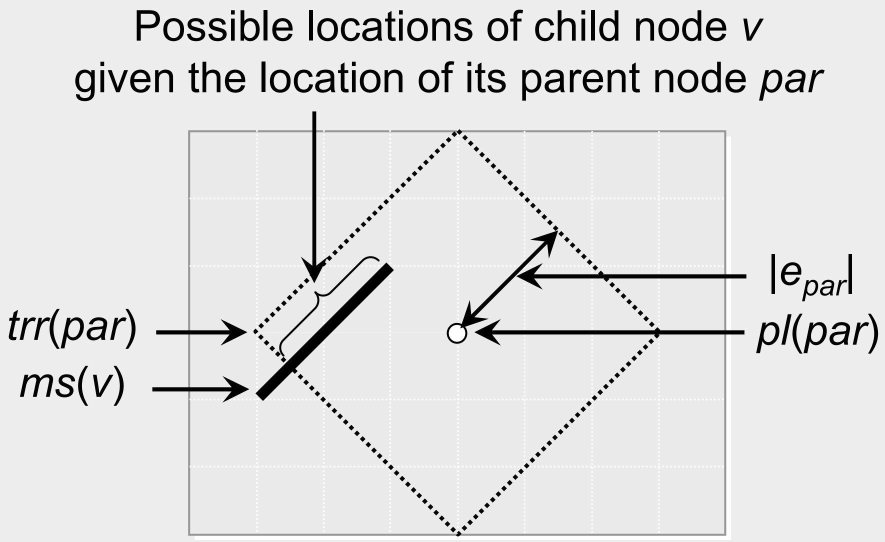
Build Tree of Segments Algorithm (DME Bottom-Up Phase)

Input: set of sinks *S* and tree topology *G*(*S*,*Top*)
Output: merging segments *ms*(*v*) and edge lengths $|e_v|$, $v \in G$

```
foreach (node $v \in G$, in bottom-up order)
   if ($v$ is a sink node)                    // if $v$ is a terminal, then $ms(v)$ is a
      $ms[v] = PL(v)$                         //   zero-length Manhattan arc
   else                                       // otherwise, if $v$ is an internal node,
      $(a,b)$ = CHILDREN($v$)                 //   find $v$'s children and
      CALC_EDGE_LENGTH($e_a,e_b$)             //   calculate the edge length
      $trr[a][core] = MS(a)$                  // create $trr(a)$ – find merging segment
      $trr[a][radius] = |e_a|$                //   and radius of $a$
      $trr[b][core] = MS(b)$                  // create $trr(b)$ – find merging segment
      $trr[b][radius] = |e_b|$                //   and radius of $b$
      $ms[v] = trr[a] \cap trr[b]$            // merging segment of $v$
```

Deferred-Merge Embedding (DME)

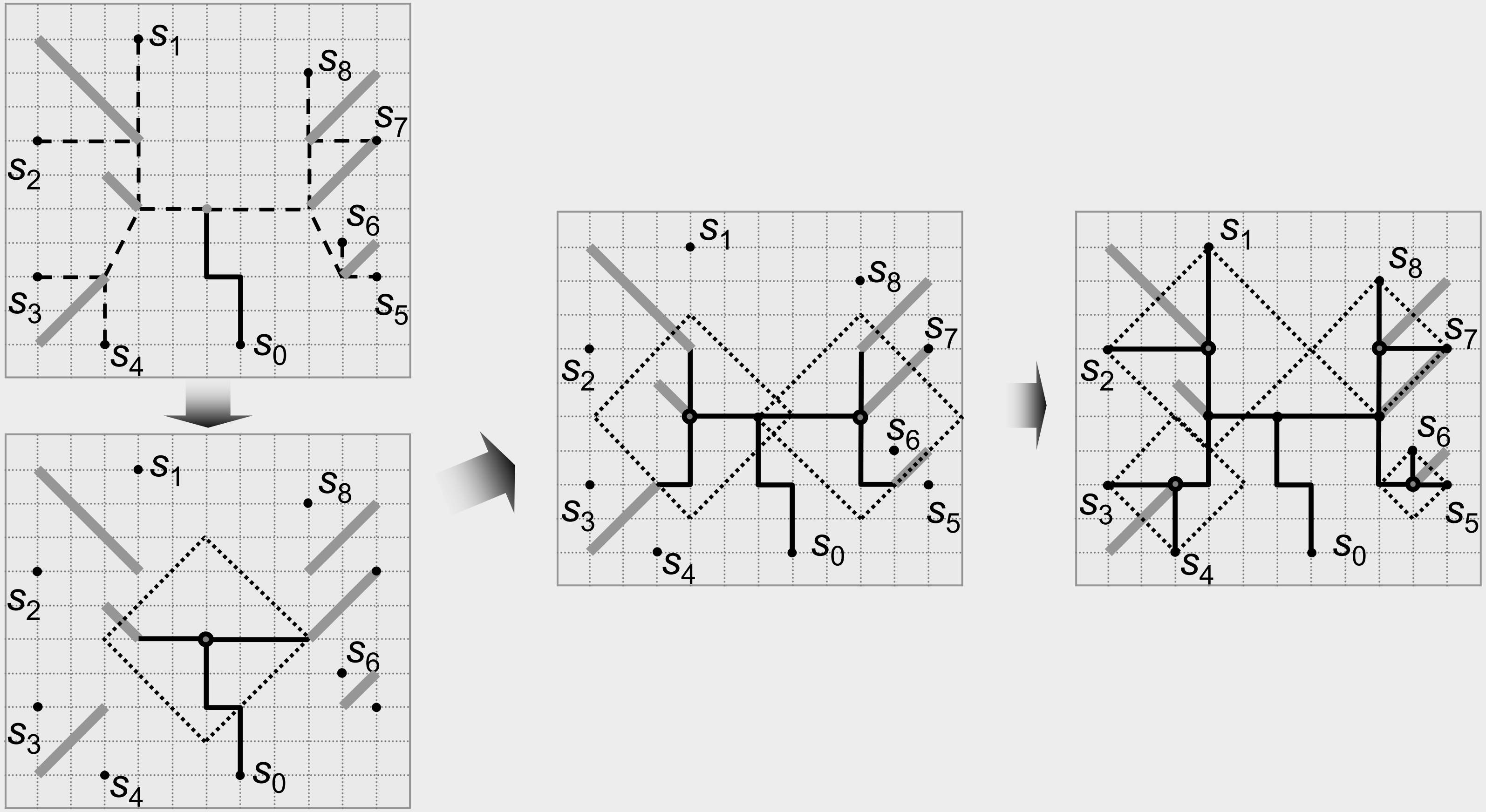
Find Exact Locations (DME Top-Down Phase)


Possible locations of child node *v*
given the location of its parent node *par*
trr(*par*)
ms(*v*)
$|e_{par}|$
pl(*par*)

7.5.1 Constructing Trees with Zero Global Skew

Deferred-Merge Embedding (DME)

Find Exact Locations (DME Top-Down Phase)



7.5.1 Constructing Trees with Zero Global Skew

Deferred-Merge Embedding (DME)

Find Exact Locations (DME Top-Down Phase)

Input: set of sinks *S*, tree topology *G*, outputs of DME bottom-up phase
Output: minimum-cost zero-skew tree *T* with topology *G*

```
foreach (non-sink node $v \in G$ top-down order)
    if ($v$ is the root)
        $loc$ = any point in $ms(v)$
    else
        $par$ = PARENT($v$)                  // $par$ is the parent of $v$
        $trr[par][core]$ = $PL(par)$         // create $trr(par)$ – find merging segment
        $trr[par][radius]$ = $|e_v|$         //   and radius of $par$
        $loc$ = any $point$ in $ms[v] \cap trr[par]$
  $pl[v]$ = $loc$
```

7.5.2 Clock Tree Buffering in the Presence of Variation

- To address challenging skew constraints, a clock tree undergoes several optimization steps, including
  - Geometric clock tree construction
  - Initial clock buffer insertion
  - Clock buffer sizing
  - Wire sizing
  - Wire snaking
- In the presence of process, voltage, and temperature variations, such optimizations require modeling the impact of variations
  - Variation model encapsulates the different parameters, such as width and thickness, of each library element as well-defined random variables

Summary of Chapter 7 – Area Routing

- Area routing: avoiding the division into global and detailed routing
  - Doing everything at once, subject to design rules
  - Small netlists with complicated constraints
  - Analog, MCM and PCB routing

- Manhattan vs Euclidean paths
  - Euclidean paths are no longer than Manhattan, usually shorter
  - Unique Euclidean shortest path
  - Multiple Manhattan paths
  - When Euclidean shortest paths intersect, there may exist Manhattan shortest paths that do not (not vice versa)

- Net ordering is important in area routing
  - Rule 1: nets with higher aspect ratio (less flexible) routed first
  - Rule 2: nets surrounded by other nets (more constrained) routed first
  - Rule 3: nets with more pins inside other net's bounding boxes routed first

Summary of Chapter 7 – Non-Manhattan Tree Routing

- Recall that Manhattan routing is dictated by the limitations of modern semiconductor manufacturing for thin wires

- PCB routing is not subject to those limitations
  - Can use shorter connections

- Non-Manhattan connections
  - Diagonal (45- or 60-degree) segments in addition to horizontal and vertical segments
  - Create more freedom to place Steiner points

- Octilinear Steiner Tree construction
  - Algorithms are generally adapted from the Manhattan case
  - Should produce results that are at least as good as the Manhattan case

Summary of Chapter 7 – Clock Network Routing

- Similar to signal-net routing, except for
  - Very large numbers of sinks
  - The need to equalize propagation delays from the root to sinks
  - Longer routes (to satisfy the equalization constraint)
  - Typical algorithms determine topology first, then geometric embedding

- Clock skew
  - Consider propagation delay from the root to each sink
  - Skew is the maximal pairwise difference between delays (over all pairs of sinks)
  - May be limited to sinks that are within distance $d > 0$ (local skew)

- For a specified wire delay model
  - ZST: Zero-Skew Tree routing requires that skew = *0*
  - BST: Bounded-Skew Tree routing requires that skew < *Bound*

Summary of Chapter 7 – Modern Clock Tree Synthesis

- Initial clock tree construction
  - Topology determination (MMM or RGM)
  - DME embedding (different flavors for ZST and BST)
  - Working with the Elmore delay model requires more effort than working with linear delay models
- Geometric obstacles (e.g., macros)
  - May require detours
  - Can be handled during DME (complicated) or during post-processing (often achieves as good results)
- Clock-tree optimization
  - Buffer insertion
  - Buffer sizing
  - Wire sizing
  - Wire snaking by small amounts
  - Decreasing the impact of process variability