Lecture 15: Buffering

Matthew Guthaus
Professor
UCSC, Computer Science & Engineering
http://vlsida.soe.ucsc.edu
mrg@ucsc.edu

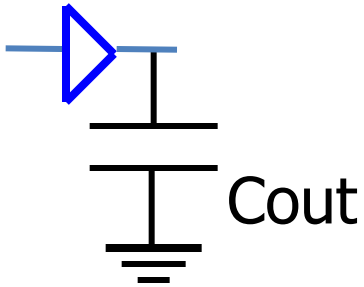
Today's Lecture

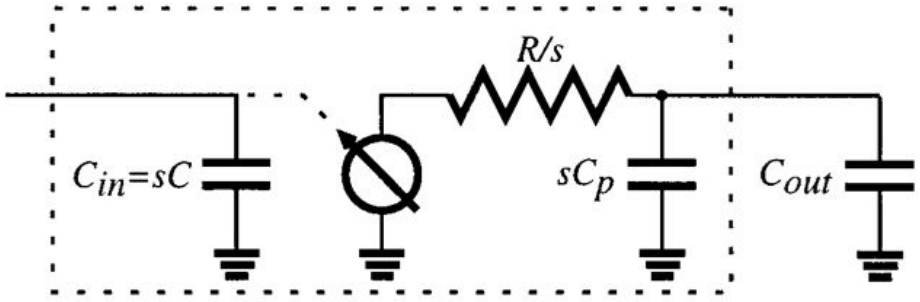
- Cascaded Buffers
  - Driving large capacitive loads
- Repeaters
  - Driving long wires
- Generalized Buffering
  - Capacitive shielding
  - Driving fanout with varying criticalities
  - Van Ginneken algorithm

Today's Lecture

- Cascaded Buffers
  - Driving large capacitive loads
- Repeaters
  - Driving long wires
- Generalized Buffering
  - Capacitive shielding
  - Driving fanout with varying criticalities
  - Van Ginneken algorithm

Linear Gate Delay Model

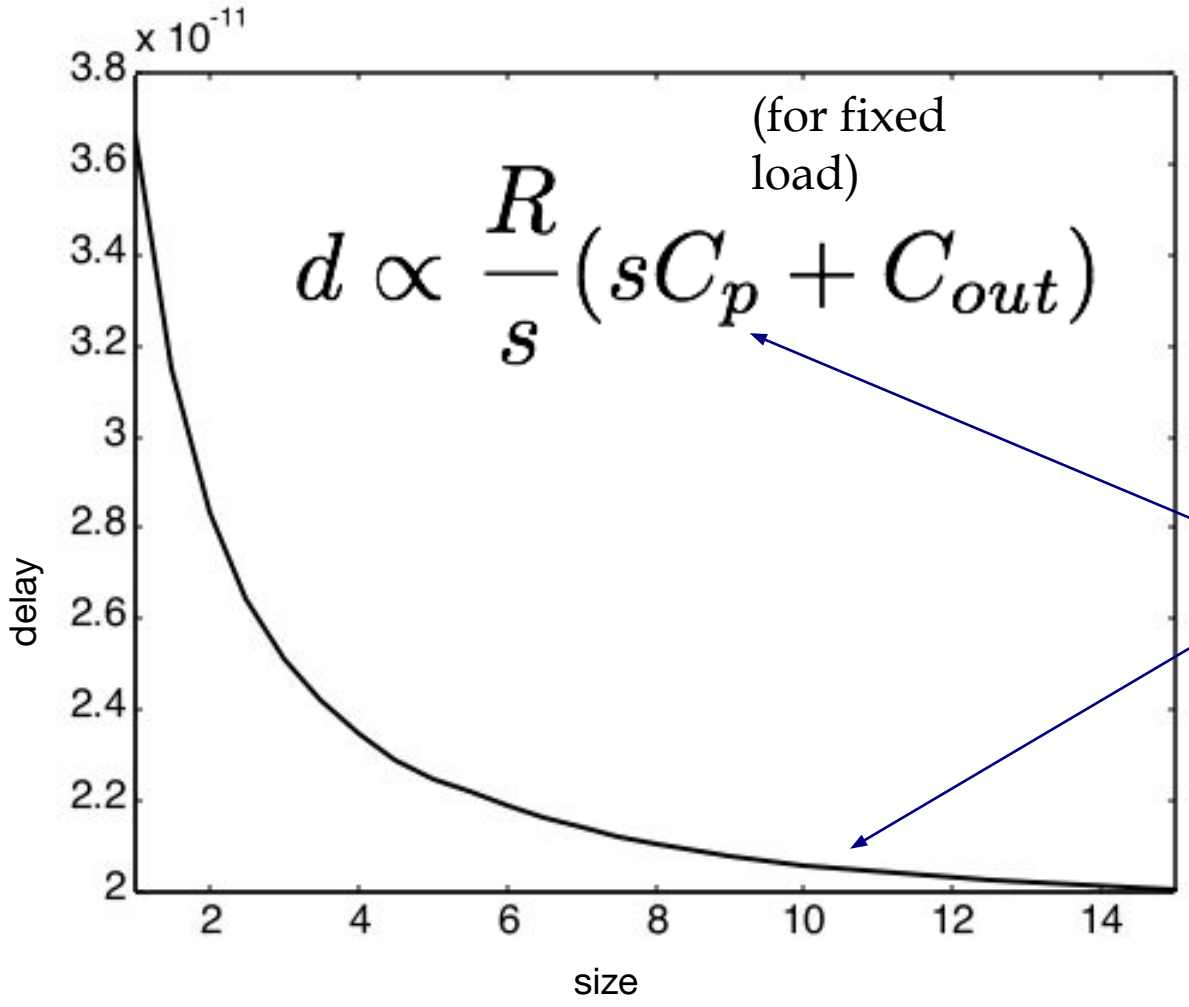




$$d \propto \frac{R}{s}(sC_p + C_{out})$$

- Sutherland, Ivan E.; Sproull, Robert F.; Harris, David F. (1999). *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann. ISBN 1-55860-557-6.

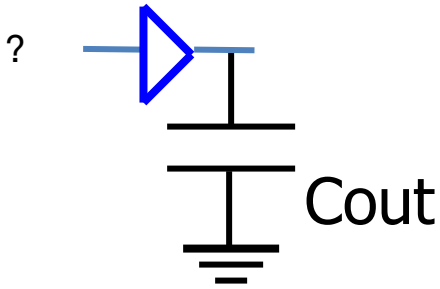
Self Loading: Another View

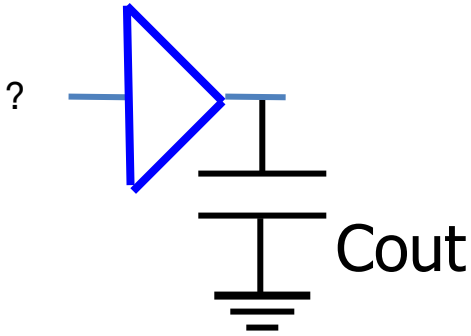


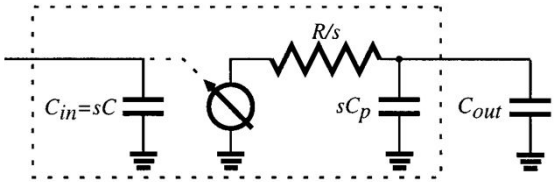
Self-loading effect: Intrinsic capacitance (C_p) dominates

Intuition Check

- Driving a big capacitor requires a big buffer/inverter
- A big buffer/inverter has a big input capacitance and will slow the previous stage
- What to do?

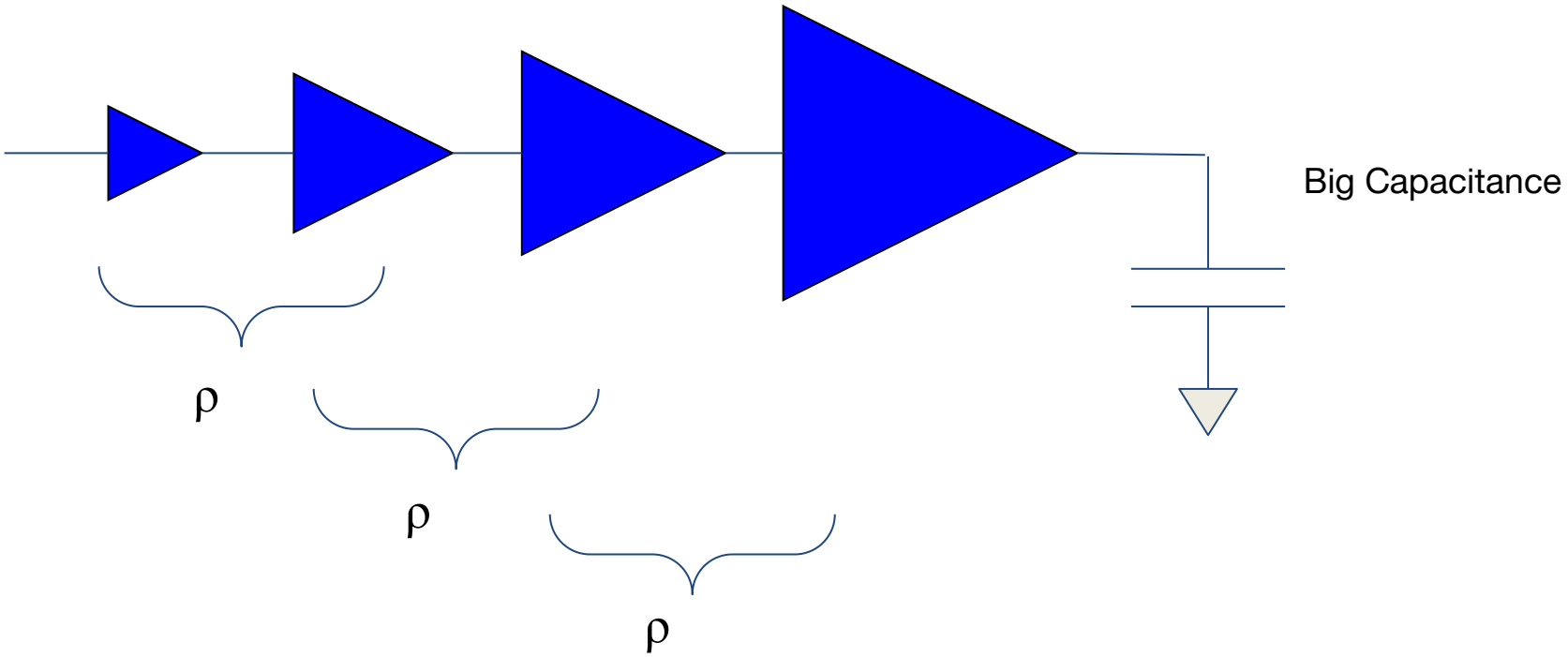






Multistage drivers

- Each stage “ramps up” to a bigger buffer (or inverter)
  - Small capacitance on first stage
  - Big buffer on last stage
- What is the ideal “ramp up”?



Best Ratio of Sizes

- Formulate equation with N stages with linear delay model and solve to minimize delay
- Has no closed-form solution

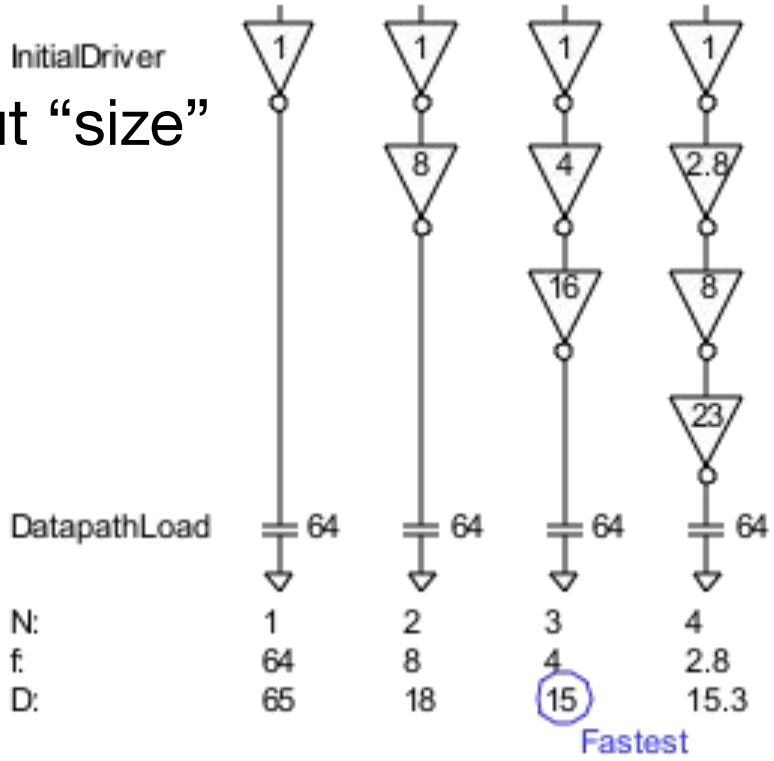
$$p_{inv} + \rho\left(1 - \ln\rho\right) = 0$$

- P_{inv} is the parasitic delay (RC_p)
- Neglecting parasitic delay ($P_{inv} = 0$)
  - $\rho = 2.718$ (e)
- For $P_{inv} = 1$, solve numerically
  - $\rho = 3.59$

- Sutherland, Ivan E.; Sproull, Robert F.; Harris, David F. (1999). *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann. ISBN 1-55860-557-6.

Best Number of Stages

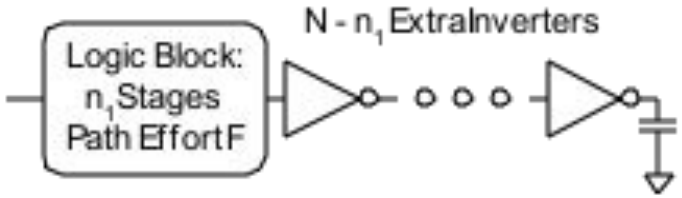
- How many stages should a driver use?
  - Minimizing number of stages is not always fastest
  - Each stage should “ramp up” equally
- Example: drive 64-bit datapath with unit inverter
- Brute force:
  - N is number of stages
  - f is ratio of output to input “size”
  - D is delay

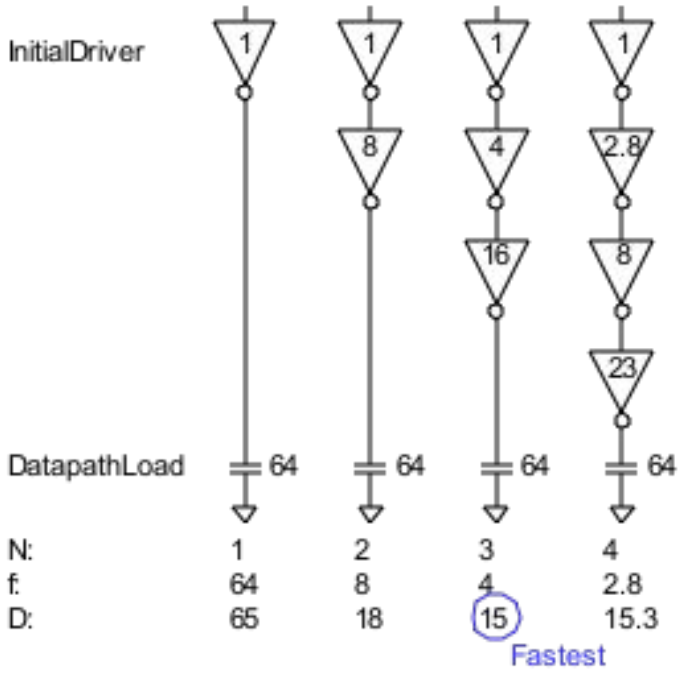


Best Number of Stages

- Equally divide the ratio of sizes
  - $\log_{3.59}(64) \cong \log_4(64) = 3$
- How to compute arbitrary logs:
  - $\log_4(64) = \log(64)/\log(4)$

$$\frac{\partial D}{\partial N} = 0$$



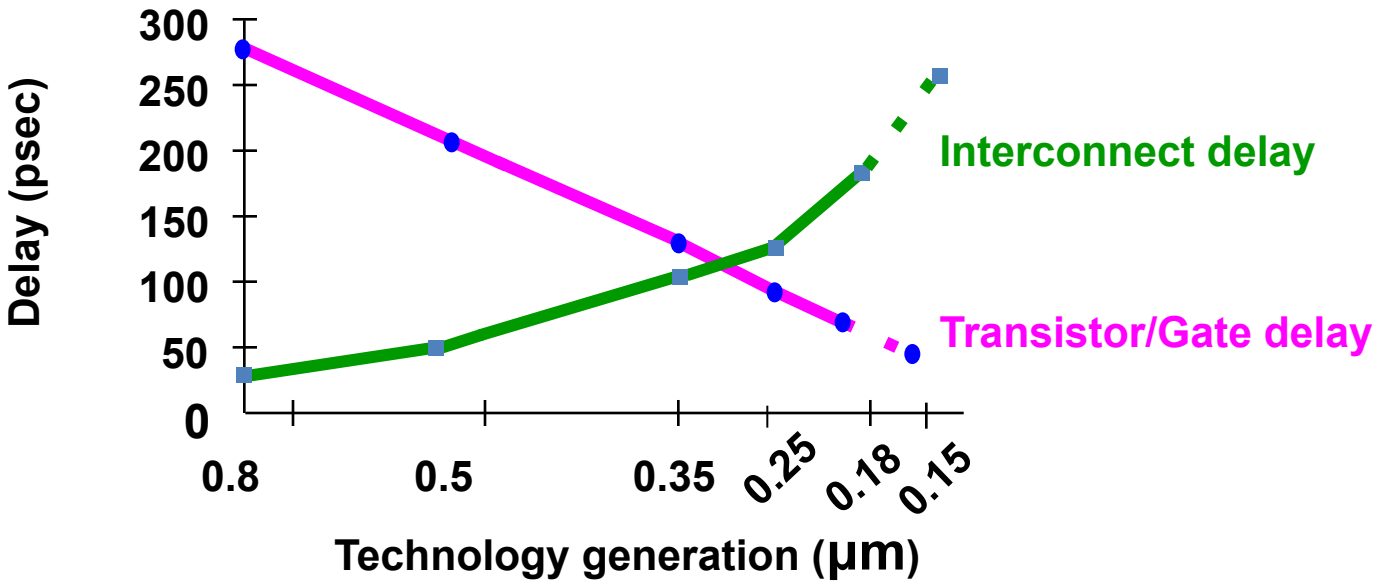


- Sutherland, Ivan E.; Sproull, Robert F.; Harris, David F. (1999). *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann. ISBN 1-55860-557-6.

Today's Lecture

- Cascaded Buffers
  - Driving large capacitive loads
- Repeaters
  - Driving long wires
- Generalized Buffering
  - Capacitive shielding
  - Driving fanout with varying criticalities
  - Van Ginneken algorithm

Why is this trend?

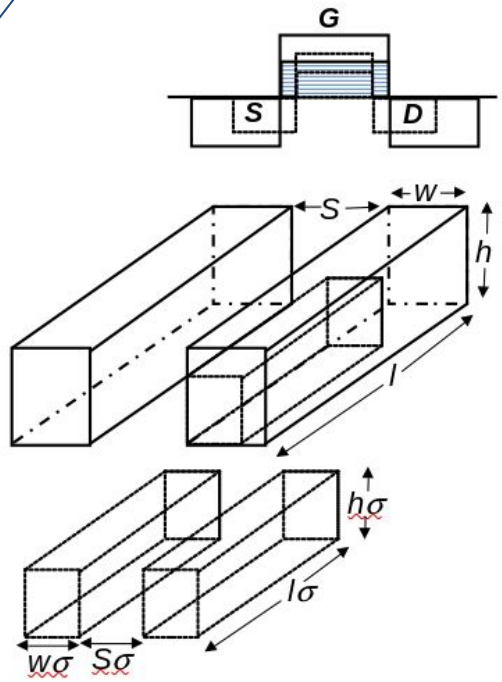


Source: Gordon Moore, Chairman Emeritus, Intel Corp.

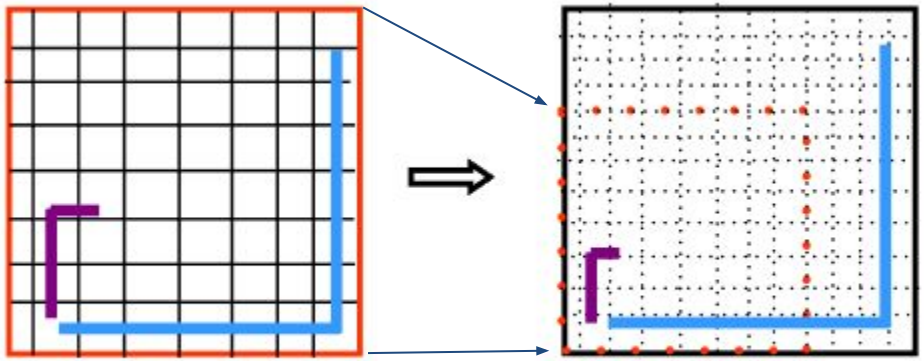
Dennard Scaling

- **Ideal process scaling:**
  - Device geometries shrink by s *(= 0.7x)*
    - *"Moore's Law"*
    - Device delay shrinks by s
  - Wire geometries shrink by s (or σ)
    - ***Unit resistance R*** **:** $\rho/(w_s.h_s) = r/s^2$
    - ***Unit coupling capacitance***
      Cc **:** $(h_s)/(S_s)$
    - **Resistance doubled, capacitance roughly unchanged for unit length**
    - How about the change in wire length?

$s^2 = 0.7^2 = 0.49$



Wire length scaling



- Global (long) interconnect lengths don't shrink
  - Global interconnect link cells far apart
  - Maximum chip size stays roughly the same (more transistors though!)
- Local (short) interconnect lengths shrink by s
  - Local interconnects link cells nearby

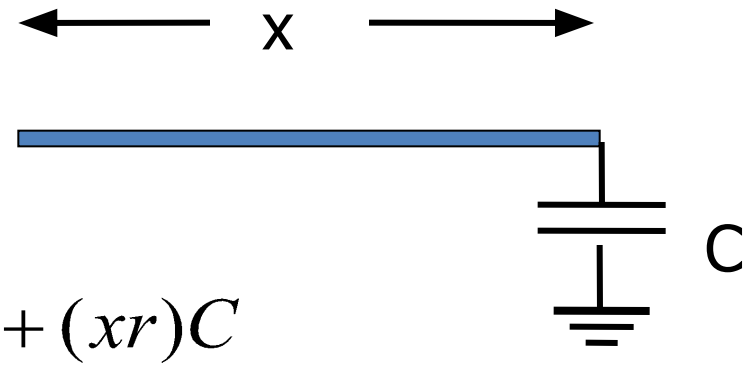
Interconnect delay scaling

- **Delay of a wire of length *l* :**
  $\tau_{int} = (rl)(cl) = rcl^2$ **(a quadratic function of length)**
- **Local interconnects :**
  $\tau_{int} : (r/s^2)(c)(ls)^2 = rcl^2$
  - **Local interconnect delay unchanged (s^2 cancels out)**
- **Global interconnects :**
  $\tau_{int} : (r/s^2)(c)(l)^2 = (rcl^2)/s^2$
  - **Global interconnect delay doubled – unsustainable!**
- ***Interconnect delay increasingly more dominant***

Elmore Delay for Wire

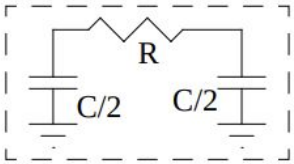
unit wire capacitance c

unit wire resistance r

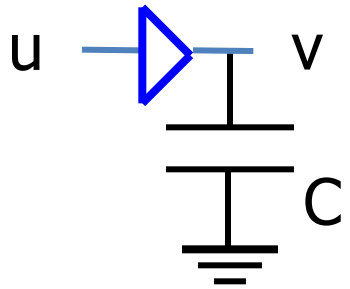


$$delay = \frac{(xr)(xc)}{2} + (xr)C$$

Above assumes a “pi” model for the wire



Elmore Delay for Buffer



$$delay(u,v) = R(b)C$$

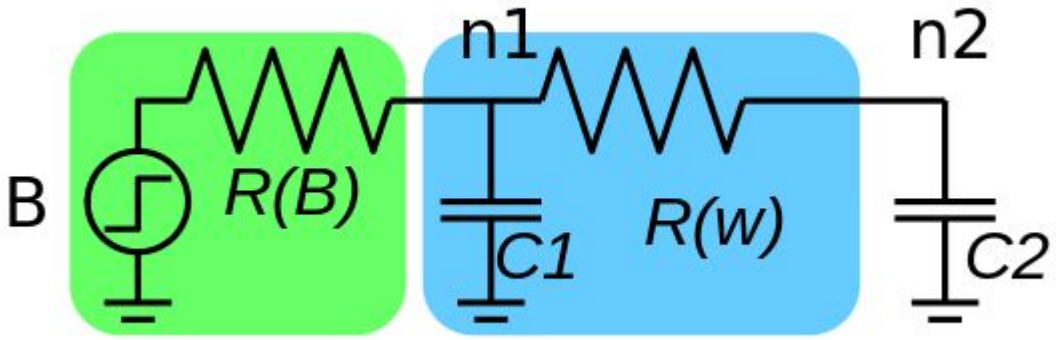
$$C(u) = C(b)$$

Input capacitance

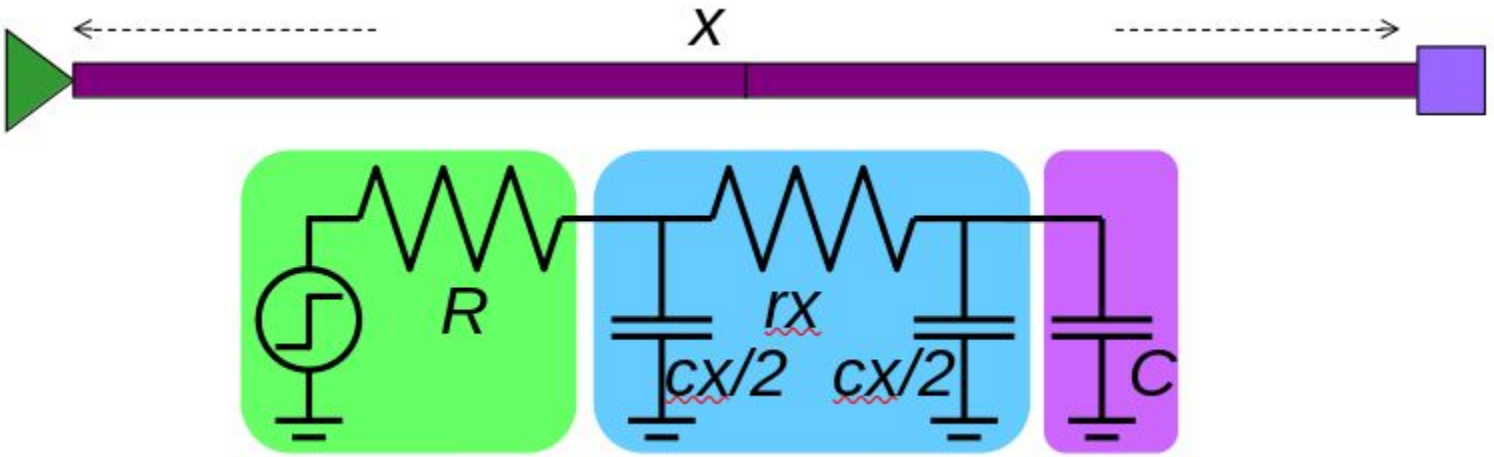
Driving resistance

Elmore Delay for A Circuit

- Delay = $\Sigma_{\text{all Ri}} \Sigma_{\text{all Cj downstream from Ri}}$ Ri*Cj
- Elmore delay to n1 R(B)*(C1+C2)
- Elmore delay to n2 R(B)*(C1+C2)+R(w)*C2

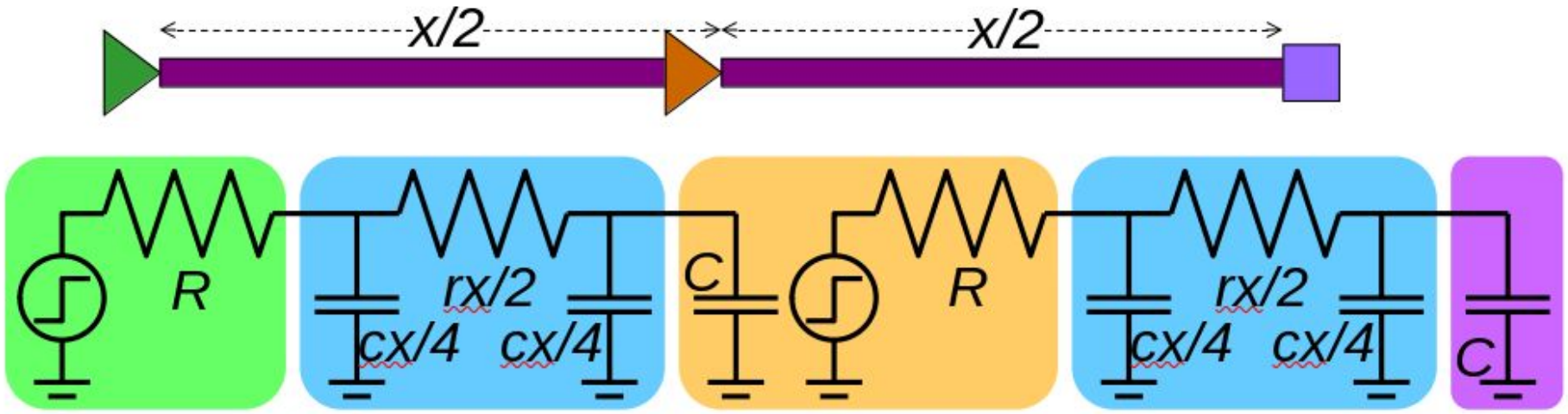


Unbuffered Wire Delay



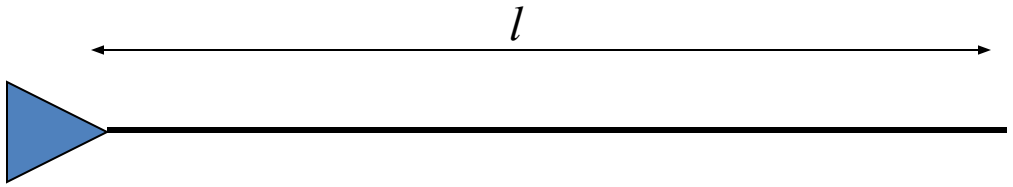
$t\_unbuf = R(cx + C) + rx(cx/2 + C)$

Buffered Wire Delay

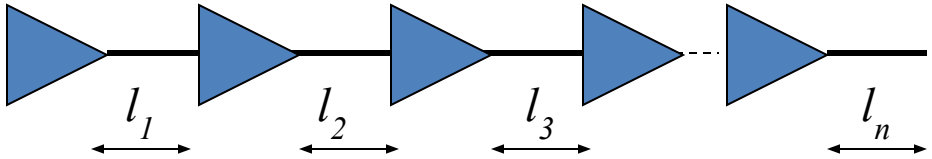


$t\_buf = 2R(cx/2 + C) + rx(cx/4 + C)$

Buffered global interconnects: Intuition



Interconnect delay = $r.c.l^2/2$



Interconnect delay = $\Sigma\, r.c.l_i^2/2 < r.c.l^2/2$ *(where* $l = \Sigma\, l_j$ *)*
since $\boldsymbol{\Sigma\,(l_j^2) < (\Sigma\, l_j)^2}$

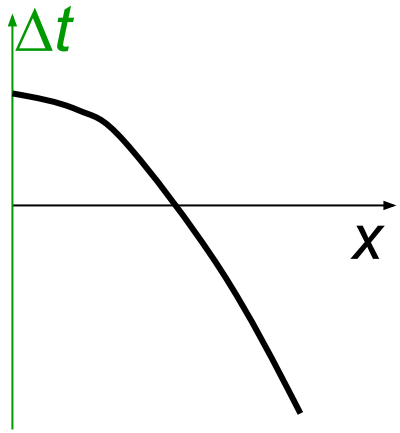
(Of course, we need to consider buffer delay as well)

Buffers Reduce Wire Delay

$$t\_unbuf = R(cx + C) + rx(cx/2 + C)$$

$$t\_buf = 2R(cx/2 + C) + rx(cx/4 + C)$$

$$t\_buf - t\_unbuf = RC - rcx^2/4$$



Buffer delay

Reduced wire delay

When does it make sense to use a buffer? x = ?

What is the optimal length?

- Can compute based on R, C of wire and gate
- However, it is ok to approximate and loss of delay is not bad
  - ~0.5mm?
  - A few 100um?
- In general, the tools approach it in another way.

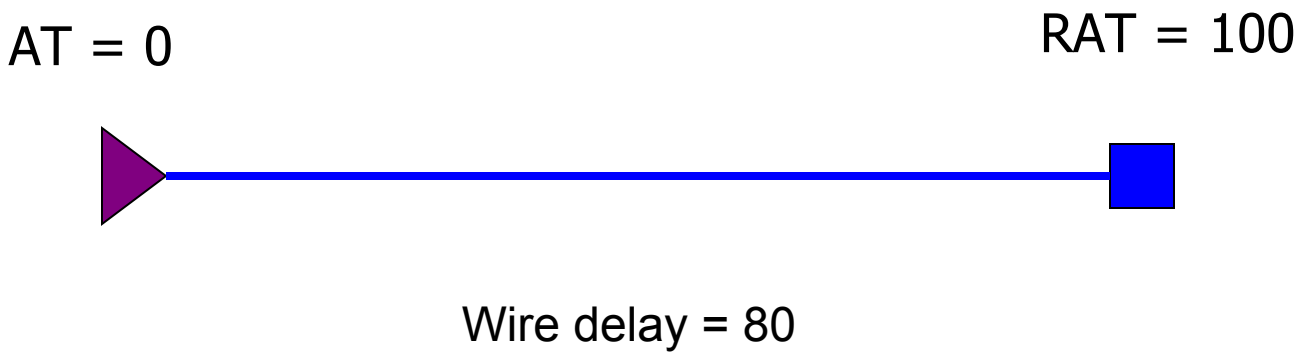
Buffer Placement

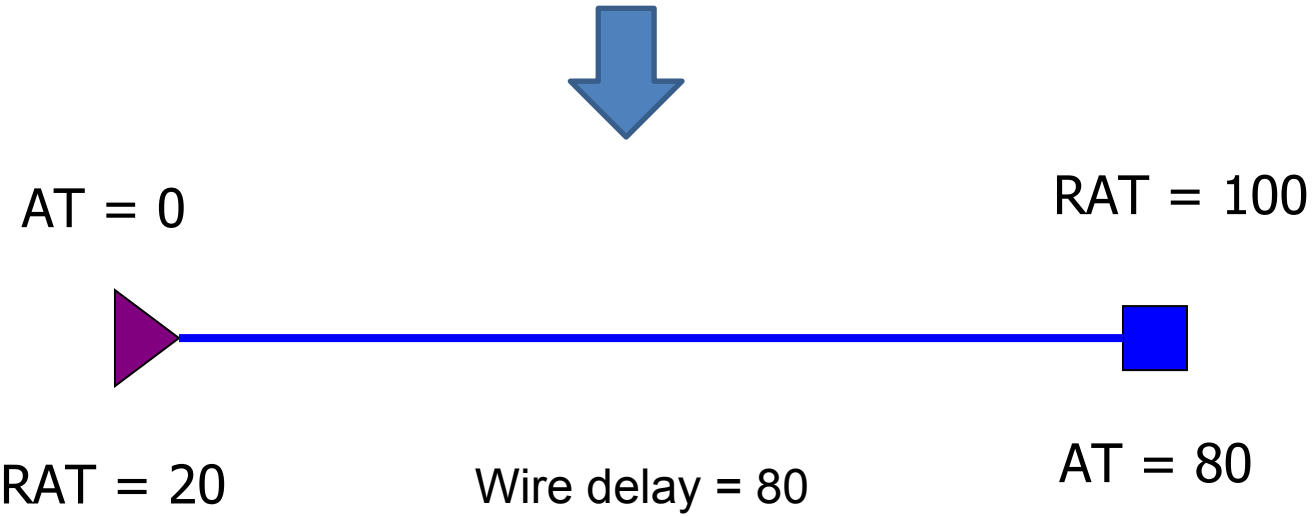
- The previous formulations assume that you can put a buffer anywhere.
  - What if there are a limited number of open spaces?
- New problem: Timing driven buffer placement and sizing.
  - Uses Static Timing Analysis!

Today's Lecture

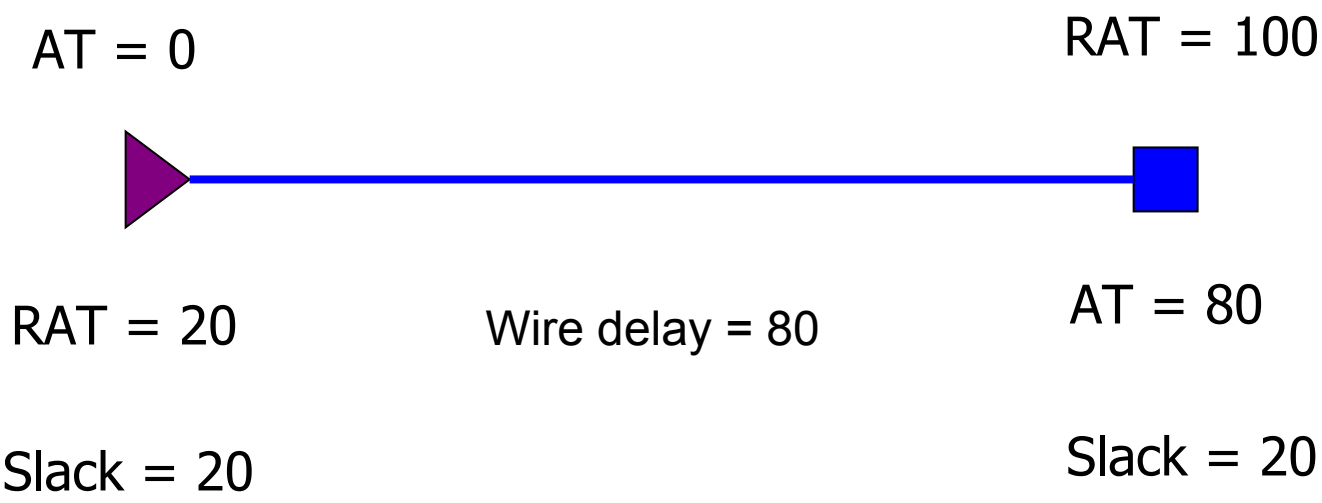
- Cascaded Buffers
  - Driving large capacitive loads
- Repeaters
  - Driving long wires
- Generalized Buffering
  - Capacitive shielding
  - Driving fanout with varying criticalities
  - Van Ginneken algorithm

RAT: Required Arrival Time



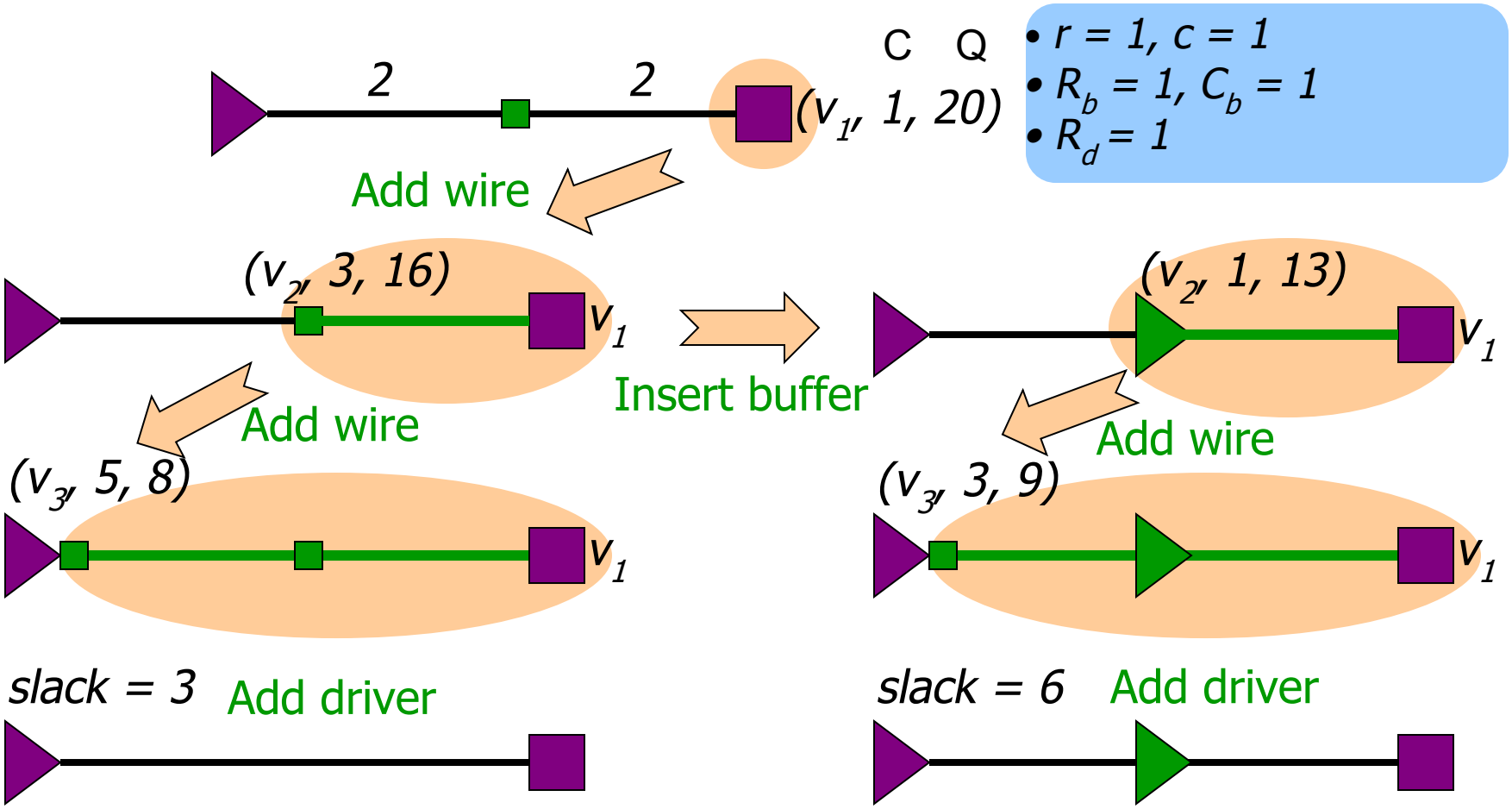


Slack: RAT - AT

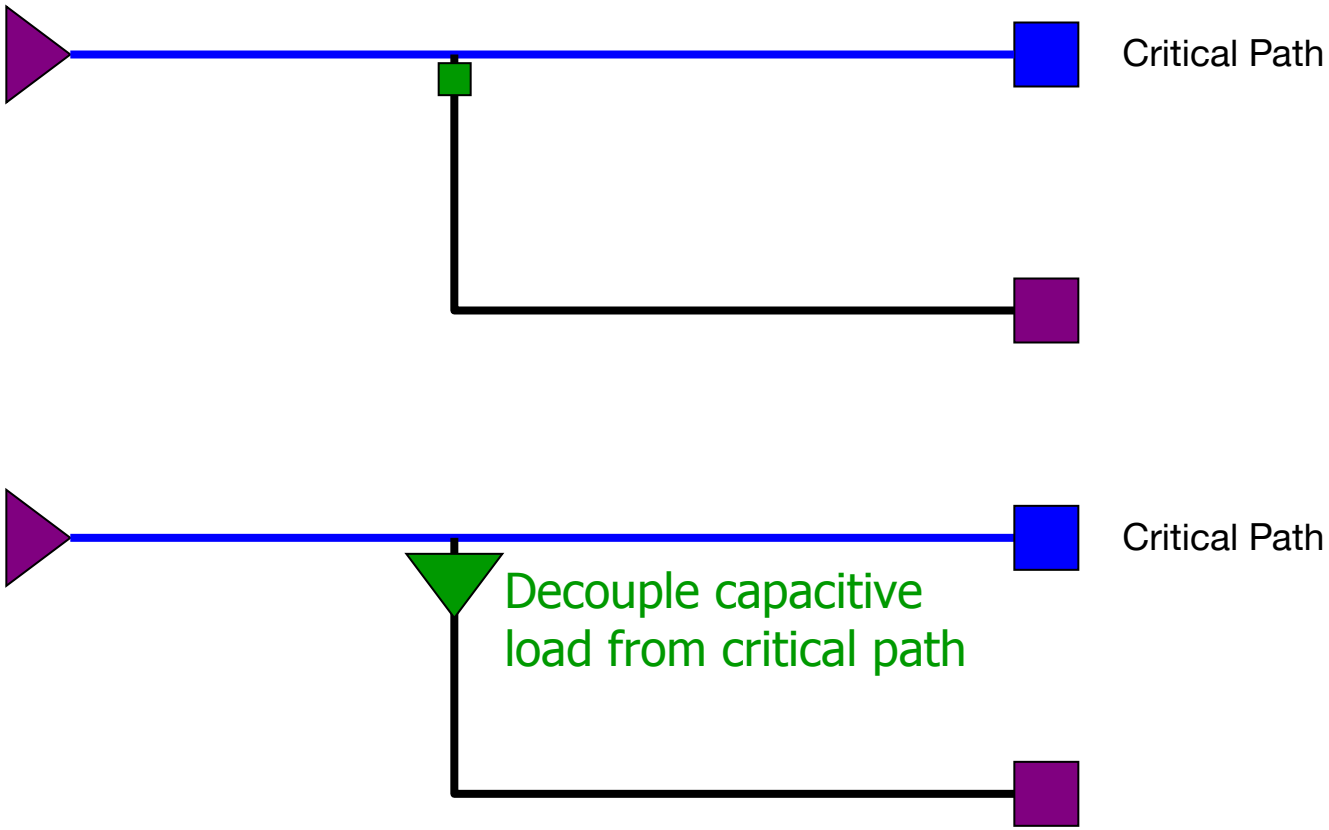


Minimizing circuit delay = maximizing RAT at driver = maximizing slack at driver

An Example for Buffer Insertion

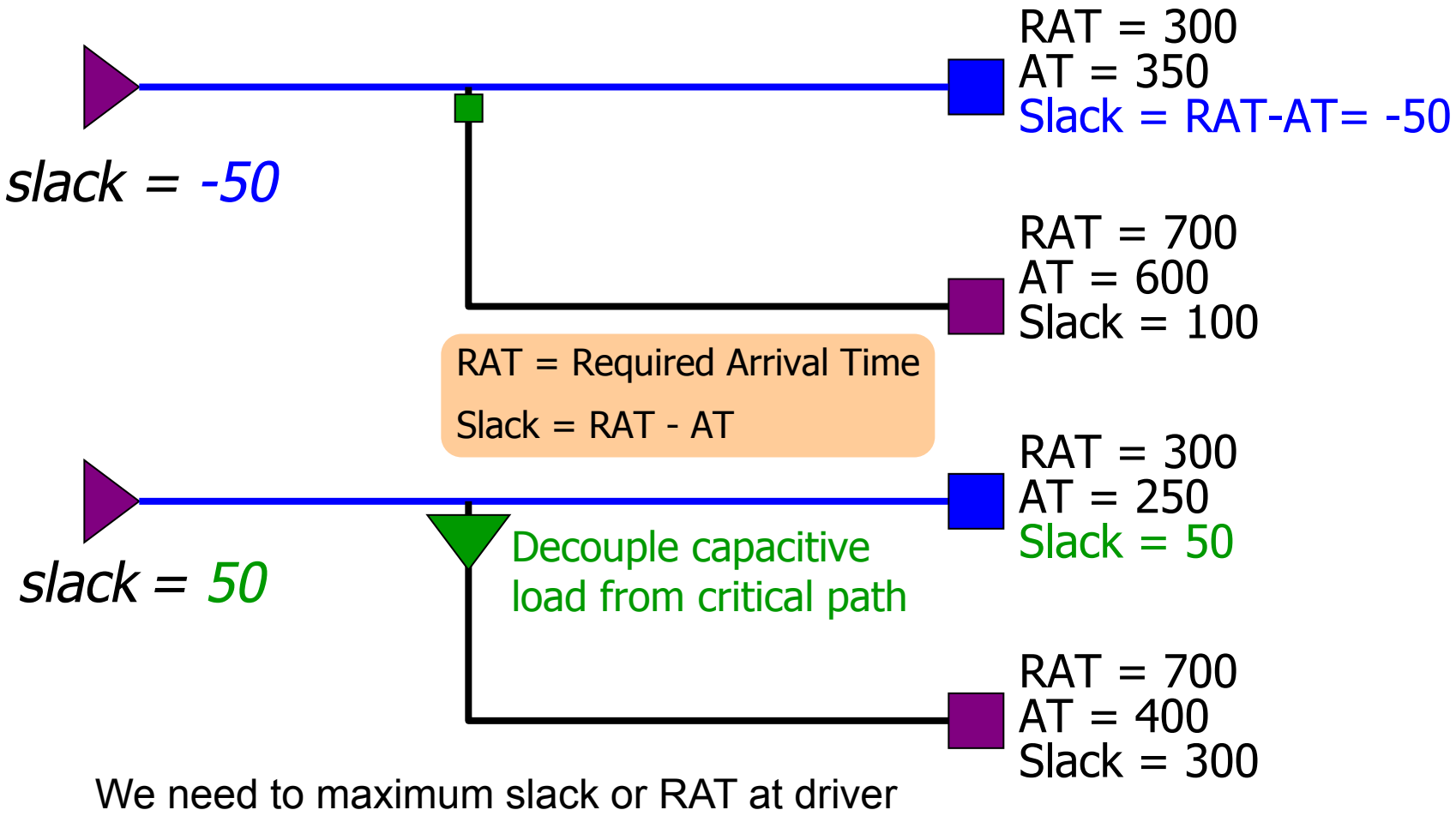


Capacitance Shielding



Makes sense if buffer input capacitance is less than fanout cap and wire

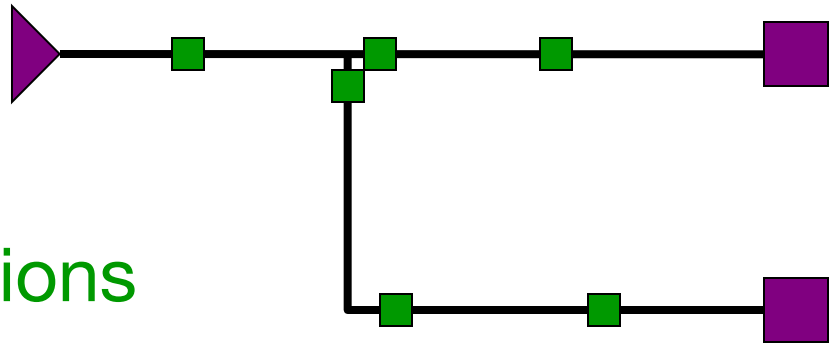
Motivation for Problem Formulation



We need to maximum slack or RAT at driver

General Timing Driven Buffering Problem

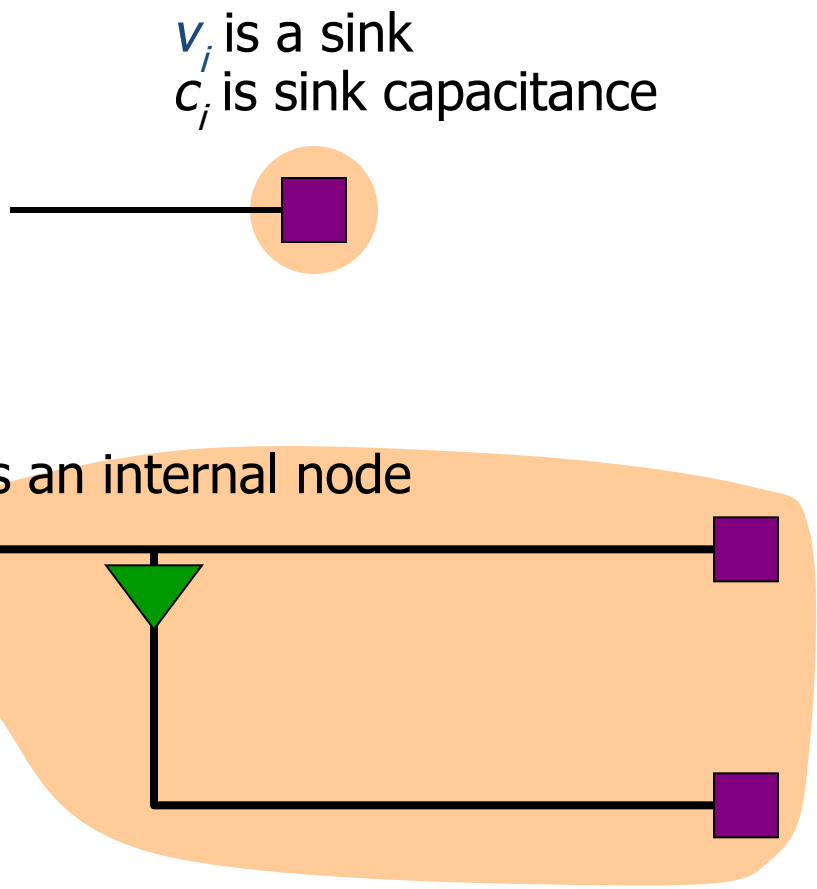
- Given
  - A Steiner tree
  - RAT at each sink
  - A buffer type
  - RC parameters
  - Candidate buffer locations



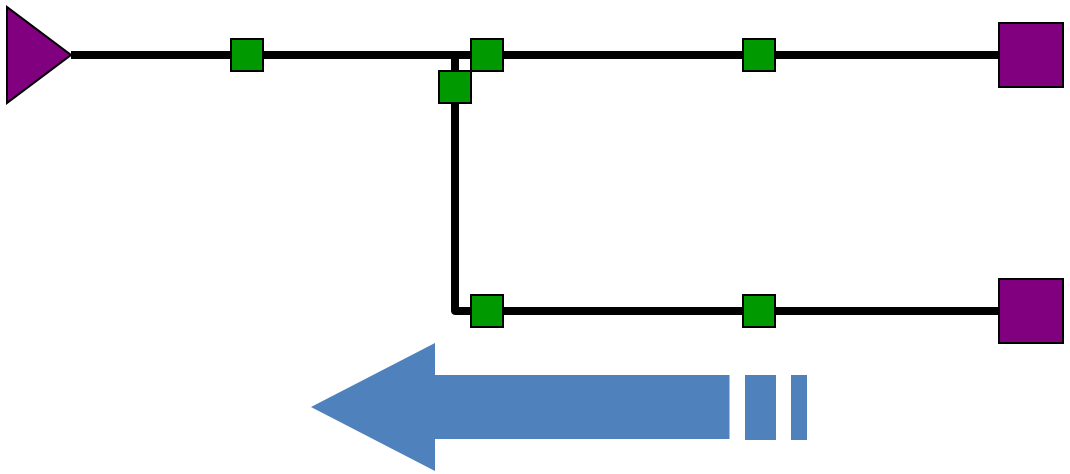
- Find buffer insertion solution such that the slack (or RAT) at the driver is maximized

Candidate Buffering Solution

- Definition
- Each candidate solution is associated with
  - v_i: a node
  - c_i: downstream capacitance
  - q_i: RAT

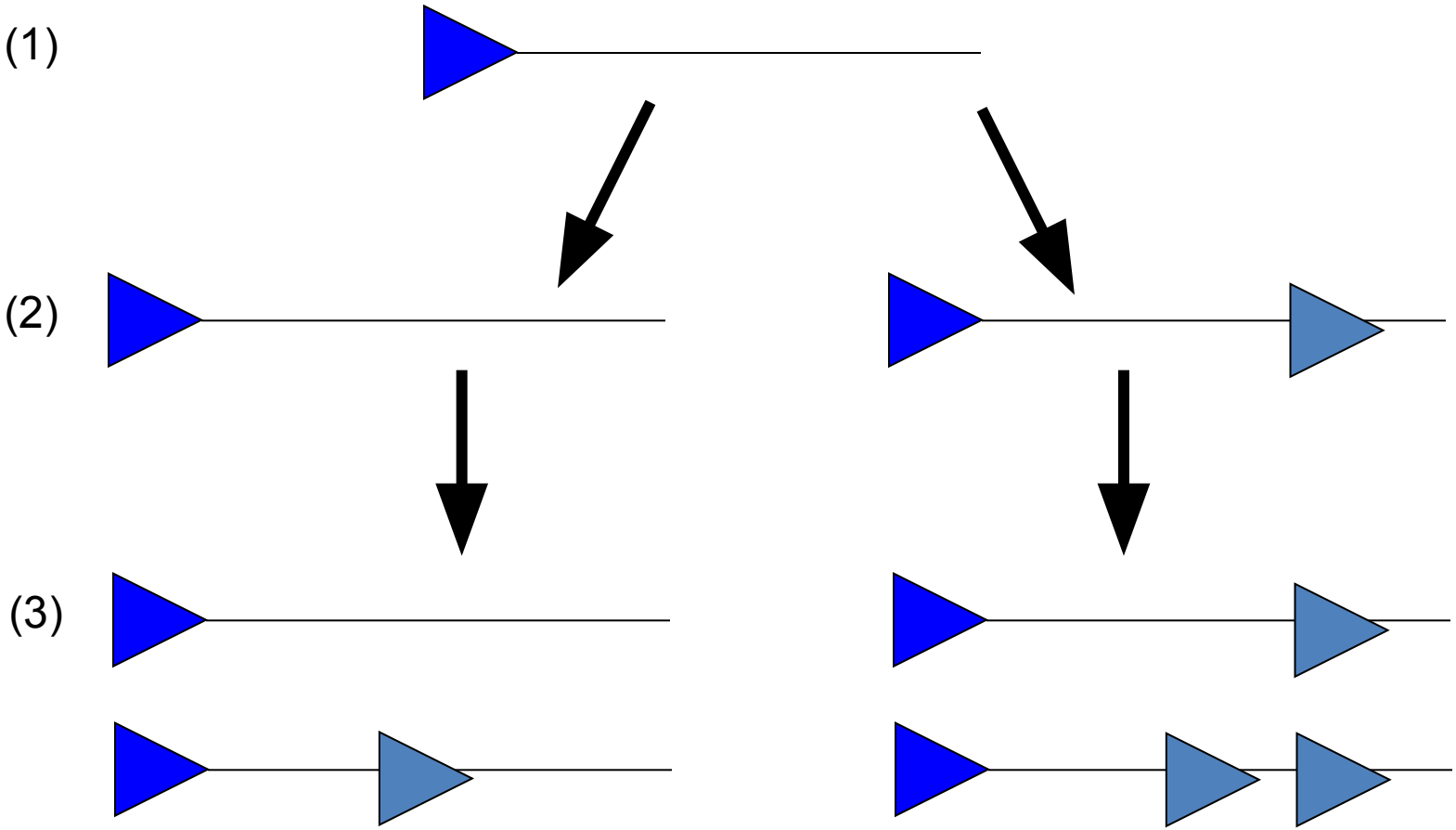


Van Ginneken's Algorithm

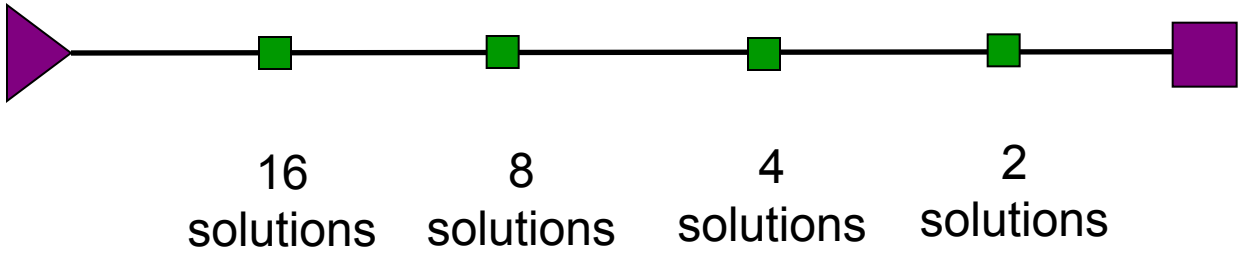


Candidate solutions are propagated toward the source

Generating Candidates



Exponential Runtime

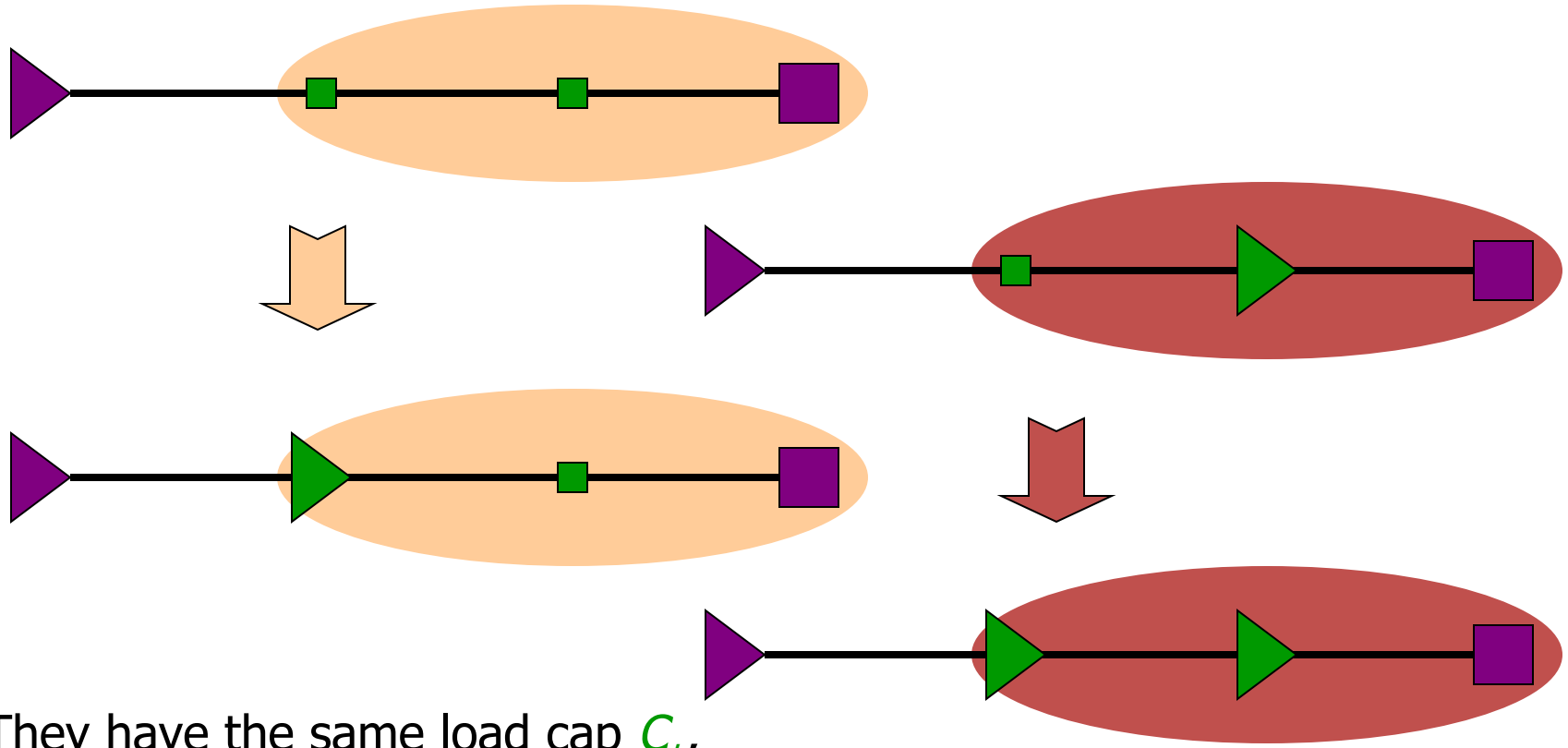


n candidate buffer locations lead to 2^n solutions

Solution Pruning

- Two candidate solutions
  - (v, c_1, q_1)
  - (v, c_2, q_2)
- Solution 1 is inferior if
  - $c_1 \geqslant c_2$: larger load
  - and $q_1 \leqslant q_2$: tighter timing
- **I.e., it keeps "pareto optimal" solutions only!**

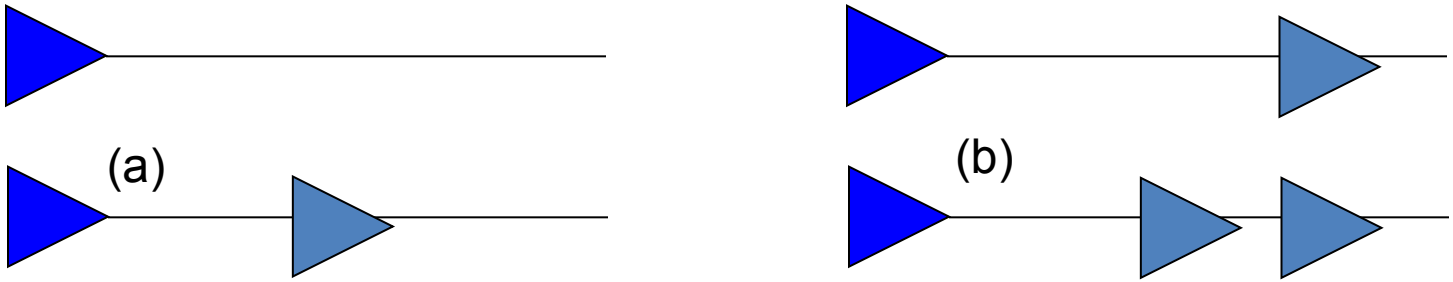
Pruning When Insert Buffer



They have the same load cap C_b,
only the one with max q is kept

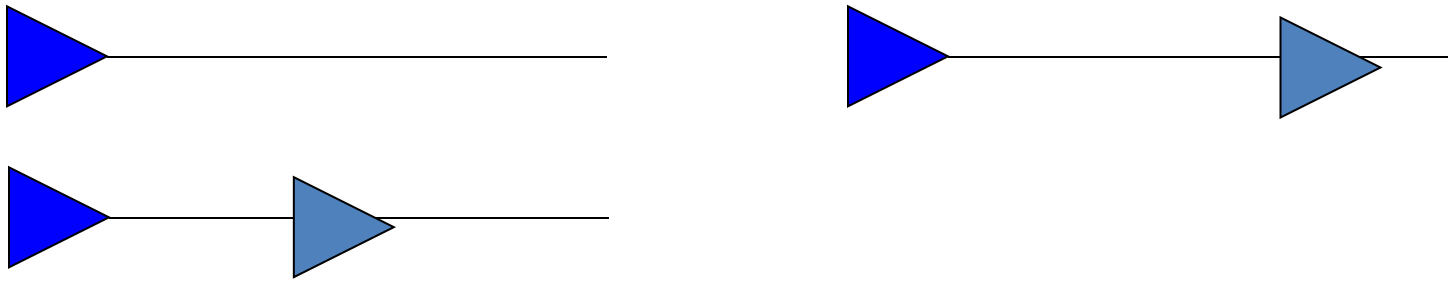
Pruning Candidates

(3)

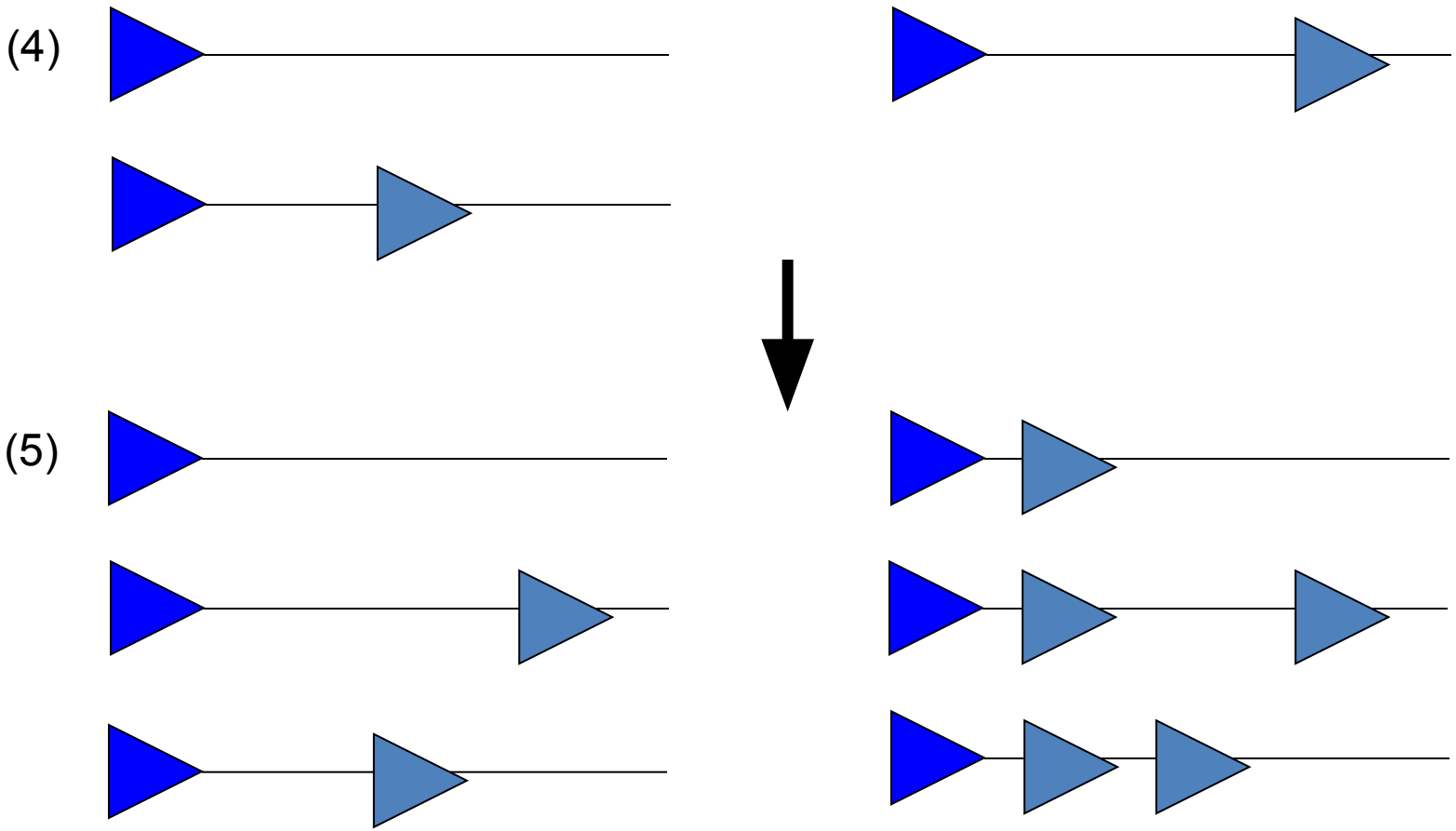


Both (a) and (b) “look” the same to the source.
Throw out the one with the worse slack

(4)



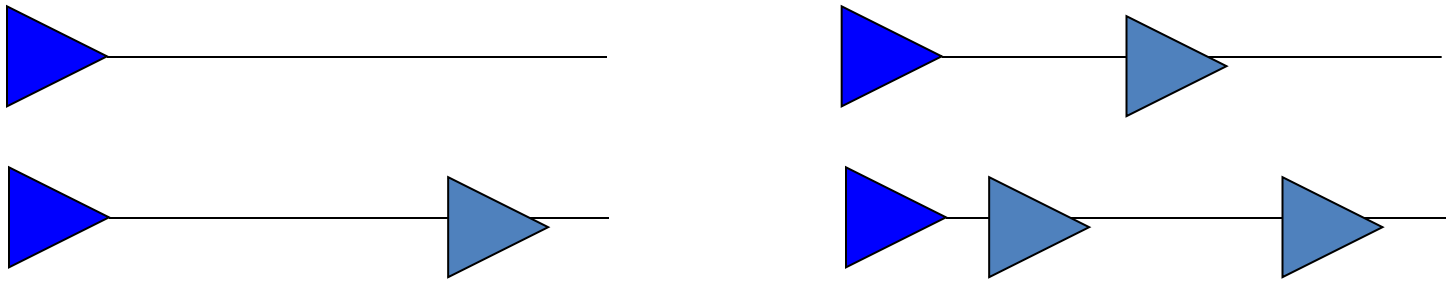
Candidate Example Continued



Candidate Example Continued

After pruning

(5)



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

Next Lecture

- Chip finishing