Repeater Insertion

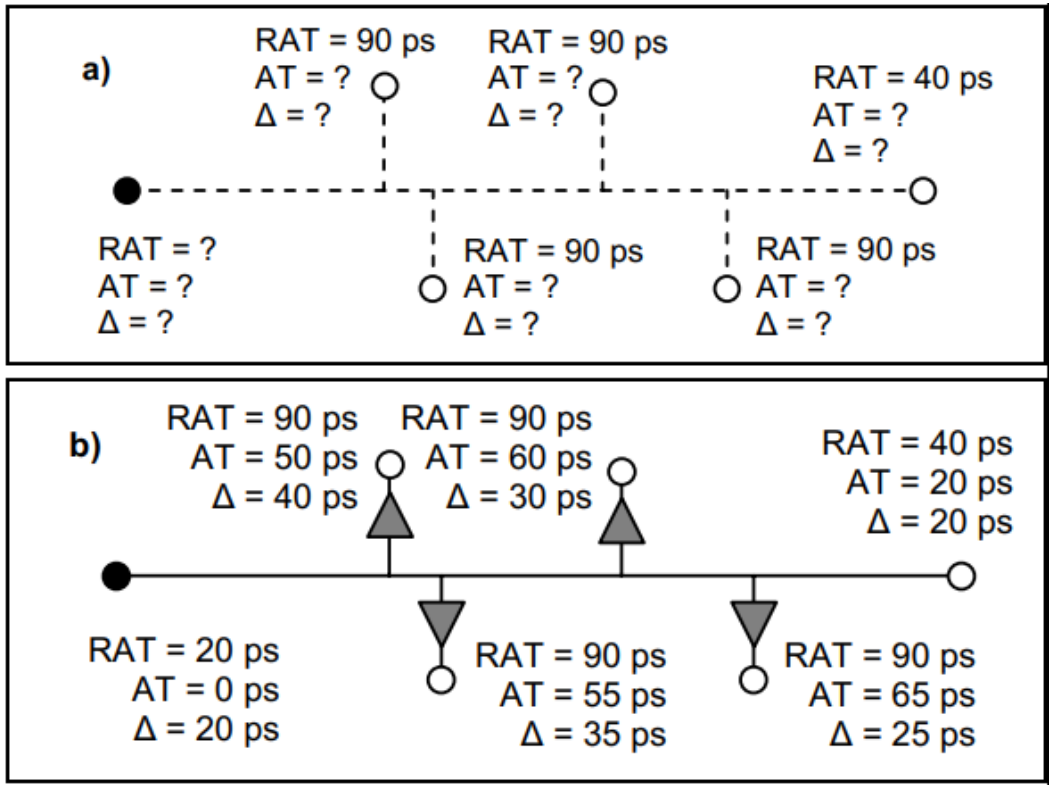
VLSI CAD

Motivation

- As feature size goes smaller, gate delay and wire delay scale in opposite directions:
  - Smaller devices imply less gate delay
  - Thinner wires lead to wire resistance increase and greater wire delay
    - Larger wire aspect ratio and better dielectrics only partially alleviate the issue
- The chip size stays the same, so does overall wire length
- The percentage of repeaters and nets that require repeaters increases each process generation

Problem to solve

- Given:
  - Source and sink pin locations
  - Sink pin required arrival times and capacitances
  - Source pin resistance
  - Repeater library
  - Routing tree
  - Unit wire resistance and capacitance
- Find:
  - Repeater locations such that slack at the source is maximized



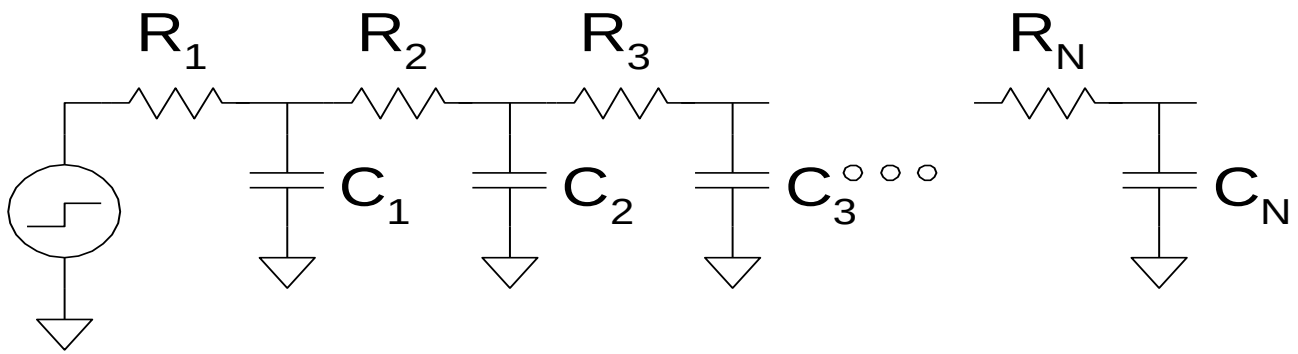
Interconnect delay: Elmore model

- Pullup or pulldown network modeled as *RC ladder*
- Elmore delay of RC ladder:

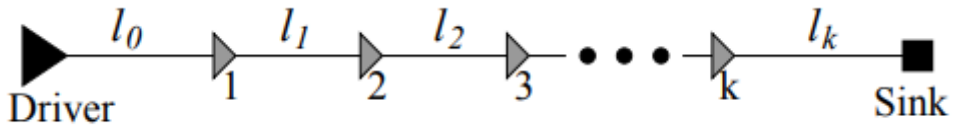
$$\tau_{DN} = \sum_{k=1}^{N} C_k R_{kk} = \sum_{k=1}^{N} CkR = RC\frac{N(N+1)}{2} = rcL^2\frac{N+1}{2N} \xrightarrow[N\to\infty]{} \frac{rcL^2}{2}$$

$$R = \frac{r * L}{N}$$

$$C = \frac{c * L}{N}$$



Optimization of two-pin net



- Goal: using repeater b minimize driver-to-sink delay by deciding the number of repeaters k and their positions
- Simplification assumptions:
  - Driver resistance is the same as repeater resistance R_b
  - Sink capacitance is the same as repeater input capacitance C_b
- Elmore delay for wire with k repeaters and $\vec{l} = (l_0, \dots l_k)$ segments:

$$t(\vec{l}) = \sum_{i=0}^{k} (\alpha l_i^2 + \beta l_i + \gamma)$$

Where $\alpha = \frac{rc}{2}, \beta = R_b c + r C_b, \gamma = R_b C_b + t_b$, t_b is intrinsic repeater delay

Optimization of two-pin net

- Problem formulation:
  - minimize $t(\vec{l}) = \sum_{i=0}^{k}(\alpha l_i^2 + \beta l_i + \gamma)$

subject to

  - $g(\vec{l}) = l - \sum_{i=0}^{k} l_i = 0$ (1)

- Necessary condition (Kuhn-Tucker condition) for optimal solution: $\vec{\nabla} t(\vec{l}) + \lambda \vec{\nabla} g(\vec{l}) = 0$
  - Where λ is the Lagrangian multiplier

- $l_i = \frac{\lambda - \beta}{2\alpha}, i = 0, \dots, k$ (2)
- λ, β, α are constants, hence repeaters need to be equally spaced in order to minimize the delay
- The value of λ can be found by plugging (2) to (1)

Optimization of two-pin net

- Optimum number of repeaters in general case:

$$k = \left\lceil -\frac{1}{2} + \sqrt{1 + \frac{2(rcl + r(C_b - C_l) - c(R_b - R_d))^2}{rc(R_b C_b + t_b)}} \right\rceil$$

- Optimal length of segments

$$l_0 = \frac{1}{k+1}\left(l + \frac{k(R_b - R_d)}{r} + \frac{(C_L - C_b)}{c}\right)$$

$$l_1 = \cdots = l_{k-1} = \frac{1}{k+1}\left(l - \frac{(R_b - R_d)}{r} + \frac{(C_L - C_b)}{c}\right)$$

$$l_k = \frac{1}{k+1}\left(l - \frac{(R_b - R_d)}{r} - \frac{k(C_L - C_b)}{c}\right)$$

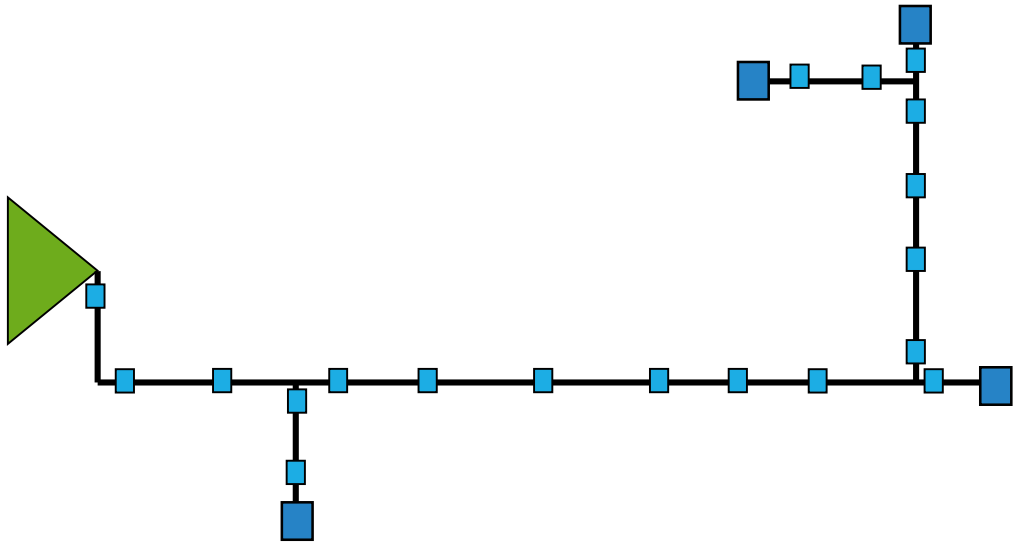
C. J. Alpert and A. Devgan. Wire segmenting for improved buffer insertion. In Proceedings of the ACM/IEEE Design Automation Conference, pages 588–593, 1997.

Van Ginneken's algorithm

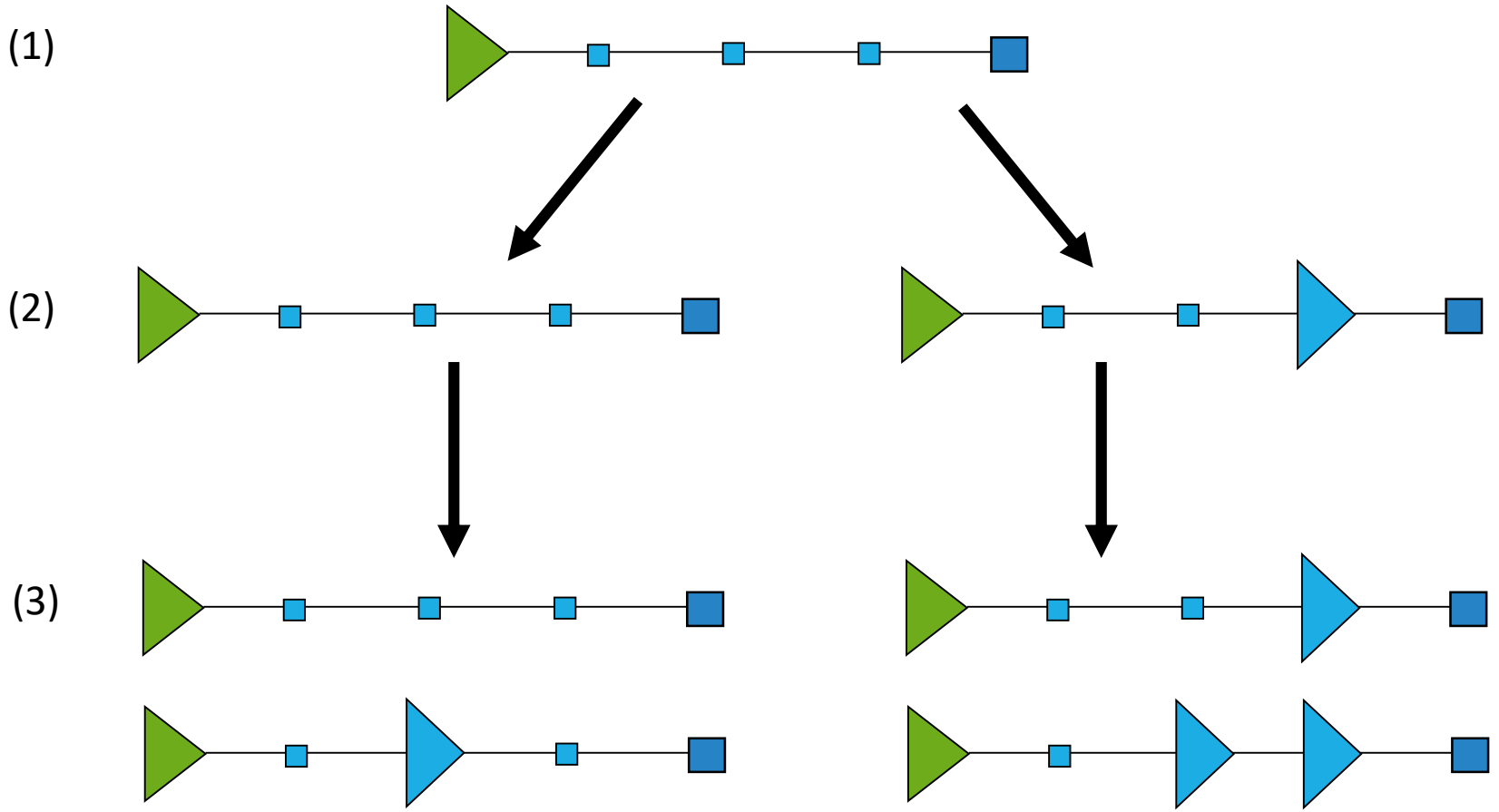
- Famous algorithm for optimal repeater insertion to a multi-sink routing tree
- Multi-sink net routing tree is fixed and candidate buffer locations are given
- Elmore delay model
- The algorithm can find optimal buffering solution that maximizes slack at source pin
- If there are n candidate buffer locations, the computational complexity is $O(n^2)$

Van Ginneken's algorithm

- Bottom-up from sinks toward the driver
- Generate list of candidate solution at each node
- Three operations may be performed at a node: adding wire, inserting buffer, branch merging
- Inferior solutions are pruned
- At source, pick the best candidate

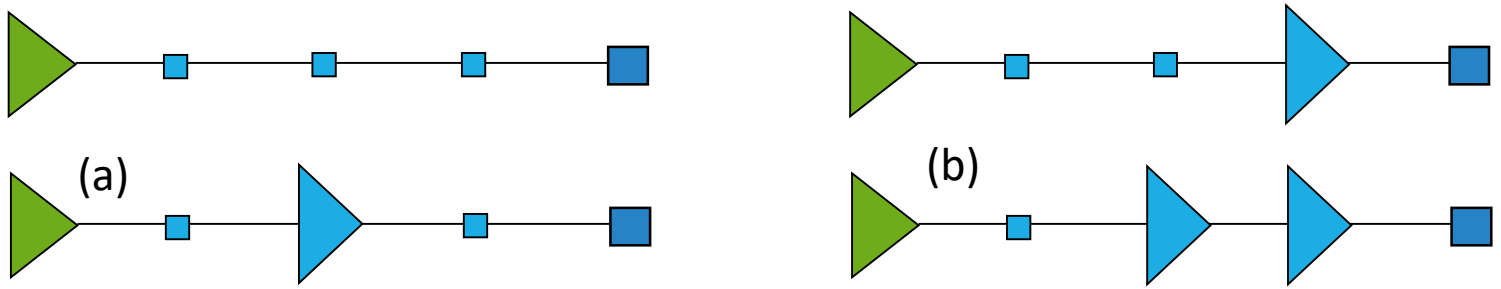


Generating Candidates



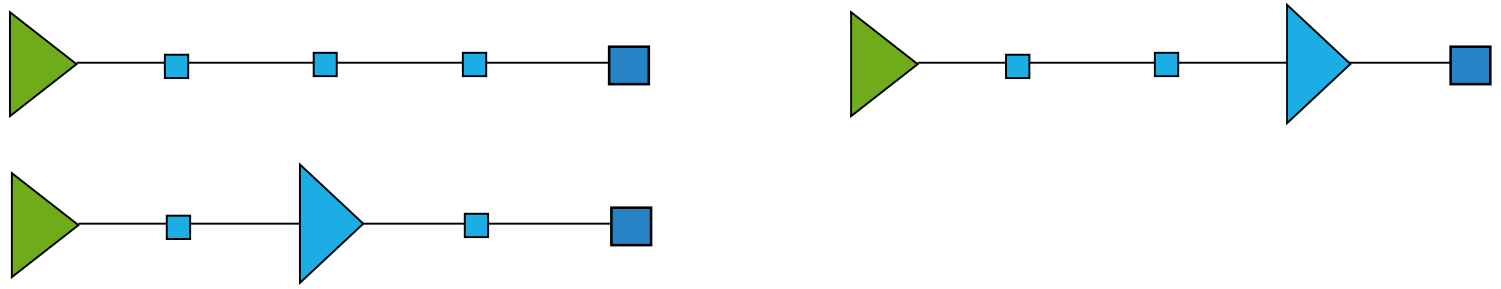
Pruning Candidates

(3)

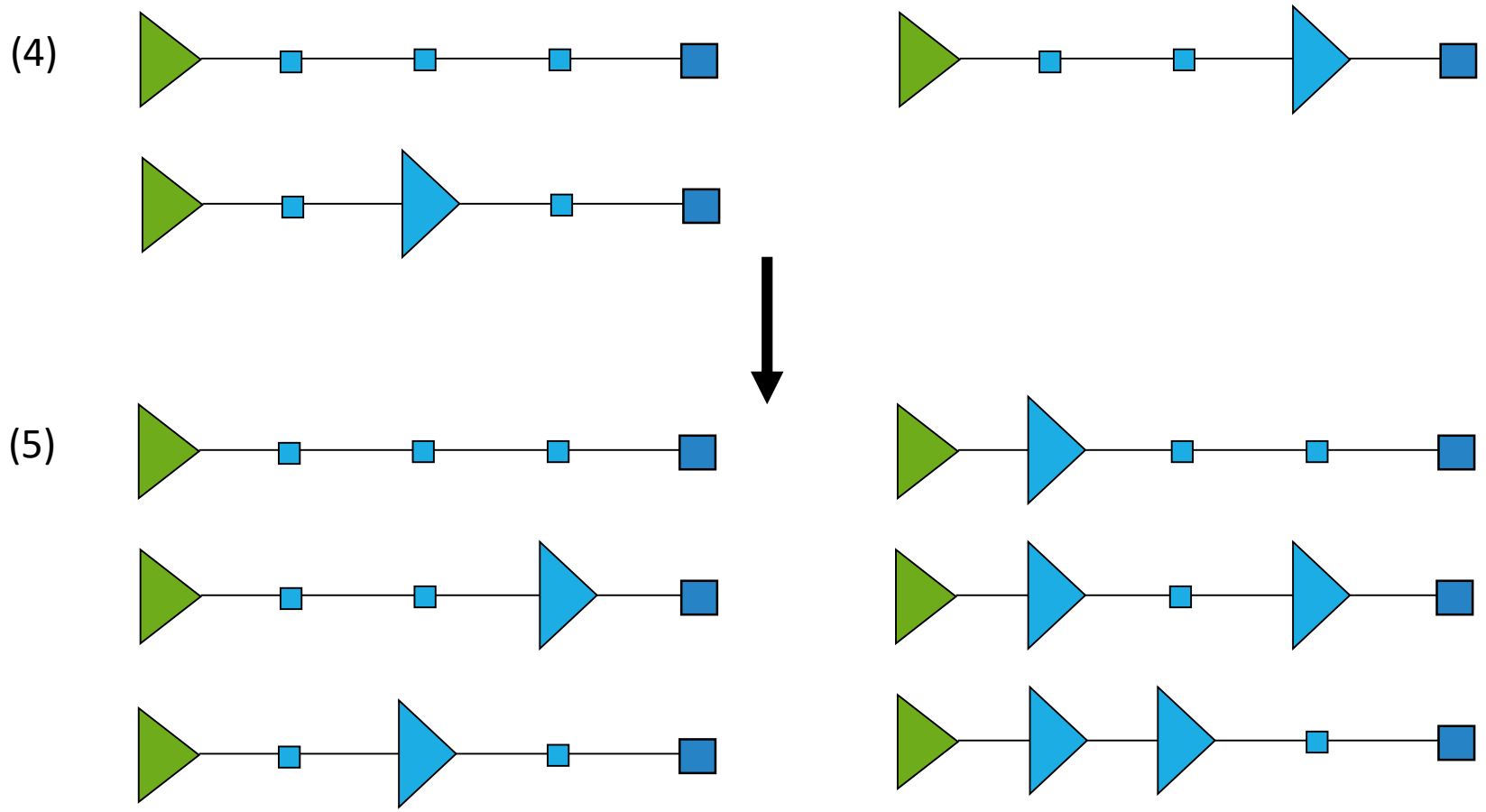


Both (a) and (b) “look” the same to the source.
Throw out the one with the worst slack

(4)



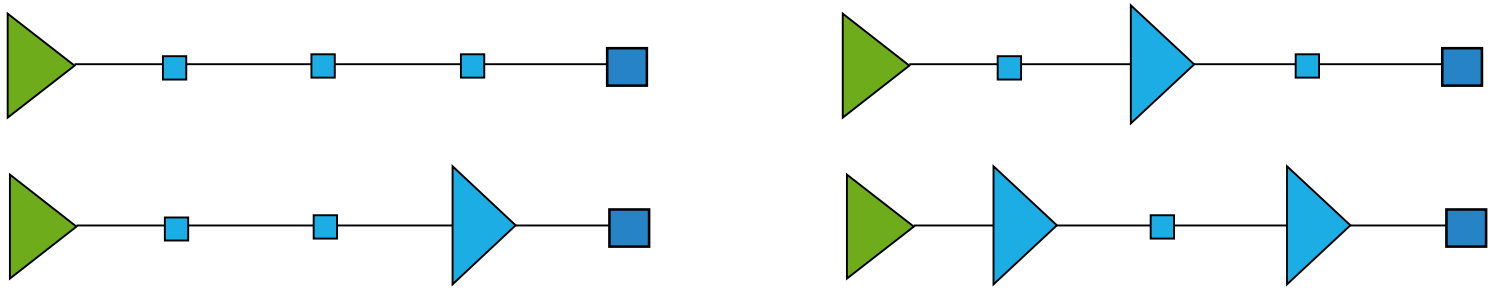
Candidate Example Continued



Candidate Example Continued

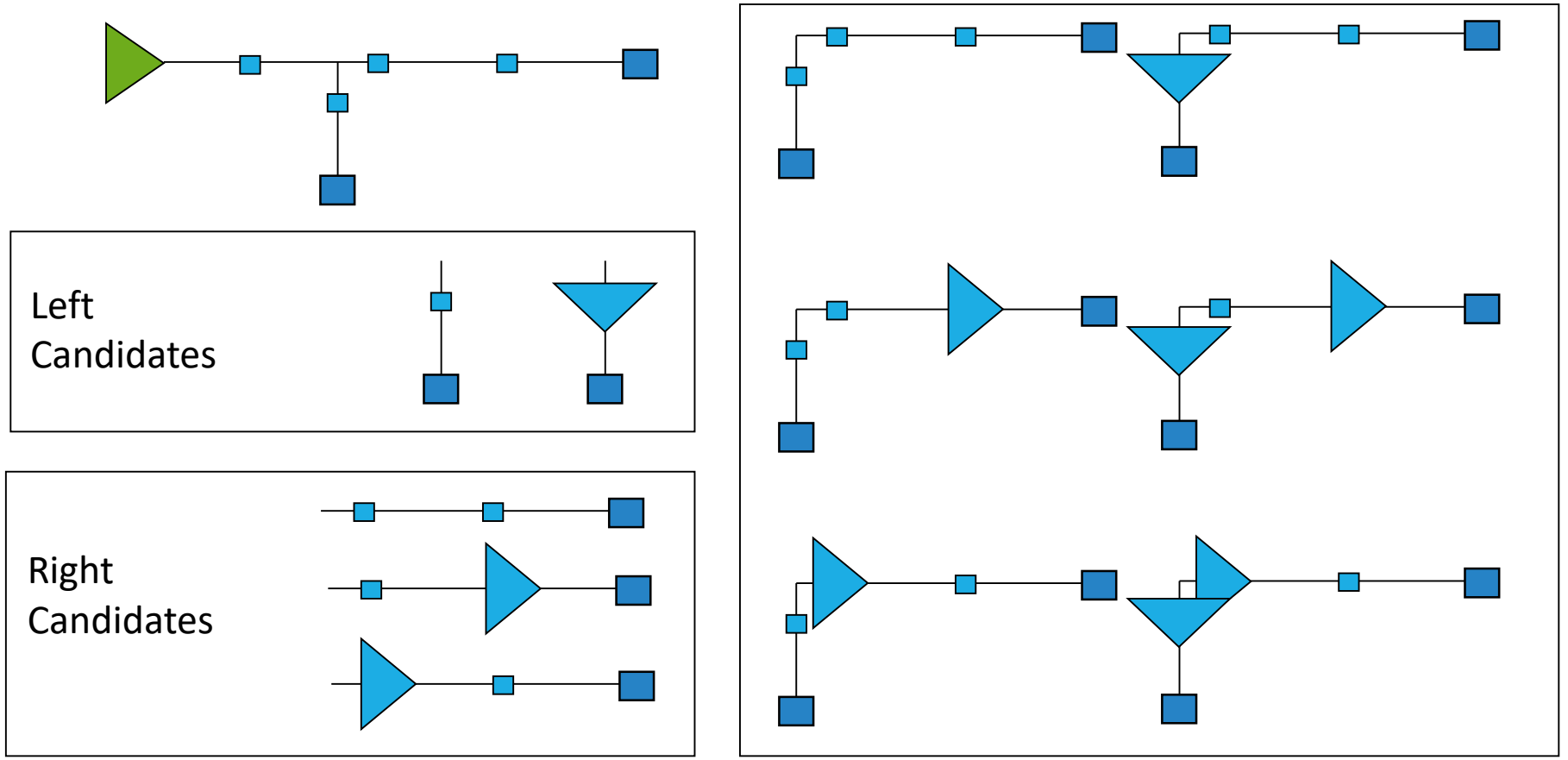
After pruning

(5)

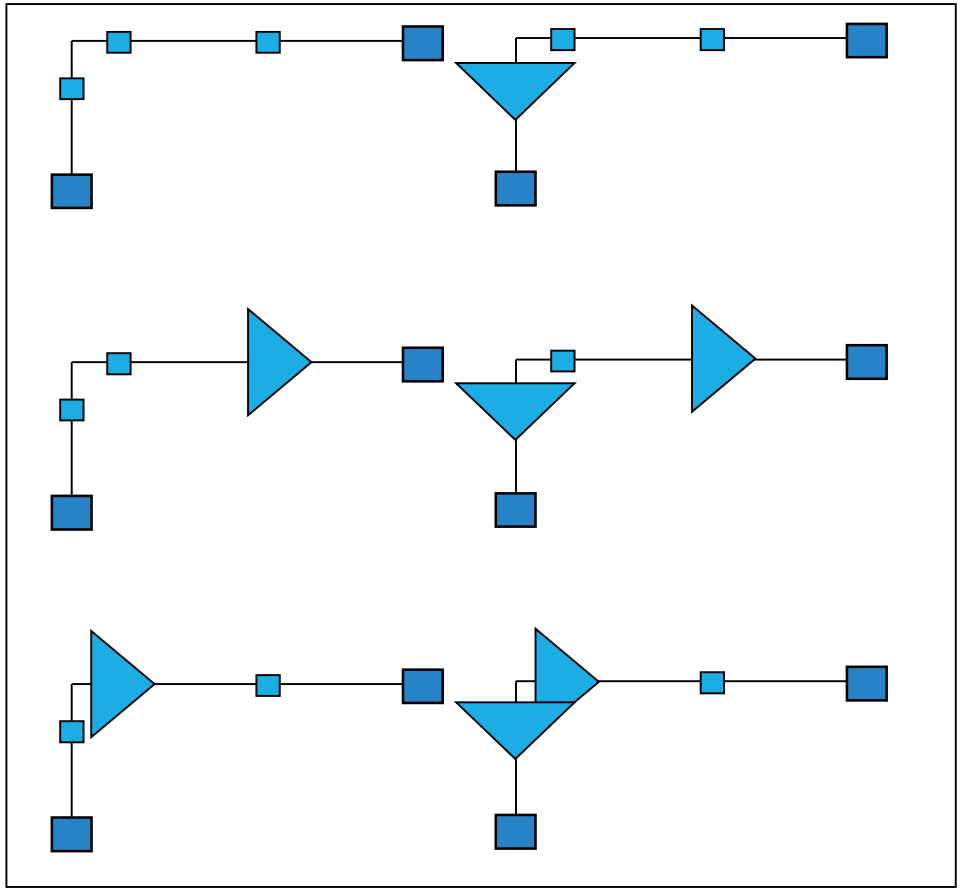


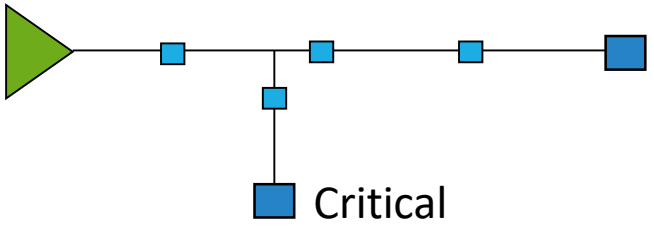
At driver, compute which candidate maximizes slack. Result is optimal.

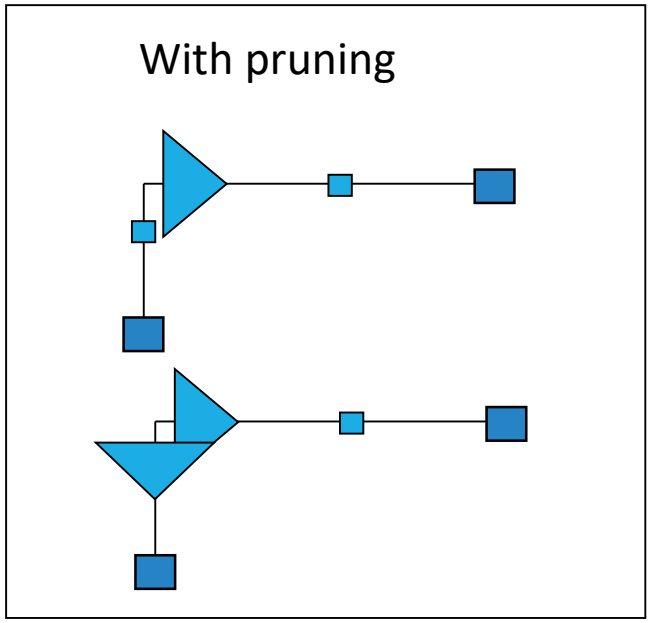
Merging Branches



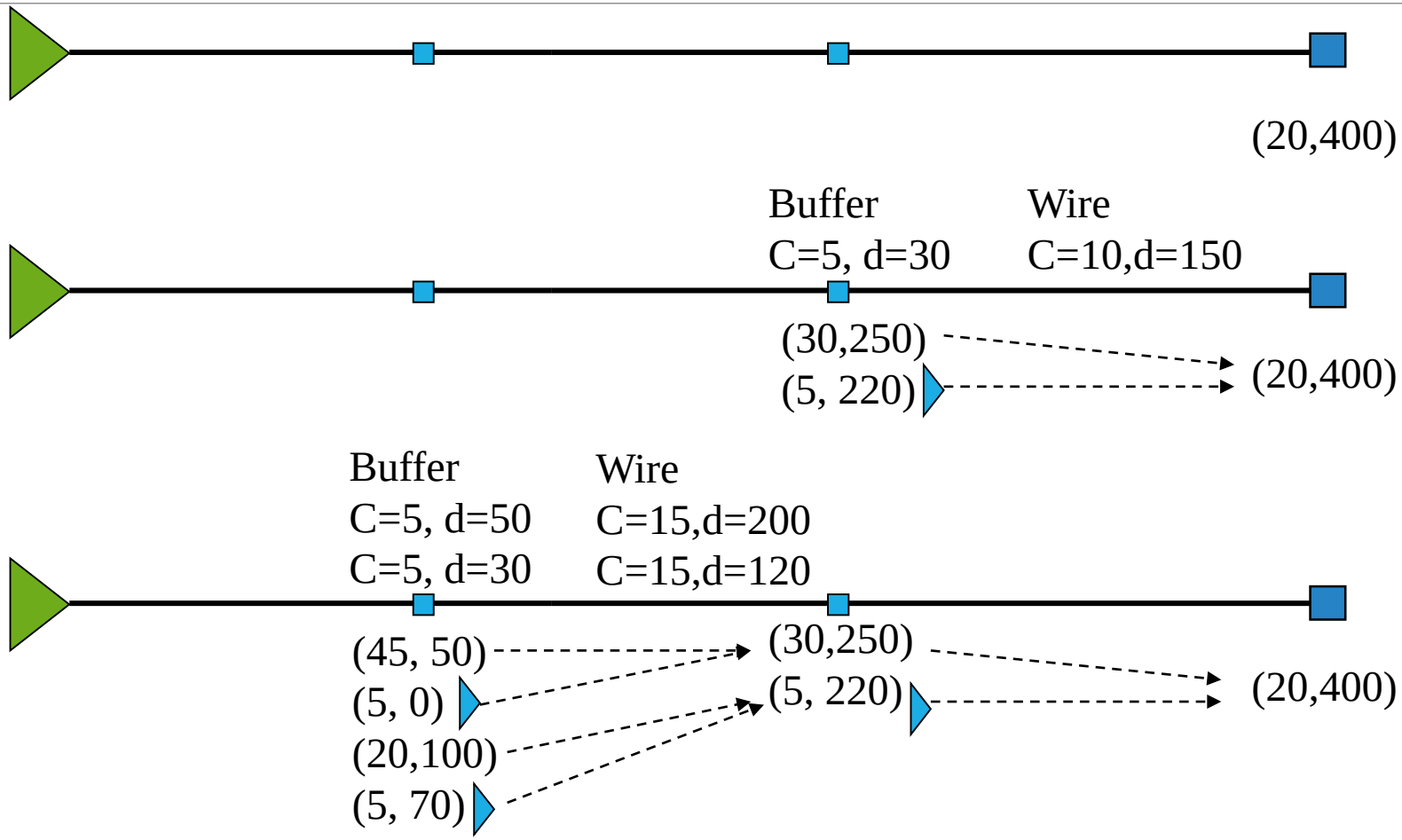
Pruning Merged Branches



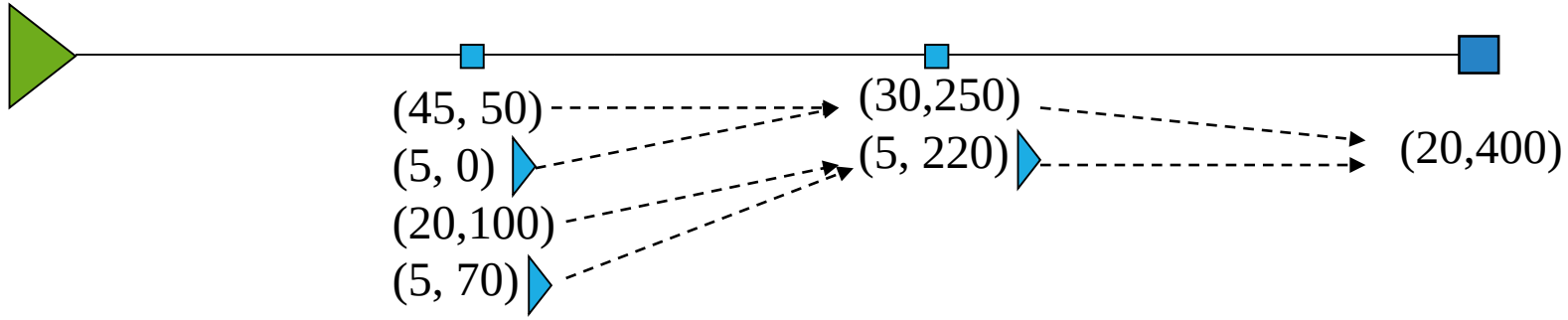




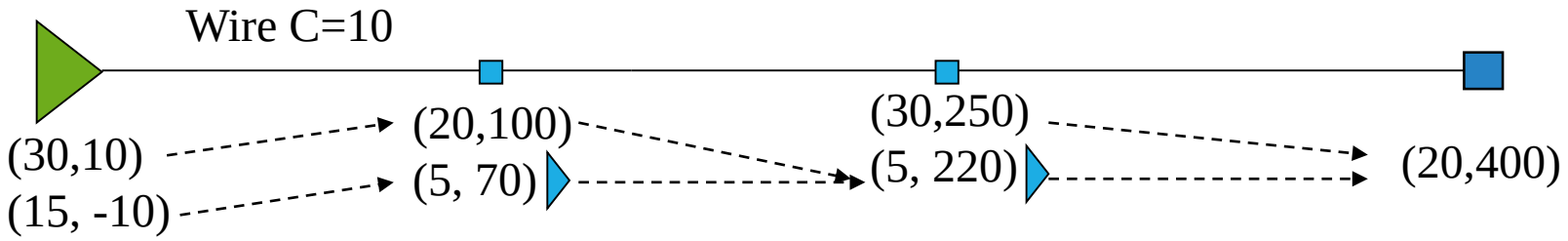
Van Ginneken Example



Van Ginneken Example Cont'd



(5,0) is inferior to (5,70). (45,50) is inferior to (20,100)



Pick solution with largest slack, follow arrows to get solution

Algorithm: van Ginneken's algorithm.

Input: T: routing tree, B: buffer library

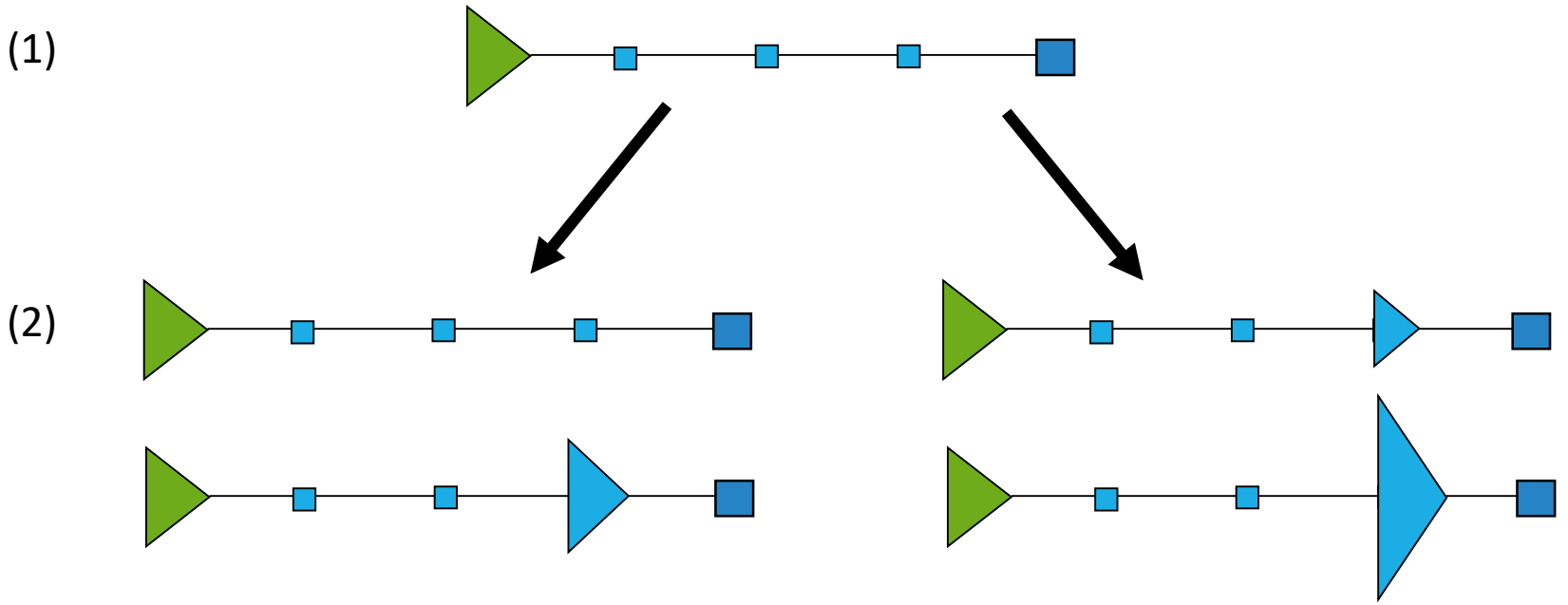
Output: γ which maximizes slack at driver

1. for each sink s, build a solution set $\{\gamma_s\}$, where $Q(\gamma_s) = RAT(s)$ and $C(\gamma_s) = C(s)$
2. for each branching point/driver v_t in the order given by a postorder traversal of T, let T' be each of the branches T_1, T_2 of v_t and Γ' be the solution set corresponding to T', do
3. for each wire e in T', in a bottom-up order, do
4. for each $\gamma \in \Gamma'$, do
5. $C(\gamma) = C(\gamma) + C(e)$
6. $Q(\gamma) = Q(\gamma) - D(e)$
7. prune inferior solutions in Γ'
8. if the current position allows buffer insertion, then
9. for each $\gamma \in \Gamma'$, generate a new solution γ'
10. set $C(\gamma') = C(b)$
11. set $Q(\gamma') = Q(\gamma) - R(b) \cdot C(\gamma) - K(b)$
12. $\Gamma' = \Gamma' \bigcup \{\gamma'\}$ and prune inferior solutions
13. // merge Γ_1 and Γ_2 to Γ_{v_t}
14. set $\Gamma_{v_t} = \emptyset$
15. for each $\gamma_1 \in \Gamma_1$ and $\gamma_2 \in \Gamma_2$, generate a new solution γ'
16. set $C(\gamma') = C(\gamma_1) + C(\gamma_2)$
17. set $Q(\gamma') = \min\{Q(\gamma_1), Q(\gamma_2)\}$
18. $\Gamma_{v_t} = \Gamma_{v_t} \bigcup \{\gamma'\}$ and prune inferior solutions
19. return γ with the largest slack

Van Ginneken Extensions

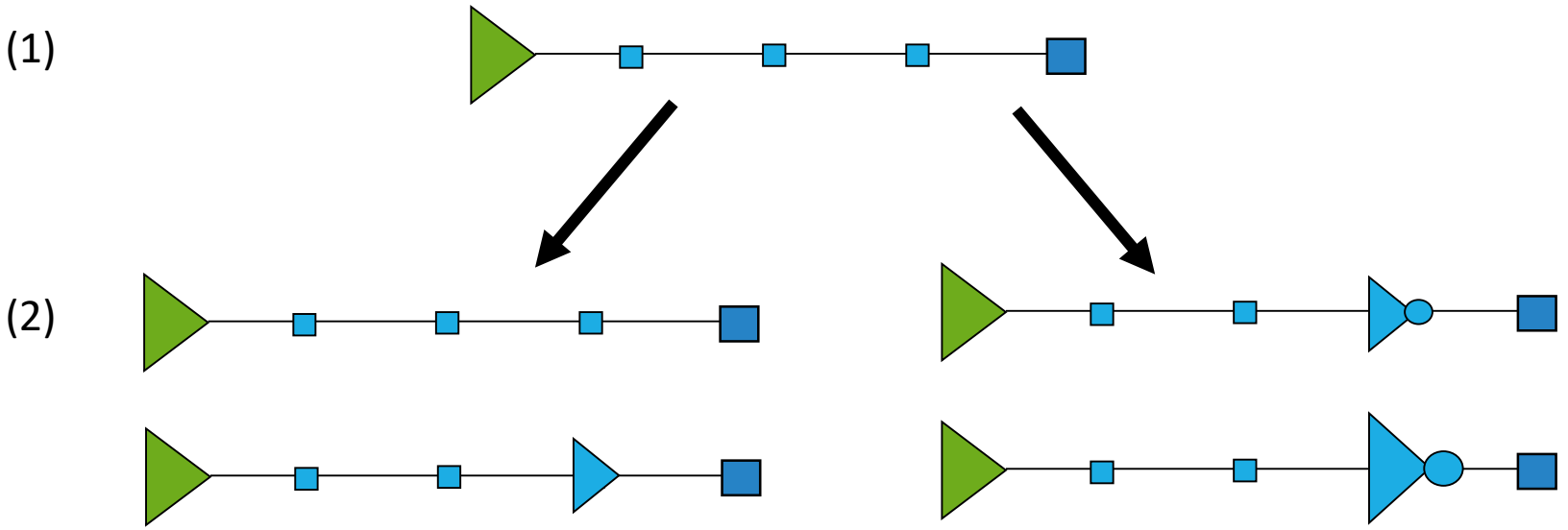
- Multiple buffer types
- Inverters
- Polarity constraints
- Controlling buffer resources
- Capacitance constraints
- Slew constraints
- Blockage recognition

Multiple Buffer Types



Time complexity increases from $O(n^2)$ to $O(n^2B^2)$
where B is the number of different buffer types

Inverters

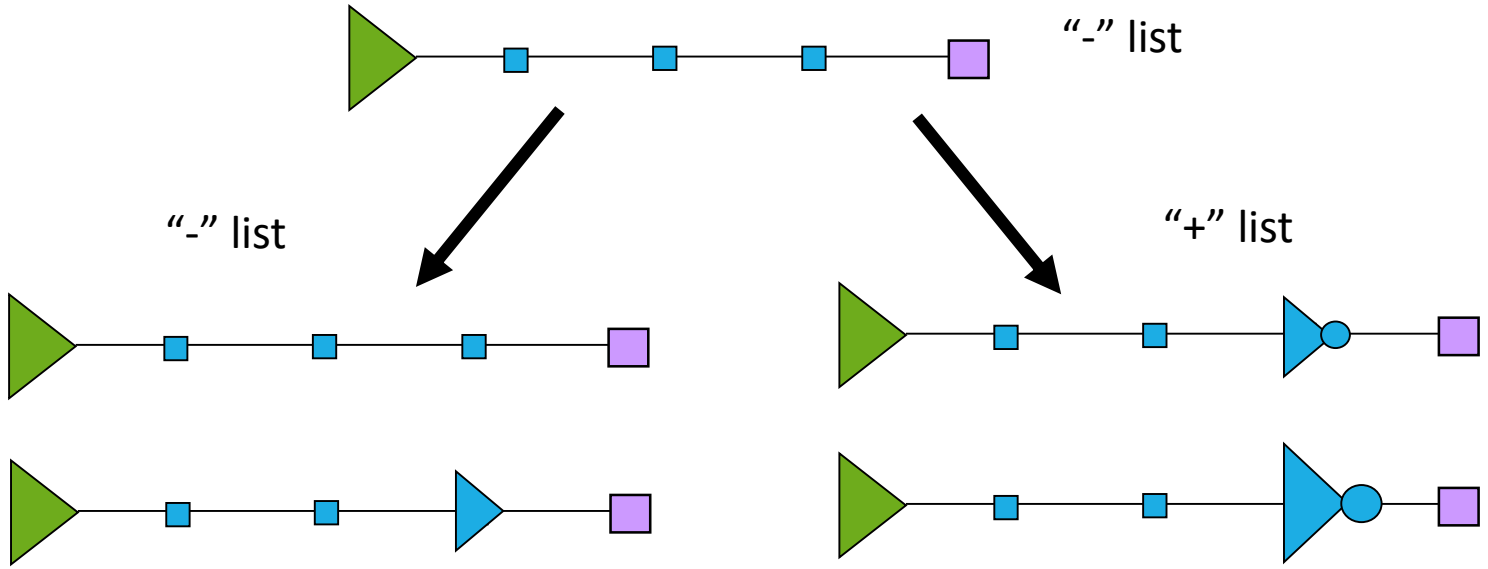


- Maintain a “+” and a “-” list of candidates
- Only merge branches with same polarity
- Throw out negative candidates at source

Polarity Constraints

Some sinks are positive, some negative

Put negative sinks into “-” list



Controlling Buffering Resources

Before, maintain list of capacitance slack pairs

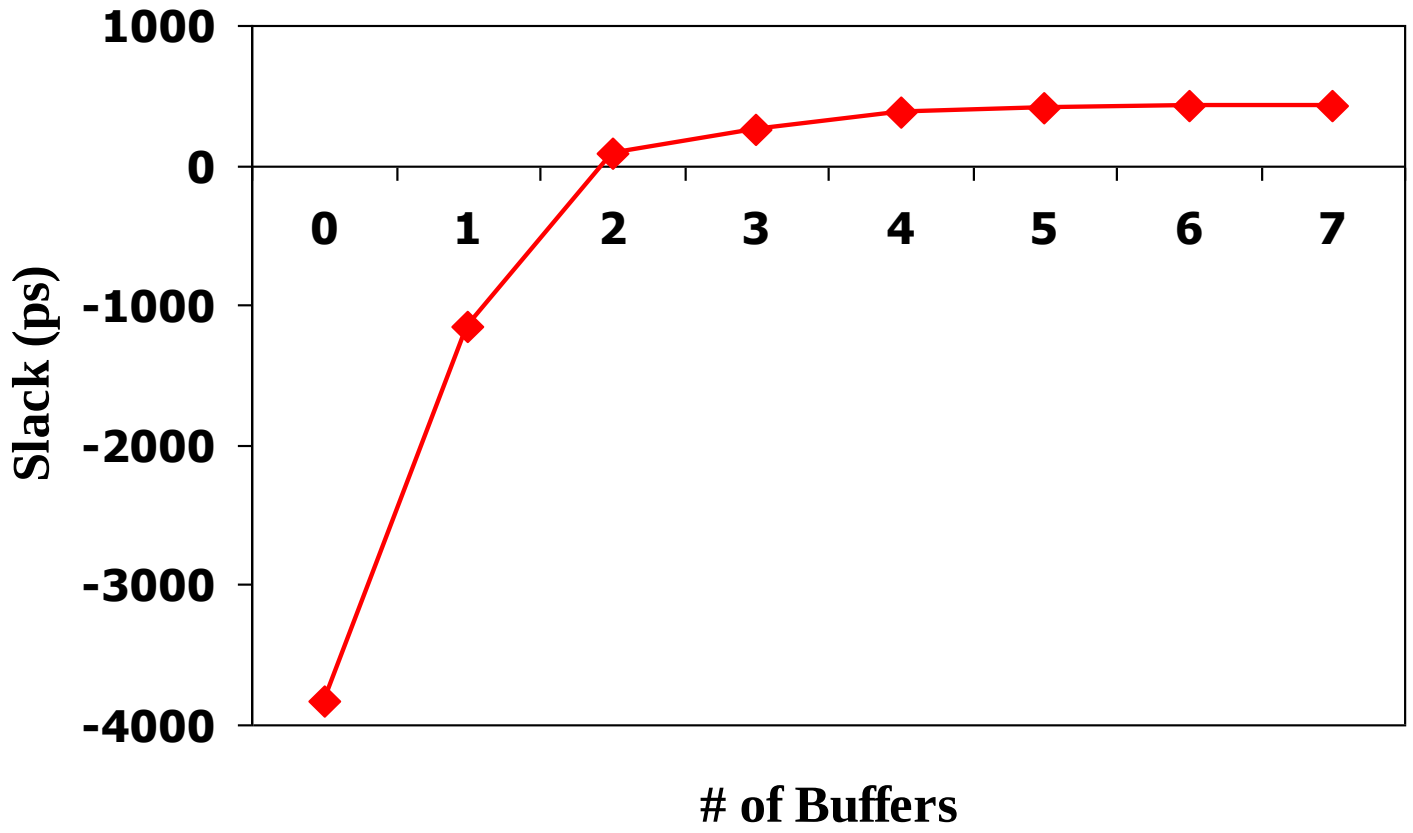
$(C_1, q_1), (C_2, q_2), (C_3, q_3)\ (C_4, q_4), (C_5, q_5)$
$(C_6, q_6), (C_7, q_7), (C_8, q_8)\ (C_9, q_9)$

Now, store an array of lists, indexed by # of buffers

$3 \longrightarrow (C_1, q_1, 3), (C_2, q_2, 3), (C_3, q_3, 3)$
$2 \longrightarrow (C_4, q_4, 2), (C_5, q_5, 2)$
$1 \longrightarrow (C_6, q_6, 1), (C_7, q_7, 1), (C_8, q_8, 1)$
$0 \longrightarrow (C_9, q_9, 0)$

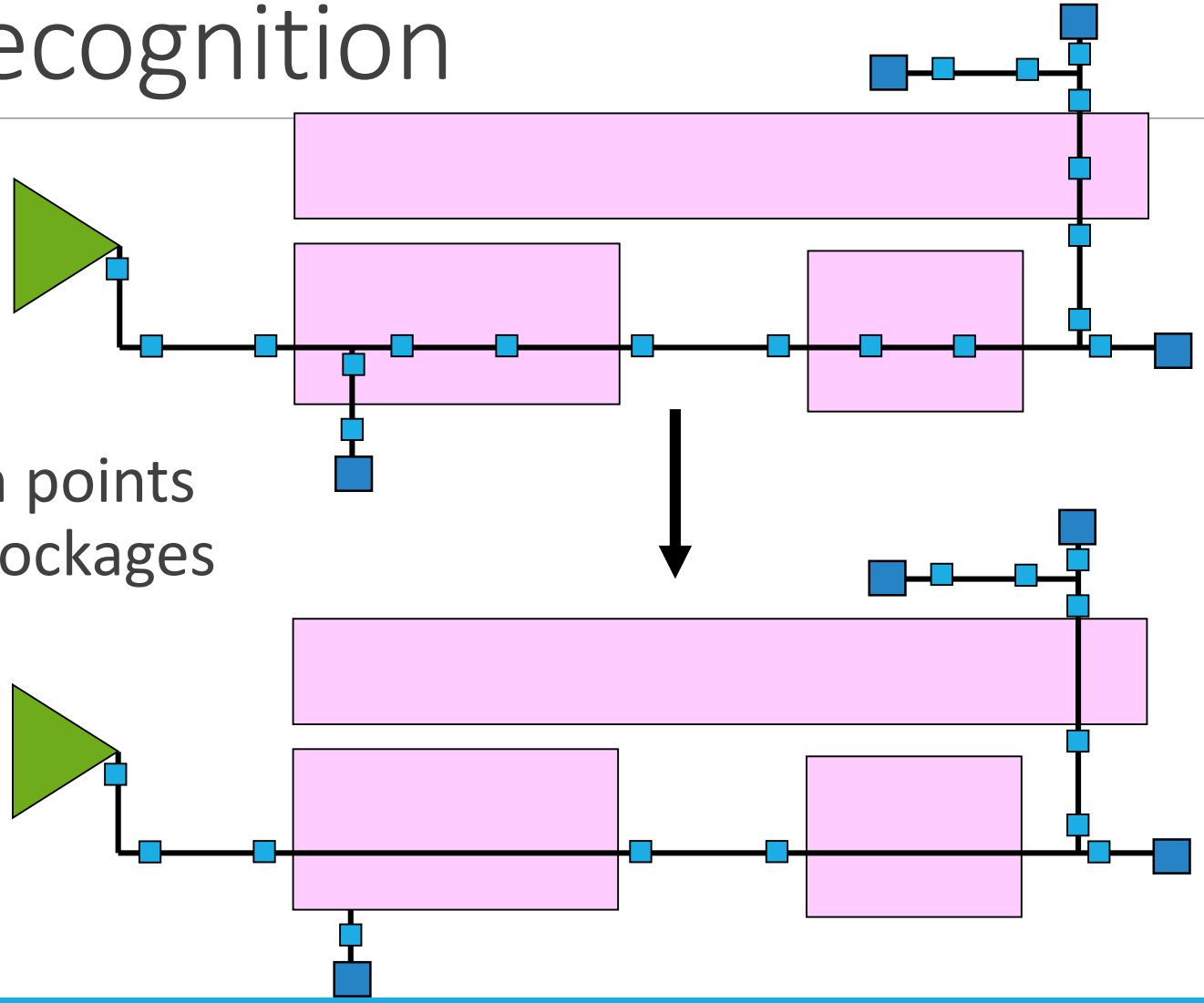
Prune candidates with inferior cap, slack, and #buffers

Buffering Resource Trade-off



Blockage Recognition

Delete insertion points
that run over blockages

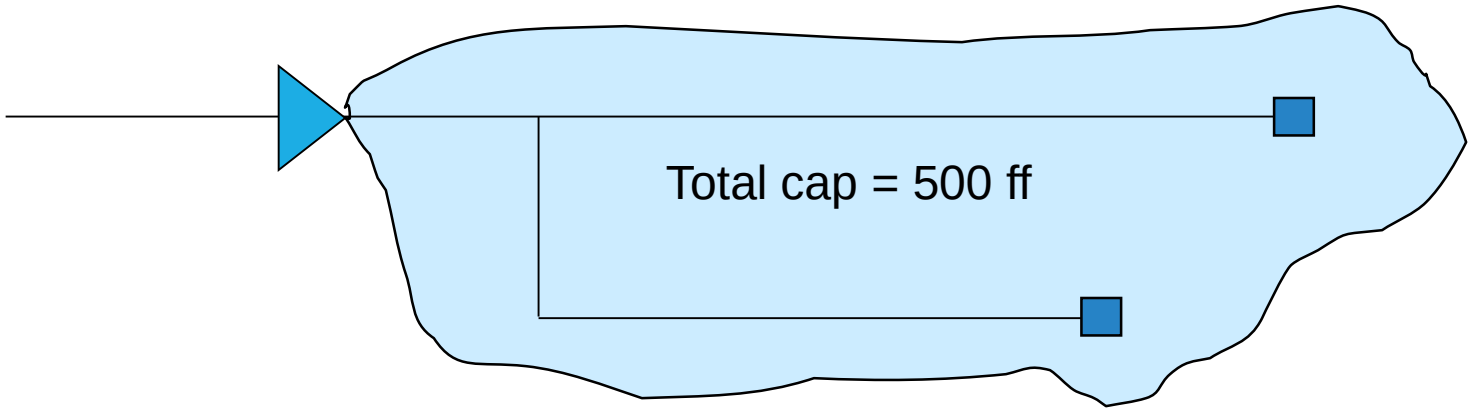


Capacitance Constraints

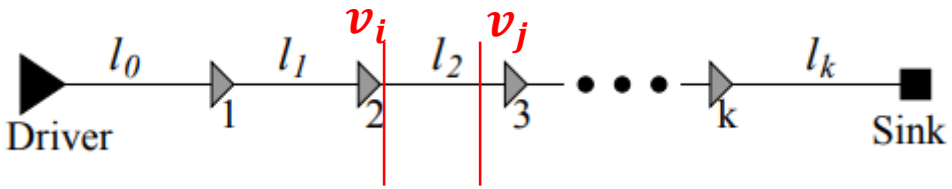
Each gate g drives at most C(g) capacitance

When inserting buffer g, check downstream capacitance.

If bigger than C(g), throw out candidate



Slew constraints



- Simple model for slew rate in a buffered wire:

$$S(v_j) = \sqrt{S_{bout}(v_i)^2 + S_w(p)^2}, \text{ where } S_w(p)^2 = \ln 9 \cdot D(p), D(p) \text{ is Elmore delay between } v_i \text{ and } v_j$$

- Slew rate at buffer output depends as slew rate at its input as:

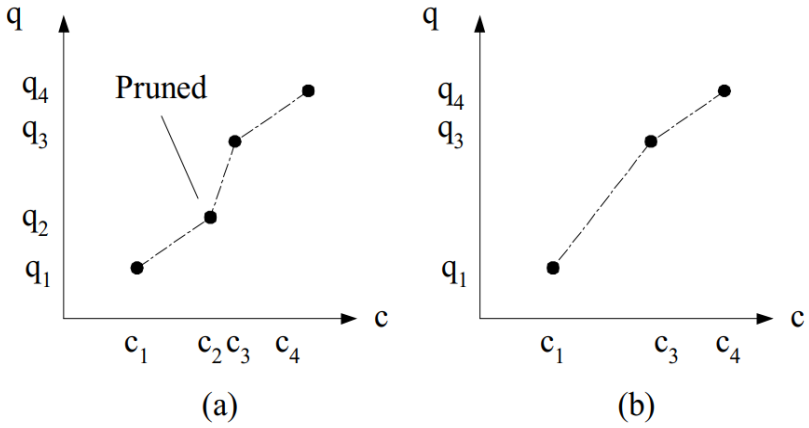
$$S_{bout} = b_0 + b_1 \cdot C(v) + b_2 S_{in}$$

- At input of each buffer one can assume a fixed input slew. This fixed slew is equal to the maximum slew constraint and therefore is always satisfied
- In a van Ginneken style buffering algorithm, if a candidate solution has a slew rate greater than given slew constraint, it is pruned out and will not be propagated any more

Predictive pruning

- The load seen at a particular node must be driven by some minimal amount of upstream wire or gate resistance
- By anticipating the upstream resistance R_{min} ahead of time, one can prune out more potentially inferior candidates earlier rather than later, which reduces the total number of candidates generated.
- Let α_1 and α_2 be two non-redundant candidates such that $C(\alpha_1) < C(\alpha_2)$ and $Q(\alpha_1) < Q(\alpha_2)$. If $Q(\alpha_2) - R_{min} \cdot C(\alpha_2) \leq Q(\alpha_1) - R_{min} \cdot C(\alpha_1)$, then α_2 is pruned
- Predictive pruning preserves optimality
- Using $R > R_{min}$ to prune candidates helps to achieve further speedup and may introduce a little degradation of solution quality

Convex pruning



- More potentially inferior candidates can be pruned out by comparing three neighboring candidate solutions simultaneously
- Let α_1, α_2 and α_3 be three non-redundant candidates such that $C(\alpha_1) < C(\alpha_3) < C(\alpha_3)$ and $Q(\alpha_1) < Q(\alpha_2) < Q(\alpha_3)$. If

$$\frac{Q(\alpha_2) - Q(\alpha_1)}{C(\alpha_2) - C(\alpha_1)} < \frac{Q(\alpha_3) - Q(\alpha_2)}{C(\alpha_3) - C(\alpha_2)}$$

then we call α_2 non-convex, and prune it.

- For 2-pin nets, convex pruning preserves optimality
- For multi-pin nets when the upstream could be a merging vertex, non-redundant candidates that are pruned by convex pruning could still be useful.

Repeater insertion as part of physical synthesis

- Many nets need repeater insertion in modern large synthesized blocks
- Timing criticality of sink terminals of a net (and thus, good routing tree topology) depends on the environment (synthesis results for other nets)
- Eventually, we need to optimize paths, not nets
- Repeater insertion during physical synthesis could be organized as follows:
  - Preliminary repeater insertion: fix slew violations trying to minimize total repeater area
  - Main repeater insertion phase
  - Incremental optimization of timing critical nets

Main phase: topology generation

- Consider all sink terminals as equally timing critical.
- Use Prim-Dijkstra algorithm for the performance-driven routing tree design. Cost function estimated while choosing a sink to add:
  - $cost(v_j) = \alpha \cdot d_{ij} + \beta \cdot r_j$, where d_{ij} is the length of edge e_{ij} and r_j is the distance between driver and v_j
- Minimize number of edges that connect nodes with different polarity
- Generate several candidates using different (α, β)

Main phase: repeater insertion

1. Initialization
   a) Store initial buffer tree as T_{best}
   b) Remember real $RAT(i)$ for each sink
   c) Drop existing net buffering, i.e. remove all buffers an inverters from the tree
   d) Set counter C to zero; set accumulated slack of each sink $AS(i)$ to zero; set $RAT(i)$ to zero
2. Repeater insertion
   a) Run van Ginnken algorithm to obtain a few buffering solutions $\{S\}$
   b) Increment counter C
3. Timing analysis
   a) Restore real real $RAT(i)$ for each sink
   b) Calculate accurate timing for each solution
4. Decide best tree from $\{S, T_{best}\}$ and store it as T_{best}; stop if $C = C_{max}$
5. Preparation for the next iteration
   a) Increment $AS(i)$ by value of its negative slack $NS(i)$ for tree T_{best}
   b) Set $RAT(i) = AS(i)/C$
   c) Return to stage 2

Refinement for timing critical nets

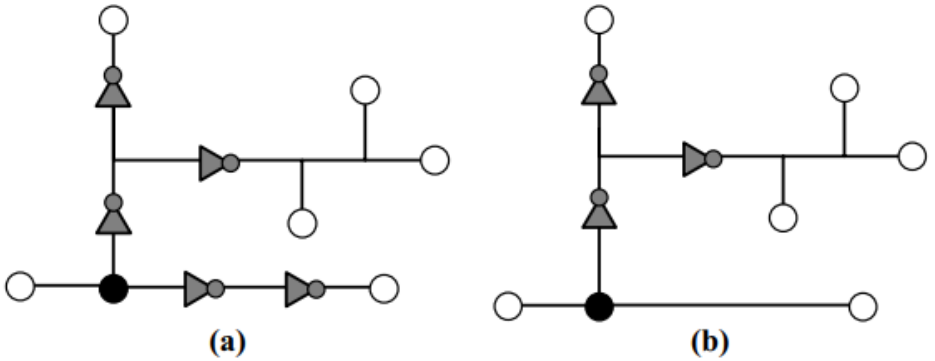


Figure 5. Removing a pair of inverters. The delay between the starting point and the end point of the branch may become larger, but the slack at driver can still be positive.

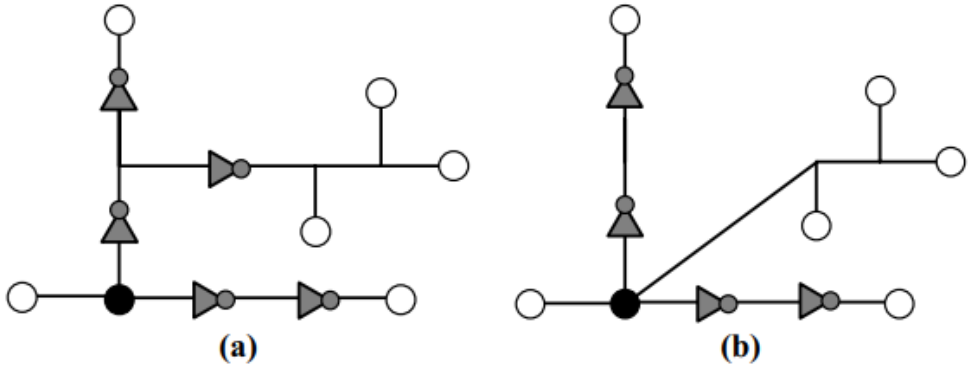


Figure 6. Removing a single repeater (inverter).

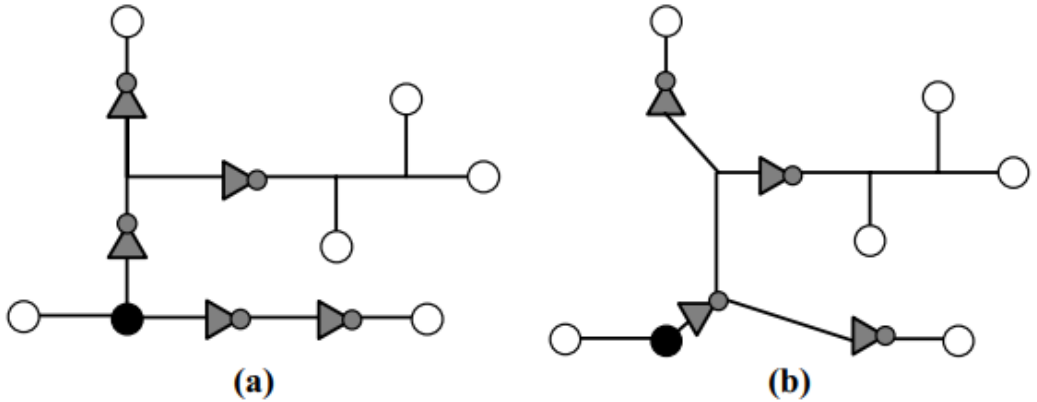


Figure 7. Merging of nearby repeaters of the same type.

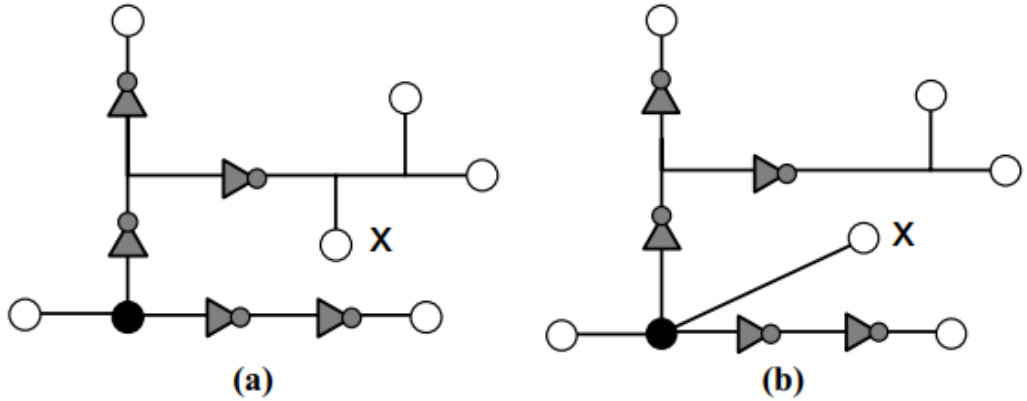


Figure 8. A critical sink X is reconnected closer to driver.

References

- Handbook of Algorithms for Physical Design Automation. Chapter 26, Buffer Insertion Basics by Jiang Hu, Zhuo Li, and Shiyan Hu
- Buffer and FF Insertion, Charles J. Alpert, IBM Corp. (Tutorial: Optimization Strategies for Physical Synthesis and Timing Closure ICCAD-2001)
- A Practical Repeater Insertion Flow, by N. Ryzhenko and O. Venger, GLSVLSI'08

Backup