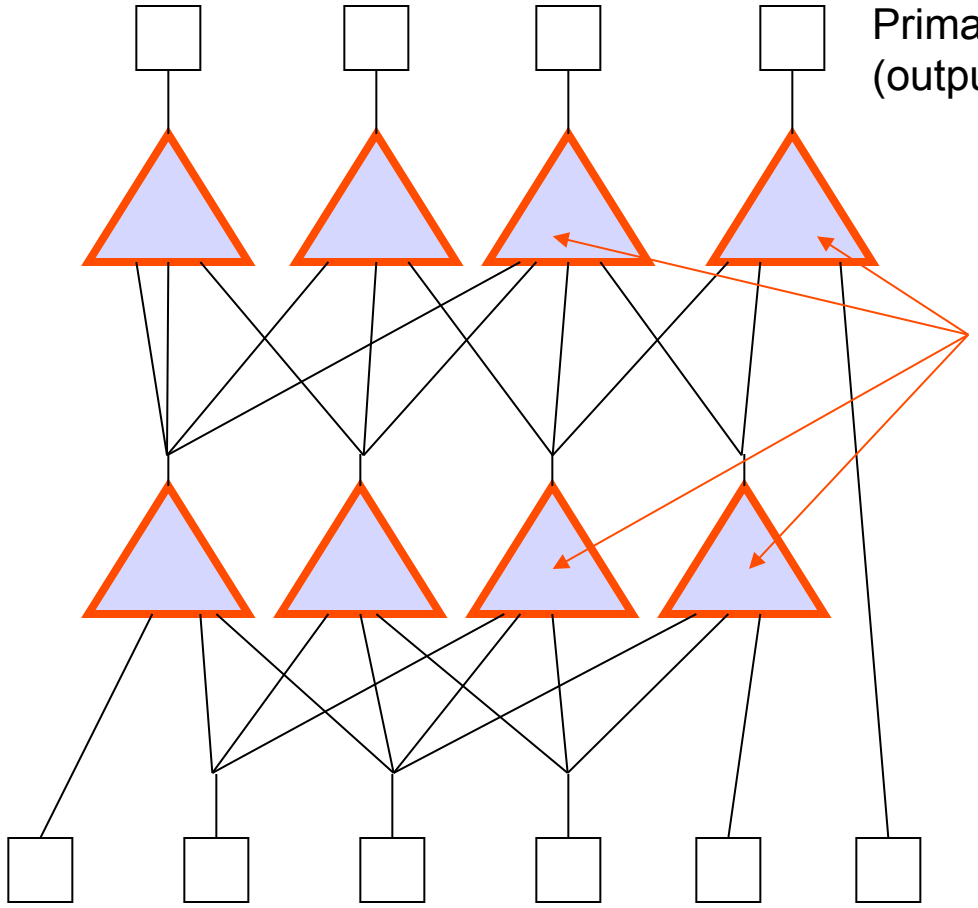
VLSI System Design
Part III : Technology Mapping (3)

Lecturer : Tsuyoshi Isshiki
Dept. Communications and Computer Engineering,
Tokyo Institute of Technology
isshiki@vlsi.ce.titech.ac.jp

Timing-Driven Technology Mapping (1)

1. Input :
   - Circuit description : Boolean Network (DAG)
   - Timing Constraints : Maximum arrival time from primary inputs to primary outputs (often corresponding to maximum clock period)
2. Output :
   - Technology-mapped gate-level netlist satisfying the specified timing constraints with minimum circuit area
3. Computation flow
   - Partition the target DAG into trees, leaf-DAGs or cones (call this the *circuit blocks*)
   - At each circuit block : delay-optimal tree covering
   - At each connections between circuit blocks (multiple fan-out nets) : fan-out optimization
   - Nodes on non-critical paths : area recovery
   - Paths with timing violations : Boolean Network restructuring

Timing-Driven Technology Mapping (2)



Primary outputs
(output pins, register inputs)

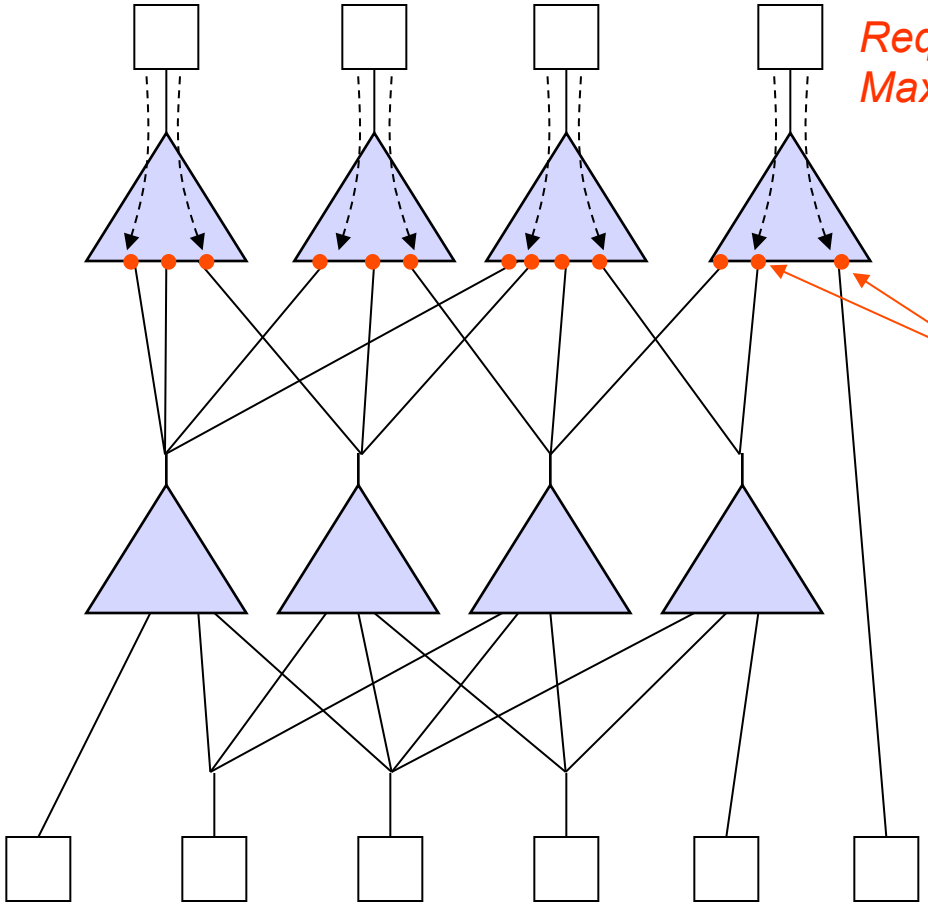
Tree / leaf-DAG / cone

Delay-optimal
tree covering

(assume 0 arrival time at
each leaf inputs)

Primary inputs
(input pins, register outputs)

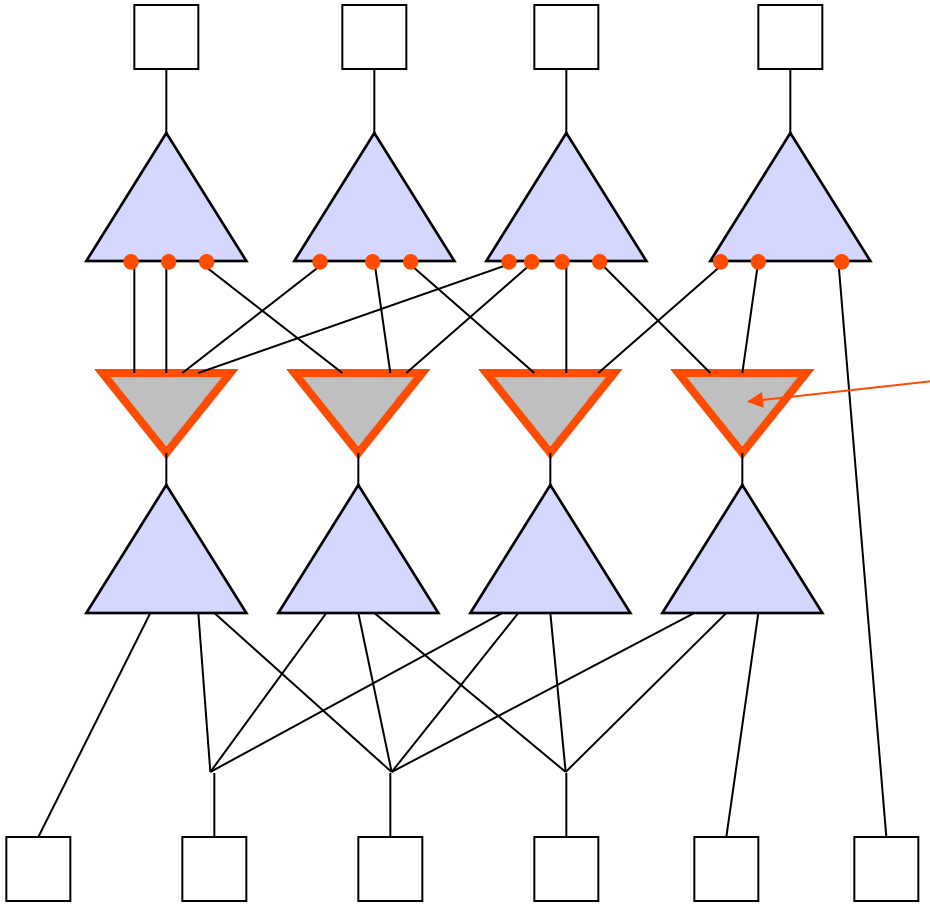
Timing-Driven Technology Mapping (3)



Required time at primary output : Maximum clock period

Calculate required times at leaf inputs on trees connecting to the primary outputs (*1st level trees*)

Timing-Driven Technology Mapping (4)

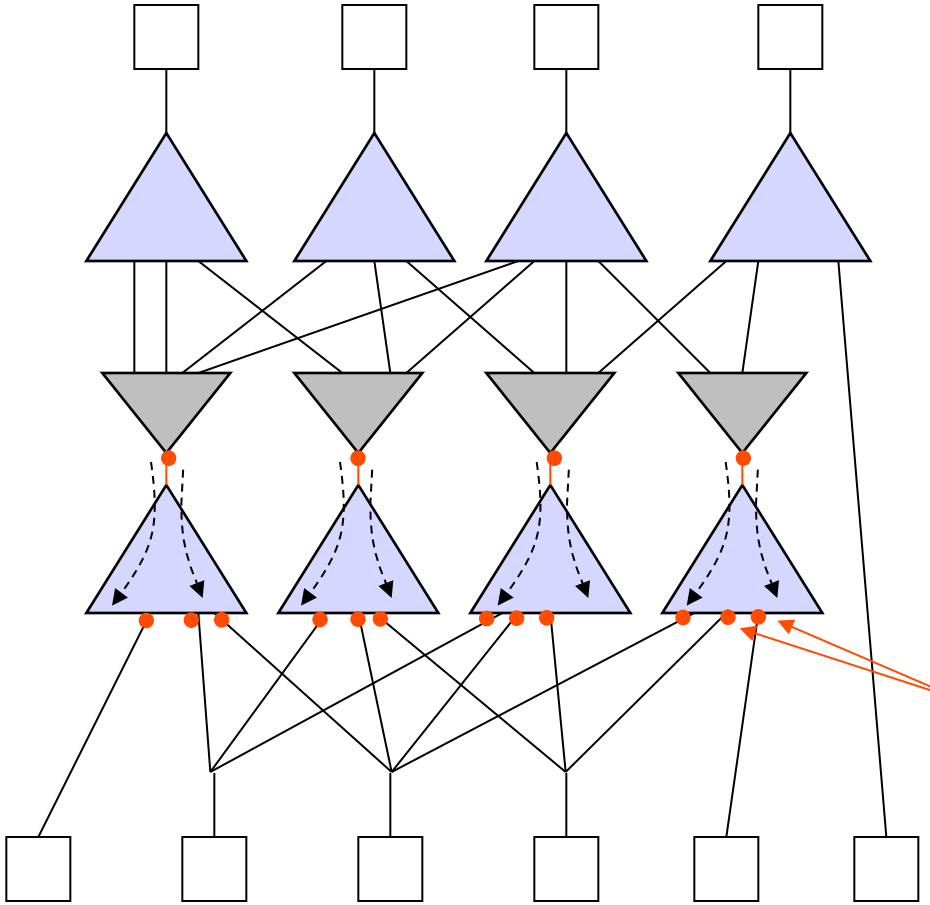


Fan-out tree (1st level)

Fan-out optimization at the leaf inputs of 1st level trees

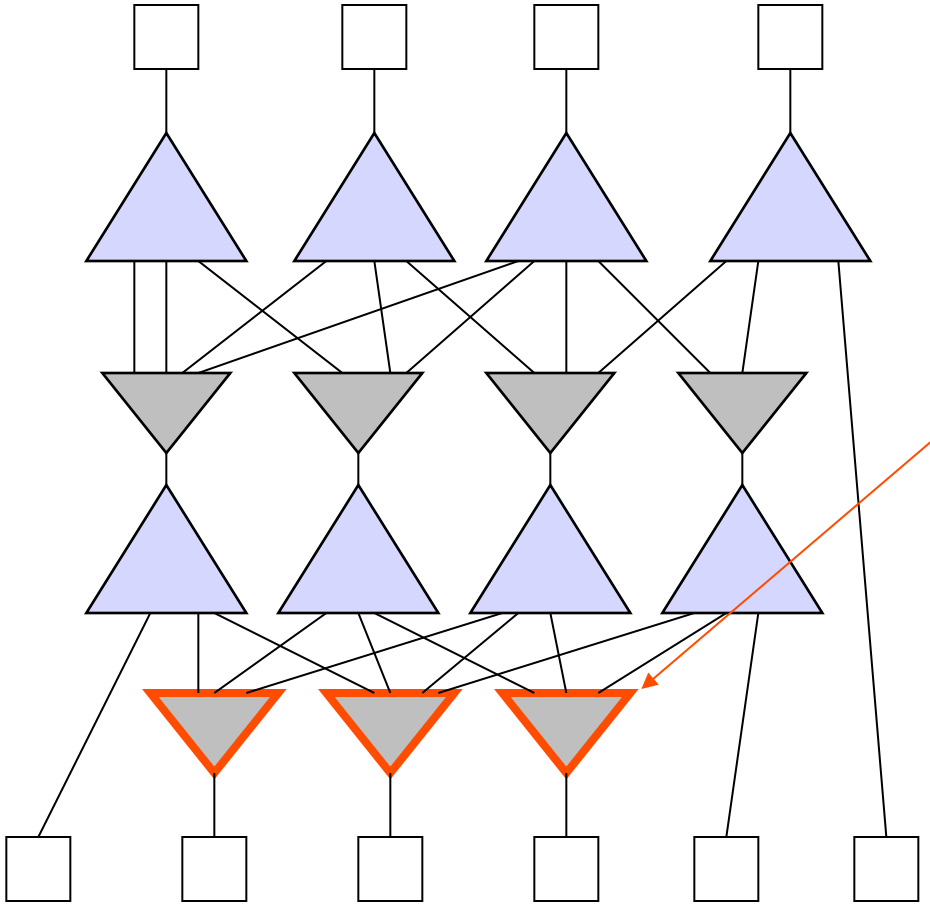
(maximize required times at the source of multiple fan-out nets)

Timing-Driven Technology Mapping (5)



Calculate required times at leaf inputs on trees connecting to 1st level fan-out trees (*2nd level trees*)

Timing-Driven Technology Mapping (6)

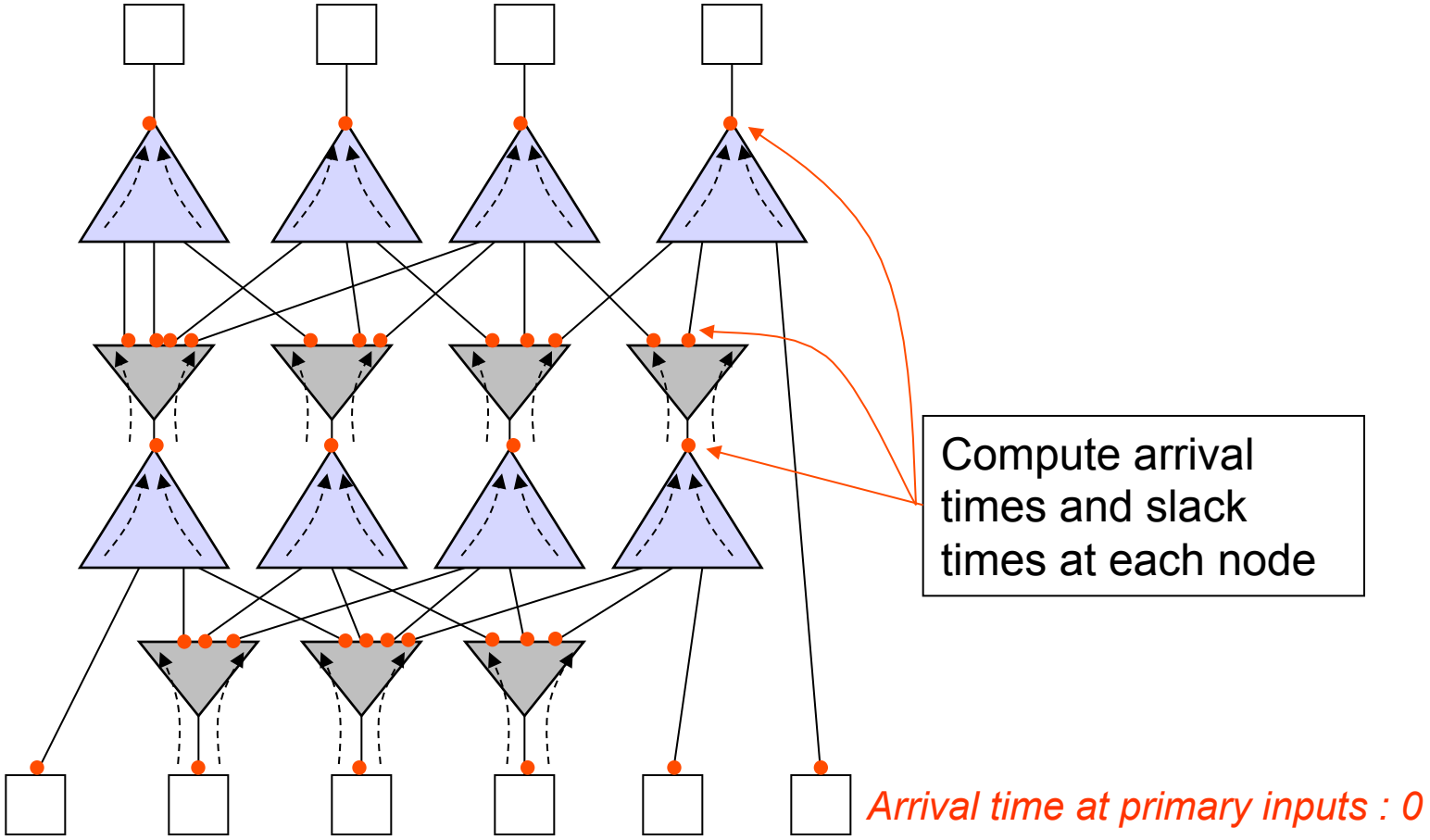


Fan-out tree (2nd level)

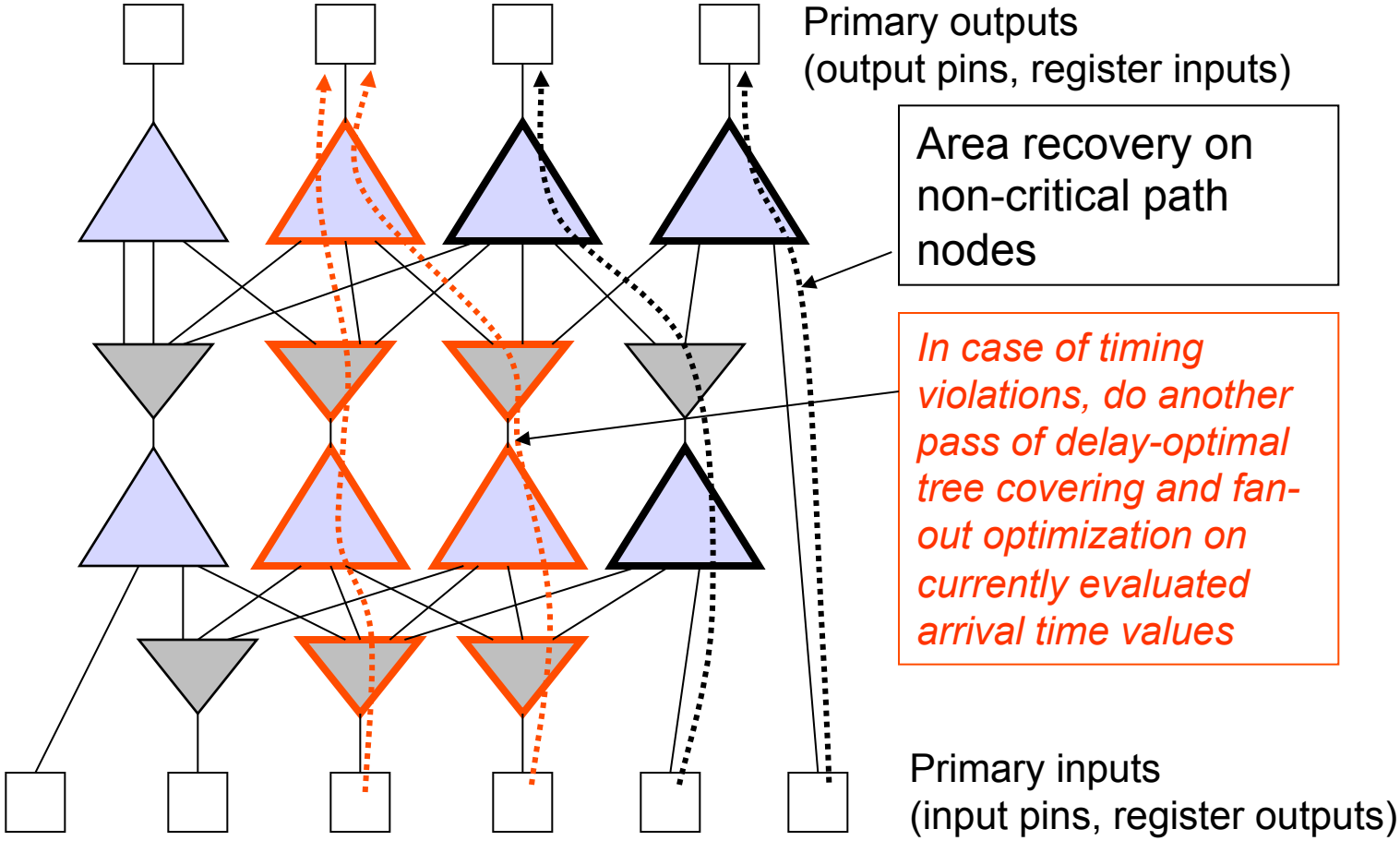
Fan-out optimization at the leaf inputs of 2nd level trees

Continue towards the primary inputs until all multiple fan-out nets have been evaluated.

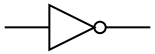
Timing-Driven Technology Mapping (7)

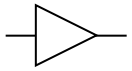
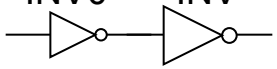
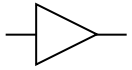
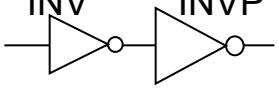


Timing-Driven Technology Mapping (8)



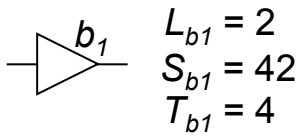
Buffer Cells for Fan-out Trees

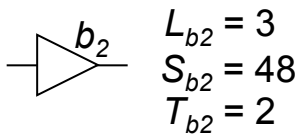
symbol	cell name	area	gate load	switching delay	output transition coef
	INV0	2	2	12	6
	INV	2	3	12	4
	INVP	3	6	12	2

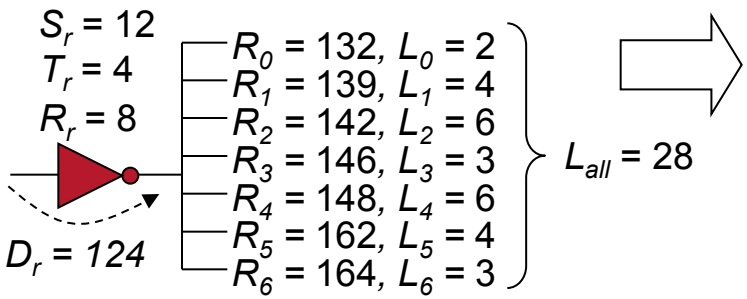
		area	gate load	switching delay	output transition coef
BUF1 	INV0 INV 	3	2	12 + 6 * 3 + 12 = 42	4
BUF2 	INV INVP 	5	3	12 + 4 * 6 + 12 = 48	2

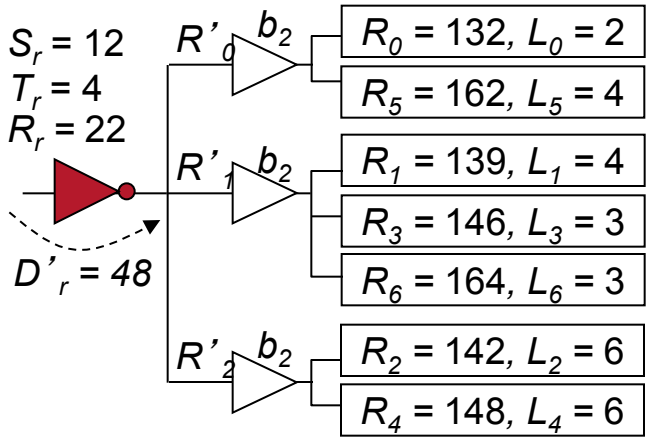
Fan-Out Optimization

- Fan-out optimization : Construct a fan-out tree which maximize the required time R_r at the net source r on following conditions
  - Net source r :
    - Output transition coefficient : T_r
      (Switching delay S_r is not really needed in the optimization)
  - Buffer cell b_j :
    - Gate load : L_{bj}
    - Switching delay : S_{bj}
    - Output transition coefficient : T_{bj}
  - Sink i (i = 1, 2, … n)
    - Gate load : L_i
    - Required time R_i



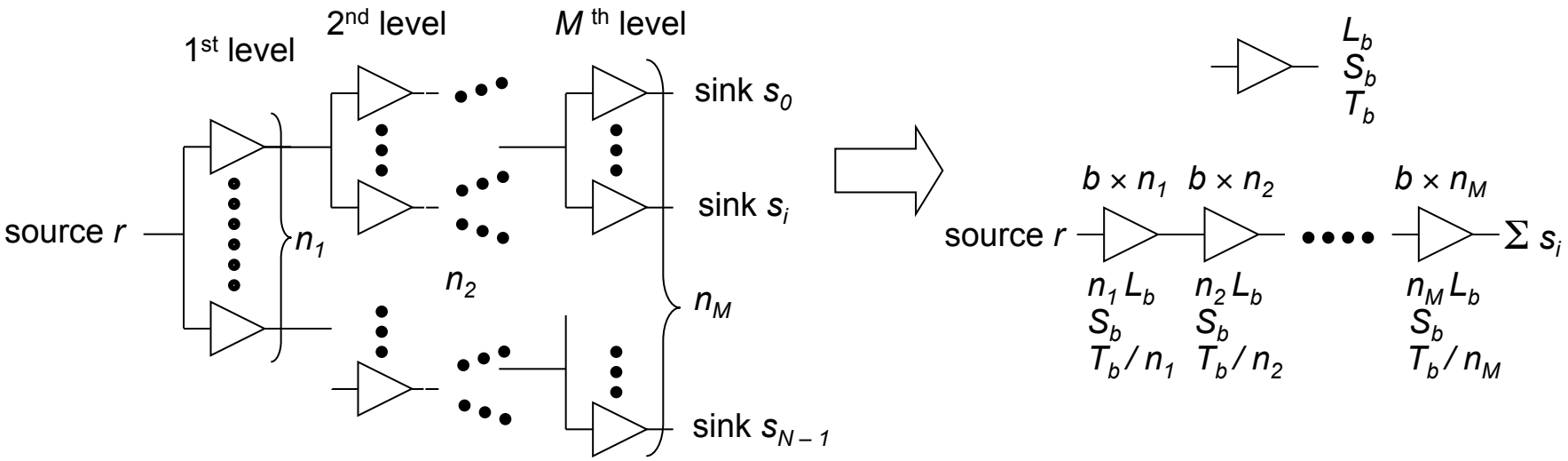




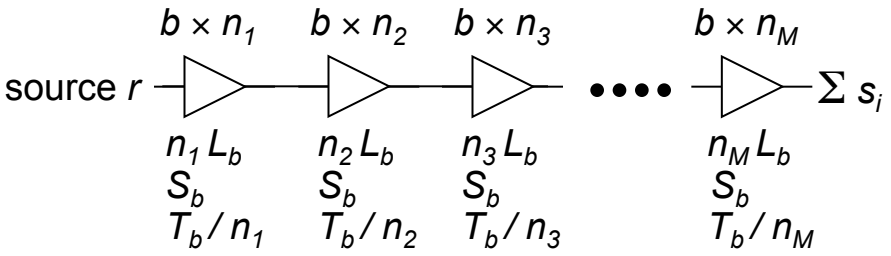


Balanced Fan-Out Tree (1)

- Assumptions :
  - Required times at all sinks are identical. *(Ex. clock signals)*
  - Fan-out tree is balanced with a height of *M*.
  - Use only one type of buffer cell *b*.
- Compute the optimal number of buffers at each level of the tree. (n_k : number of buffers at k^{th} level, $n_k > 0$)
  - The n_k buffers at k^{th} level can be modeled as a single buffer with the gate load $n_k L_b$, switching delay S_k and output transition T_b / n_k



Balanced Fan-Out Tree (2)



- Delay from source to each sink (L_{all} : sum of all sink loads):

  $D_M = T_r(n_1L_b) + S_b + (T_b/n_1)(n_2L_b) + S_b + (T_b/n_2)(n_3L_b) + \ldots + (T_b/n_M)\,L_{all}$

  $= T_r(n_1L_b) + (T_b/n_1)(n_2L_b) + (T_b/n_2)(n_3L_b) + \ldots + (T_b/n_M)\,L_{all} + M\cdot S_b$

- Partial derivative on D_M with respect to each n_k :

  $\partial D_M/\partial n_1 = T_r L_b - T_b L_b n_2/n_1^2 = 0 \Rightarrow n_1^2 = n_2 T_b/T_r$

  $\partial D_M/\partial n_2 = T_b L_b/n_1 - T_b L_b n_3/n_2^2 = 0 \Rightarrow n_2^2 = n_1 n_3 \Rightarrow n_2/n_1 = n_3/n_2$

  $\partial D_M/\partial n_3 = T_b L_b/n_2 - T_b L_b n_4/n_3^2 = 0 \Rightarrow n_3^2 = n_2 n_4 \Rightarrow n_3/n_2 = n_4/n_3$

  : : : :

  $\partial D_M/\partial n_M = T_b L_b/n_{M-1} - T_b L_{all}/n_M^2 = 0 \Rightarrow n_M^2 = n_{M-1} L_{all}/L_b$

Balanced Fan-Out Tree (3)

- The number of buffers at *k-th* level n_k when D_M is minimum:

$n_1^2 = n_2 T_b / T_r$

$n_2/n_1 = n_3/n_2 = n_4/n_3 = \ldots n_M/n_{M-1} = r$

$n_M^2 = n_{M-1} L_{all} / L_b$

$\rightarrow n_1 = r\,(T_b / T_r)\,,\ n_M = (1/r)\,(L_{all} / L_b)\ \therefore n_1 n_M = (T_b / T_r)\,(L_{all} / L_b)$

$n_1 = n_2 / r = n_3 / r^2 = \ldots = n_M / r^{M-1} \therefore n_M = n_1 r^{M-1}$

$\rightarrow n_1 n_M = n_1^2 r^{M-1} = (T_b / T_r)^2 r^{M+1} = (T_b / T_r)\,(L_{all} / L_b)$

$\rightarrow r^{M+1} = (T_r / T_b)\,(L_{all} / L_b)$

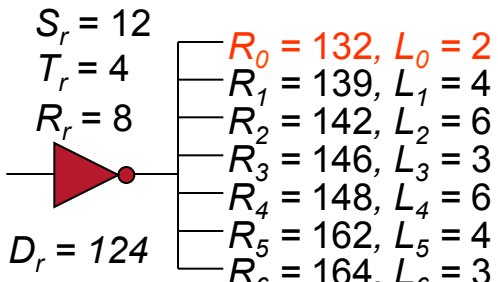
$\rightarrow r = (T_r L_{all} / T_b L_b)^{1/(M+1)}$

$\rightarrow n_k = (T_b / T_r)\,(T_r L_{all} / T_b L_b)^{k/(M+1)}$

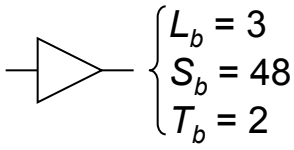
Two-Level Tree (1)

- Two-level tree : Restrict the tree height to be 1 ($M = 1$)
  - Delay from source to each sink :
    $D = T_r(n_1 L_b) + S_b + (T_b / n_1) L_{all}$
  - n_1 when D is minimized : $n_1 = (T_b L_{all} / T_r L_b)^{1/2}$
  - Use only one type of buffer
  - ✓ *Even with this restricted tree structure, this optimization problem is NP-complete.*
- Two-level tree algorithm :
  1. Sort the sinks in the increasing order of their required times (in case of a tie, the decreasing order of the gate load)
  2. Set $n_1 = \lceil (T_b L_{all} / T_r L_b)^{1/2} \rceil$.
  3. Allocate each sink (in the sorted order) to one of the n_1 buffers
     - Choose the allocation which maximizes the required time at the source node.
  4. Compute the two-level tree for each buffer cell type, and choose the fastest.
  5. This is a greedy algorithm which do not guarantee optimality, but is a baseline algorithm for other more sophisticated methods.

Two-Level Tree (2)



$L_{all} = 28$



Delay :
$D_r = T_r * L_{all} + S_r$
$= 4 * 28 + 12 = 124$

Required time :
$R_r = R_0 - D_r$
$= 132 - 124 = 8$

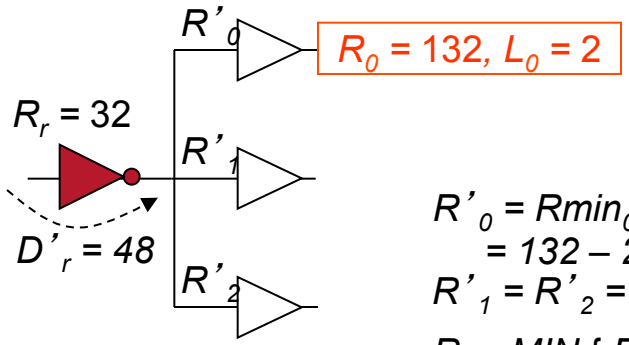
Optimal number of buffer cells :
$n_1 = \lceil (T_b L_{all} / T_r L_b)^{1/2} \rceil$
$= \lceil (56 / 12)^{1/2} \rceil$
$= \lceil 2.16 \rceil = 3$

$Rmin_i$: earliest required time among the child nodes of buffer i

L'_i : total load connected at buffer i

R'_i : required time at buffer i

$R'_i = Rmin_i - T_b * L'_i - S_b$



Delay (buffered):
$D'_r = T_r * L_b * n_1 + S_r$
$= 4 * 3 * 3 + 12 = 48$

$R'_0 = Rmin_0 - T_b * L'_0 - S_b$
$= 132 - 2 * 2 - 48 = 80$

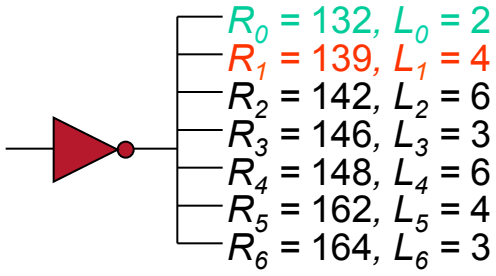
$R'_1 = R'_2 = \infty$

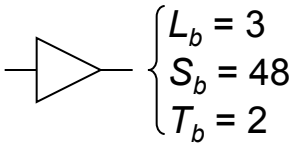
$R_r = MIN\{ R'_0, R'_1, R'_2 \} - D'_r$
$= 80 - 48 = 32$

buf	$Rmin_i$	L'_i	R'_i
0	132	2	80
1	∞	0	∞
2	∞	0	∞

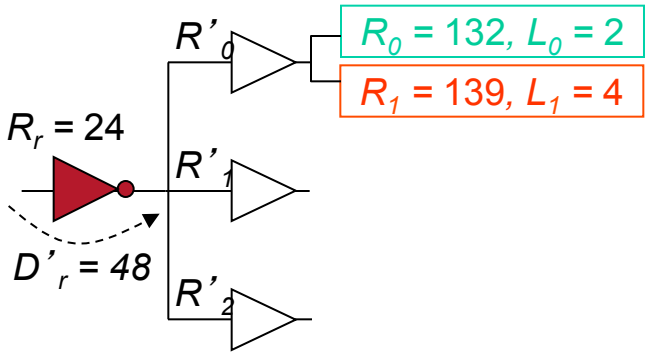
$MIN\{ R'_0, R'_1, R'_2 \} = 80$

Two-Level Tree (3)



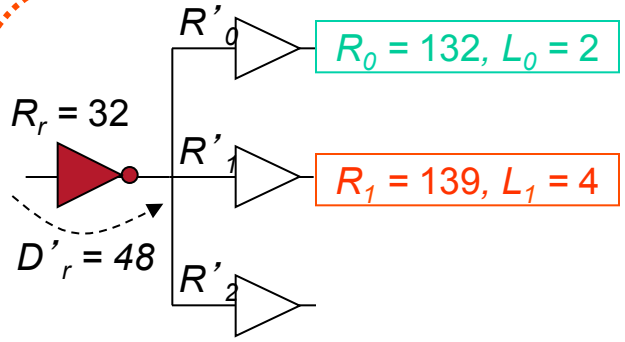


$$R'_i = Rmin_i - T_b * L'_i - S_b$$



buf	$Rmin_i$	L'_i	R'_i
0	132	6	72
1	∞	0	∞
2	∞	0	∞

$MIN\{R'_0, R'_1, R'_2\} = 72$

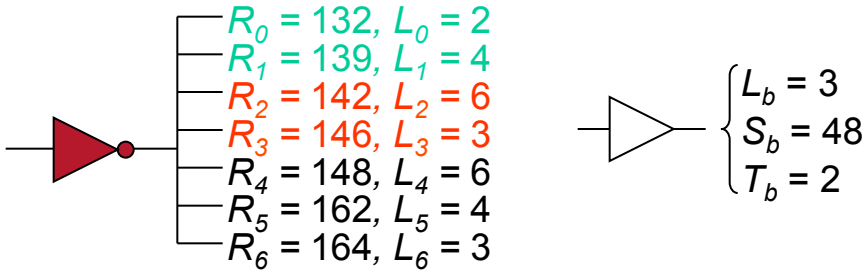


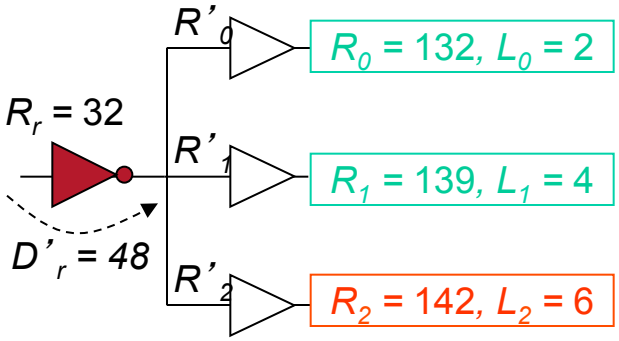
buf	$Rmin_i$	L'_i	R'_i
0	132	2	80
1	139	4	83
2	∞	0	∞

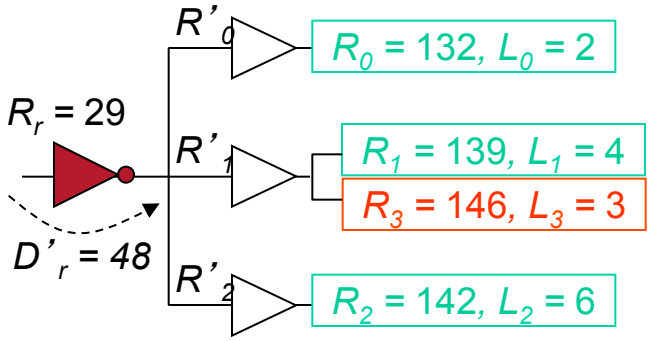
$MIN\{R'_0, R'_1, R'_2\} = 80$

This is a better allocation

Two-Level Tree (4)







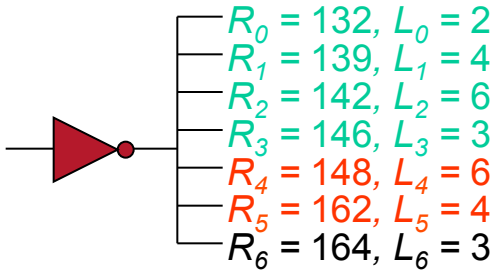
buf	$Rmin_i$	L'_i	R'_i
0	132	2	80
1	139	4	83
2	142	6	82

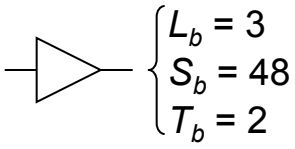
$MIN\{R'_0, R'_1, R'_2\} = 80$

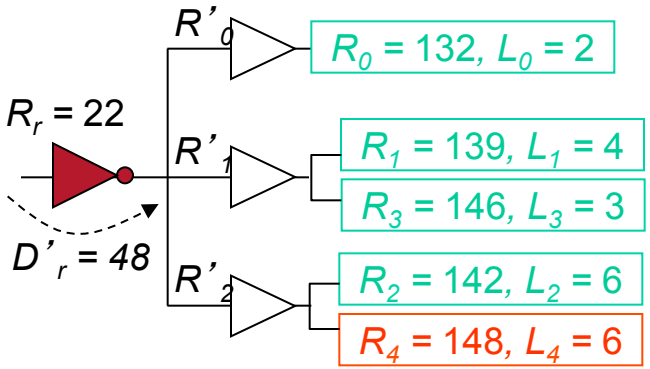
buf	$Rmin_i$	L'_i	R'_i
0	132	2	80
1	139	7	77
2	142	6	82

$MIN\{R'_0, R'_1, R'_2\} = 77$

Two-Level Tree (5)

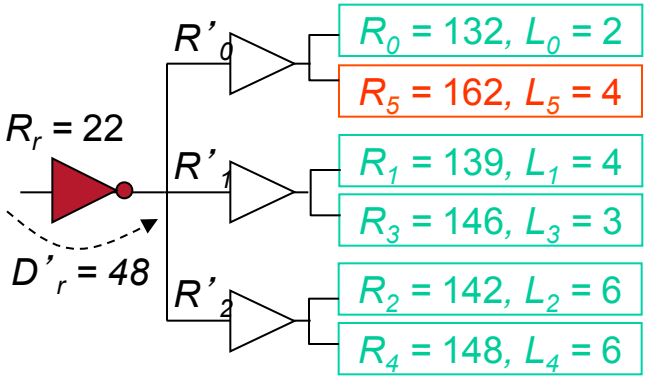






buf	$Rmin_i$	L'_i	R'_i
0	132	2	80
1	139	7	77
2	142	12	70

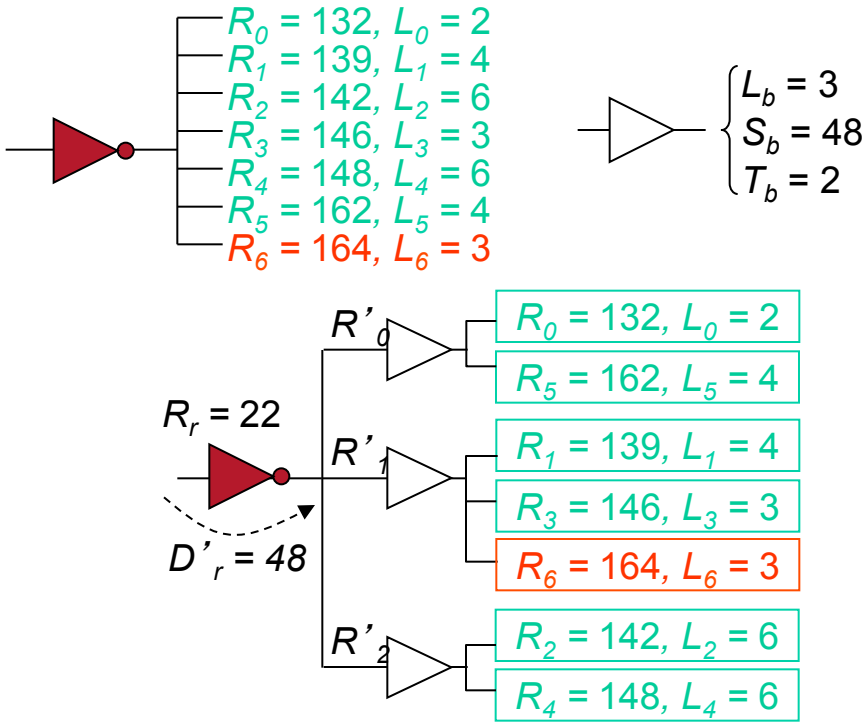
$MIN\{R'_0, R'_1, R'_2\} = 70$



buf	$Rmin_i$	L'_i	R'_i
0	132	6	72
1	139	7	77
2	142	12	70

$MIN\{R'_0, R'_1, R'_2\} = 70$

Two-Level Tree (6)



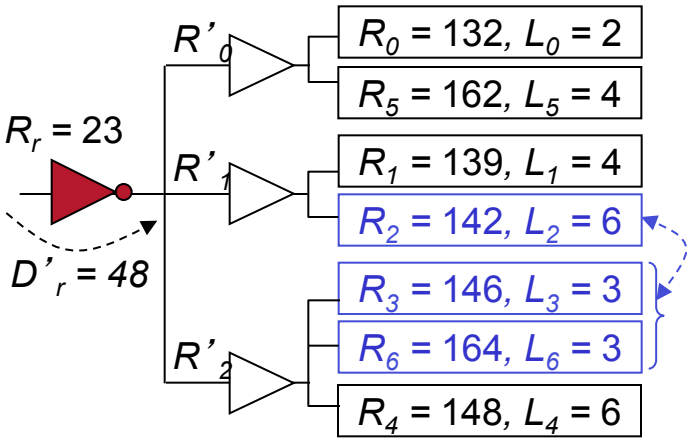
buf	$Rmin_i$	L'_i	R'_i
0	132	6	72
1	139	10	71
2	142	12	70

$MIN\{R'_0, R'_1, R'_2\} = 70$

IS THIS OPTIMAL???

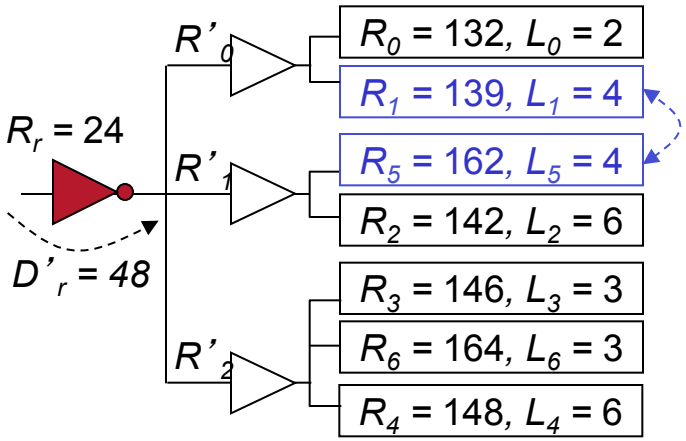
Two-Level Tree (7)

ACTUALLY, there are better solutions….



buf	$Rmin_i$	L'_i	R'_i
0	132	6	72
1	139	10	71
2	146	12	74

$MIN\{R'_0, R'_1, R'_2\} = 71$



buf	$Rmin_i$	L'_i	R'_i
0	132	6	72
1	142	10	74
2	146	12	74

$MIN\{R'_0, R'_1, R'_2\} = 72$

Combinational Merging (1)

- *Construction of general tree using multiple types of buffer cells*
  - Basic idea :
    - Incrementally insert buffer cells and connect the *k* sink nodes with the largest required times. (This expects that the effect of load reduction due to buffer insertion is larger than the penalty of added delays for these *least critical* sink nodes)
    - *k* is determined by the two-level tree equation (instead of determining the optimal number of buffer cells, compute the optimal amount of loads the buffer should drive).
    - Inserted buffers become new sink nodes (sink nodes connected to the inserted buffers are no longer sinks to the net source)

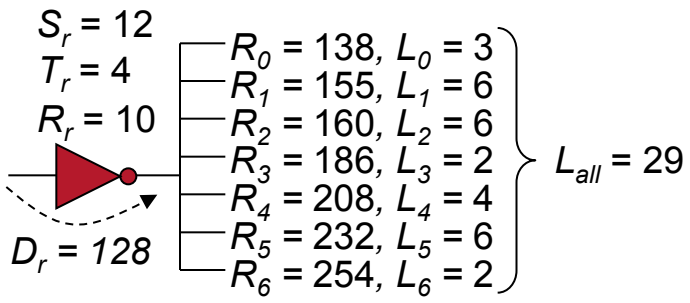
Combinational Merging (2)

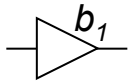
- Algorithm
  1. Sort the sinks in the increasing order of their required times (in case of a tie, the decreasing order of the gate load)
  2. For each buffer cell b_j, compute the optimal number of sinks k_{bj} (from the tail of the sink list) to be connected to b_j.
     - ✧ L_{all} : total gate loads in the sink list
     - ✧ $n_{bj} = (T_j L_{all} / T_r L_j)^{1/2}$: optimal number of buffers in two-level tree using cell b_j
     - ✧ L_{all} / n_{bj} : Optimal load per single buffer b_j
     - ✧ L'_k: total gate loads of the last k nodes in the sink list
     - ➢ k_{bj} is the smallest k which satisfies $L'_k \geq L_{all} / n_{bj}$

Combinational Merging (3)

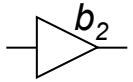
3. For each cell type b_j, let $R(b_j)$ be the required time at the source r where only a single cell of b_j is connected to r and cell b_j is connected to the last k (= k_{bj}) nodes in the sink list.
   - ✧ $R(b_j) = R'_k - T_{bj} L_k - S_{bj} - T_r L_{bj}$
   - ✧ R'_k : required time of the k-th node from the bottom of the sink list
   - ➢ Choose the cell type b_j which gives the largest $R(b_j)$ (this will have the largest speed up effect)
4. Update sink list :
   - ➢ Insert the cell b_j to the fan-out tree
   - ➢ Delete the k_{bj} nodes from the sink list (since they are buffered by b_j)
   - ➢ Add b_j to the sink list
     - ✧ Required time at the inserted b_j cell *:* $R(b_j) = R'_k - T_{bj} L_k - S_{bj}$
   - ➢ If k_{bj} is less than the total number of nodes in the sink list, go to 1.
5. Retrieve the best allocation during the whole process (allocation with the largest required time at the source). *End of process.*

Combinational Merging (4)





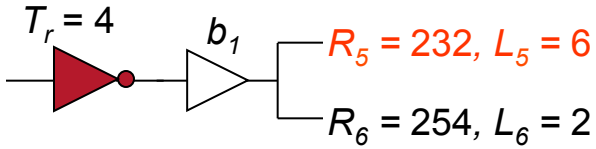
$L_{b1} = 2$
$S_{b1} = 42$
$T_{b1} = 4$



$L_{b2} = 3$
$S_{b2} = 48$
$T_{b2} = 2$

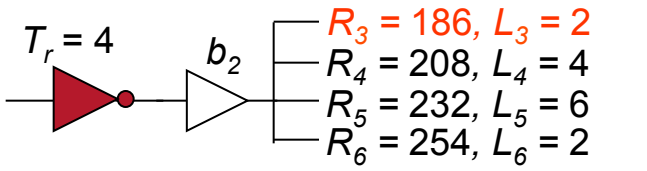
R	L	L'_k
138	3	29
155	6	26
160	6	20
186	2	14
208	4	12
232	6	8
254	2	2

$$n_{b1} = (T_{b1} L_{all} / T_r L_{b1})^{1/2}$$
$$= (116 / 8)^{1/2}$$
$$= 3.81$$
$$L_{all} / n_{b1} = 7.62$$
$$K_{b1} = 2, L'_2 = 8, R'_2 = 232$$



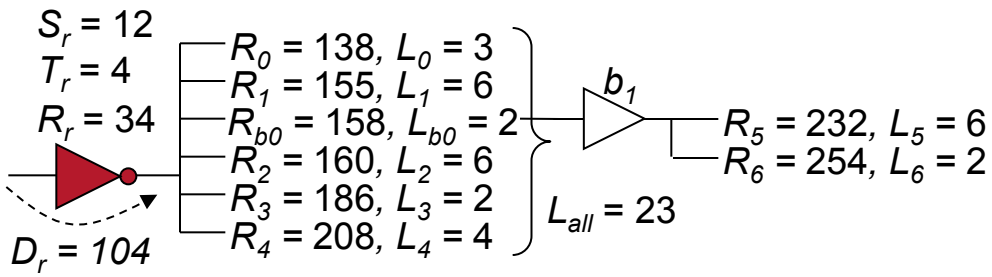
$$R(b_1) = R'_2 - T_{b1} * L'_2 - S_{b1} - T_r * L_{b1}$$
$$= 232 - 4 * 8 - 42 - 4 * 2$$
$$= 150$$

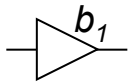
$$n_{b2} = (T_{b2} L_{all} / T_r L_{b2})^{1/2}$$
$$= (58 / 12)^{1/2}$$
$$= 2.20$$
$$L_{all} / n_{b2} = 13.19$$
$$K_{b2} = 4, L'_4 = 14, R'_4 = 186$$



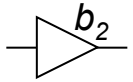
$$R(b_2) = R'_4 - T_{b2} * L'_4 - S_{b2} - T_r * L_{b2}$$
$$= 186 - 2 * 14 - 48 - 4 * 3$$
$$= 98$$

Combinational Merging (5)





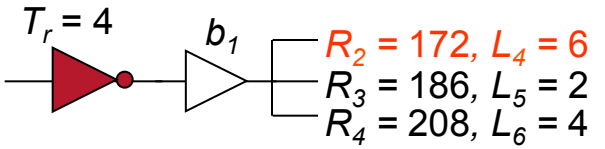
L_{b1} = 2
S_{b1} = 42
T_{b1} = 4



L_{b2} = 3
S_{b2} = 48
T_{b2} = 2

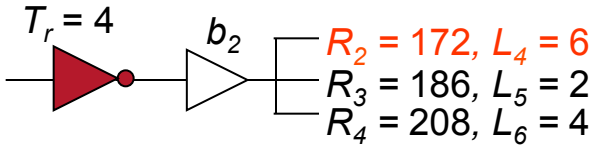
R	L	L'_k
138	3	23
155	6	20
158	2	14
160	6	12
186	2	6
208	4	4

$$n_{b1} = (T_{b1} L_{all} / T_r L_{b1})^{1/2}$$
$$= (92 / 8)^{1/2}$$
$$= 3.39$$
$$L_{all} / n_{b1} = 6.78$$
$$K_{b1} = 3,\ L'_3 = 12,\ R'_3 = 160$$



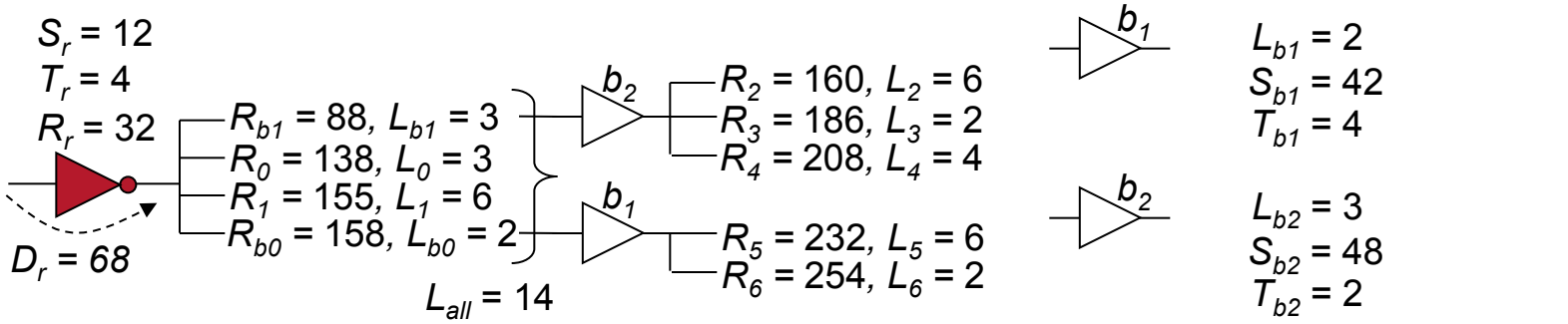
$$R(b_1) = R'_3 - T_{b1} * L'_3 - S_{b1} - T_r * L_{b1}$$
$$= 160 - 4 * 12 - 42 - 4 * 2$$
$$= 62$$

$$n_{b2} = (T_{b2} L_{all} / T_r L_{b2})^{1/2}$$
$$= (46 / 12)^{1/2}$$
$$= 1.96$$
$$L_{all} / n_{b2} = 11.75$$
$$K_{b2} = 3,\ L'_3 = 12,\ R'_3 = 160$$



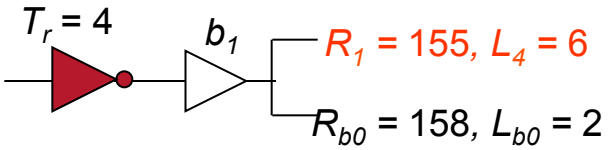
$$R(b_2) = R'_3 - T_{b2} * L'_3 - S_{b2} - T_r * L_{b2}$$
$$= 160 - 2 * 12 - 48 - 4 * 3$$
$$= 76$$

Combinational Merging (6)



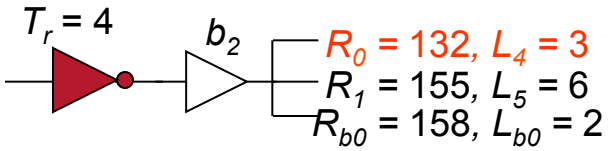
R	L	L'_k
100	3	14
138	3	11
155	6	8
158	2	2

$$n_{b1} = (T_{b1} L_{all} / T_r L_{b1})^{1/2}$$
$$= (56 / 8)^{1/2}$$
$$= 2.65$$
$$L_{all} / n_{b1} = 5.29$$
$$K_{b1} = 2,\ L'_2 = 8,\ R'_2 = 155$$



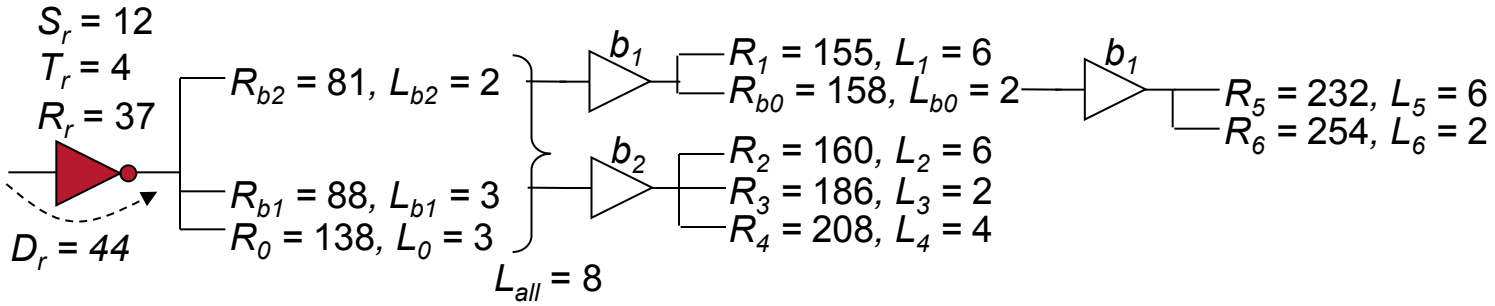
$$R(b_1) = R'_2 - T_{b1} * L'_2 - S_{b1} - T_r * L_{b1}$$
$$= 155 - 4 * 8 - 42 - 4 * 2$$
$$= 73$$

$$n_{b2} = (T_{b2} L_{all} / T_r L_{b2})^{1/2}$$
$$= (28 / 12)^{1/2}$$
$$= 1.53$$
$$L_{all} / n_{b2} = 9.17$$
$$K_{b2} = 3,\ L'_3 = 11,\ R'_3 = 138$$



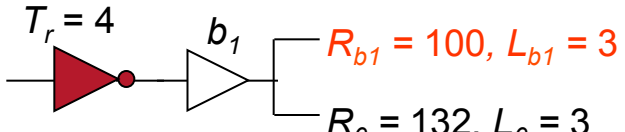
$$R(b_2) = R'_3 - T_{b2} * L'_3 - S_{b2} - T_r * L_{b2}$$
$$= 138 - 2 * 11 - 48 - 4 * 3$$
$$= 56$$

Combinational Merging (7)



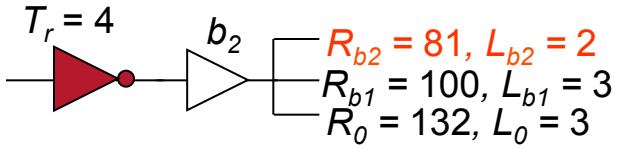
R	L	L'_k
81	2	8
88	3	6
138	3	3

$$n_{b1} = (T_{b1} L_{all} / T_r L_{b1})^{1/2}$$
$$= (32 / 8)^{1/2}$$
$$= 2.00$$
$$L_{all} / n_{b1} = 4.00$$
$$K_{b1} = 2,\ L'_2 = 6,\ R'_2 = 88$$



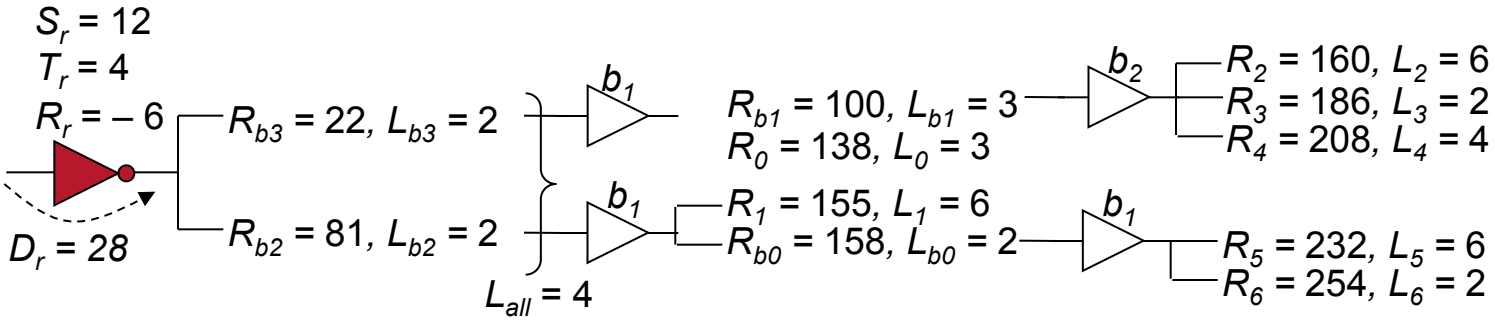
$$R(b_1) = R'_3 - T_{b1} * L'_3 - S_{b1} - T_r * L_{b1}$$
$$= 88 - 4 * 6 - 42 - 4 * 2$$
$$= 14$$

$$n_{b2} = (T_{b2} L_{all} / T_r L_{b2})^{1/2}$$
$$= (16 / 12)^{1/2}$$
$$= 1.15$$
$$L_{all} / n_{b2} = 6.93$$
$$K_{b2} = 3,\ L'_3 = 8,\ R'_3 = 81$$



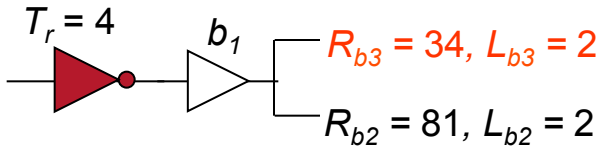
$$R(b_2) = R'_3 - T_{b2} * L'_3 - S_{b2} - T_r * L_{b2}$$
$$= 81 - 2 * 8 - 48 - 4 * 3$$
$$= 5$$

Combinational Merging (8)



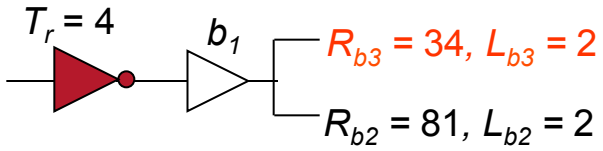
R	L	L'_k
22	2	4
81	2	2

$$n_{b1} = (T_{b1} L_{all} / T_r L_{b1})^{1/2}$$
$$= (16 / 8)^{1/2}$$
$$= 1.41$$
$$L_{all} / n_{b1} = 2.83$$
$$K_{b1} = 2, L'_2 = 4, R'_2 = 22$$



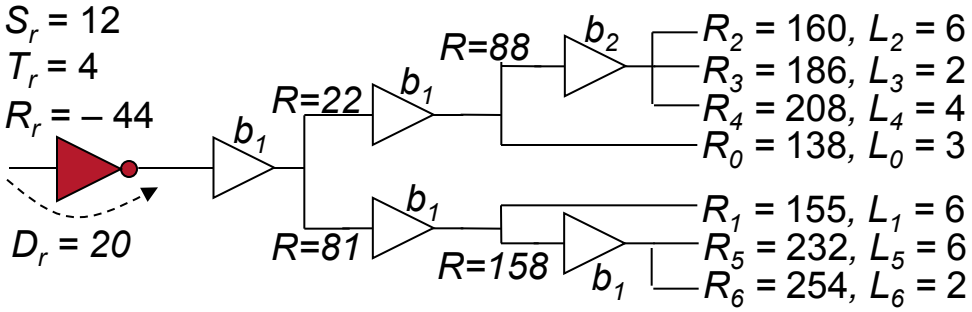
$$R(b_1) = R'_2 - T_{b1} * L'_2 - S_{b1} - T_r * L_{b1}$$
$$= 22 - 4 * 4 - 42 - 4 * 2$$
$$= -44$$

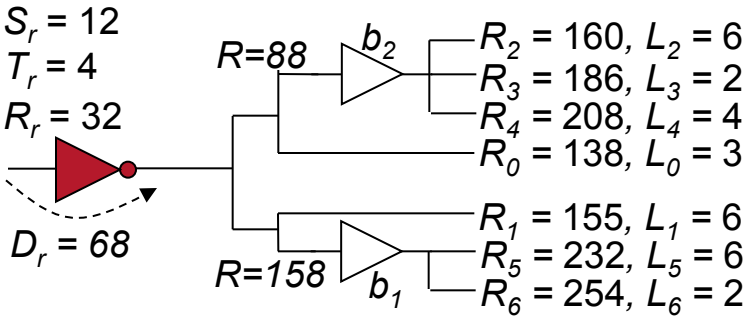
$$n_{b2} = (T_{b2} L_{all} / T_r L_{b2})^{1/2}$$
$$= (8 / 12)^{1/2}$$
$$= 0.82$$
$$L_{all} / n_{b2} = 4.90$$
$$K_{b2} = 2, L'_2 = 4, R'_2 = 22$$

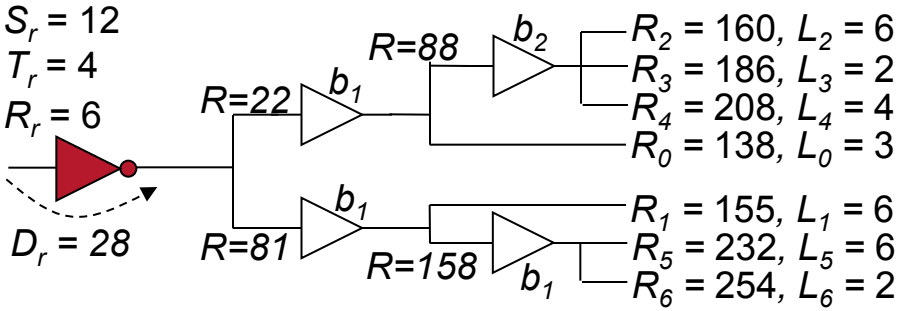


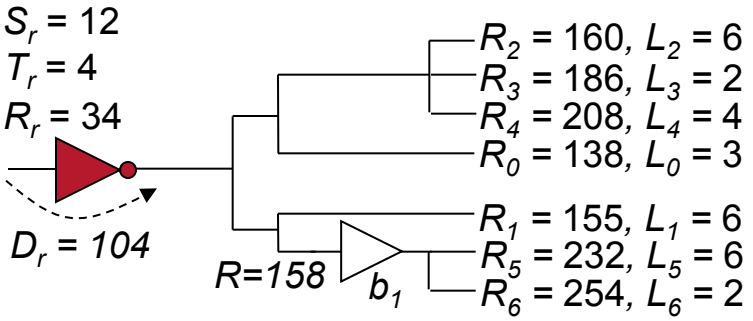
$$R(b_2) = R'_3 - T_{b2} * L'_3 - S_{b2} - T_r * L_{b2}$$
$$= 22 - 2 * 4 - 48 - 4 * 3$$
$$= -46$$

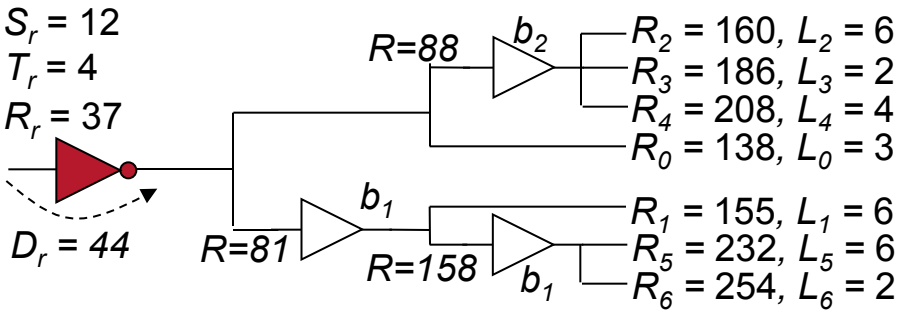
Combinational Merging (9)

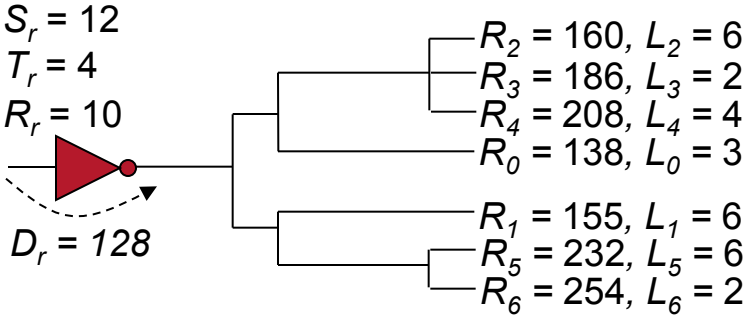




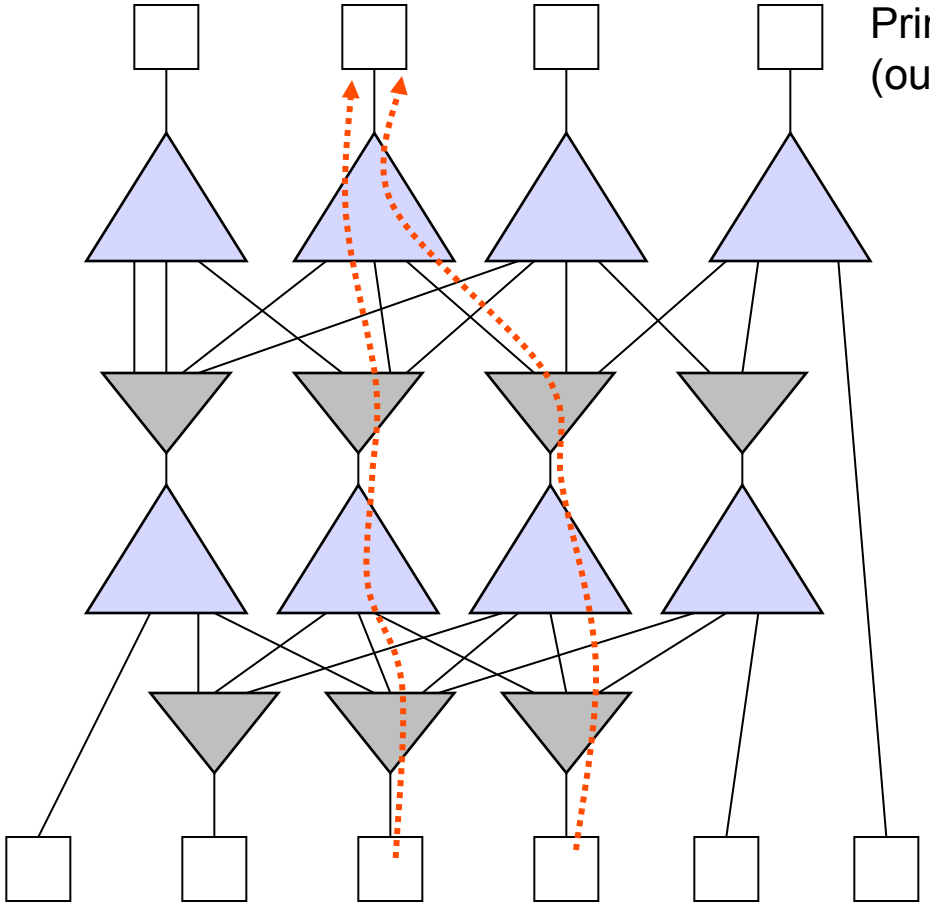








Timing-Driven Technology Mapping (9)



Primary outputs
(output pins, register inputs)

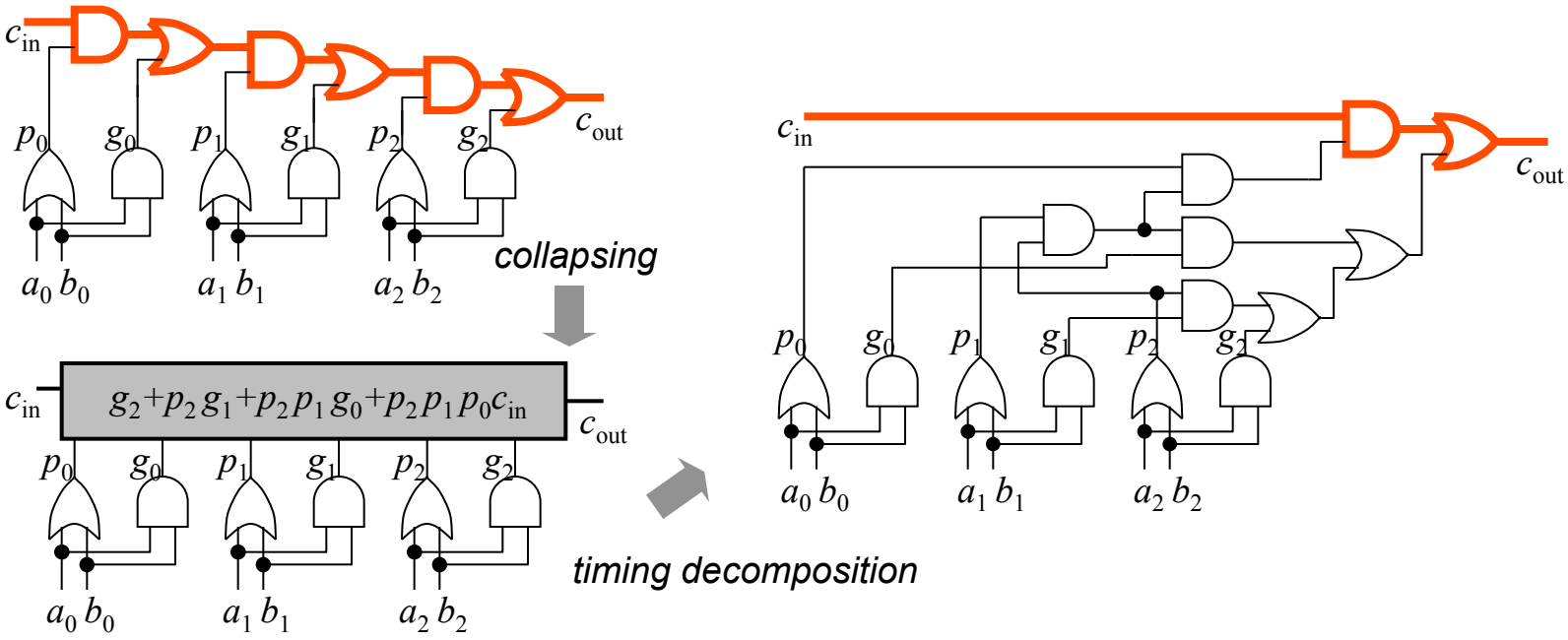
If iterative technology mapping cannot resolve timing violations
……

Boolean Network restructuring on paths with timing violations (negative slack times)

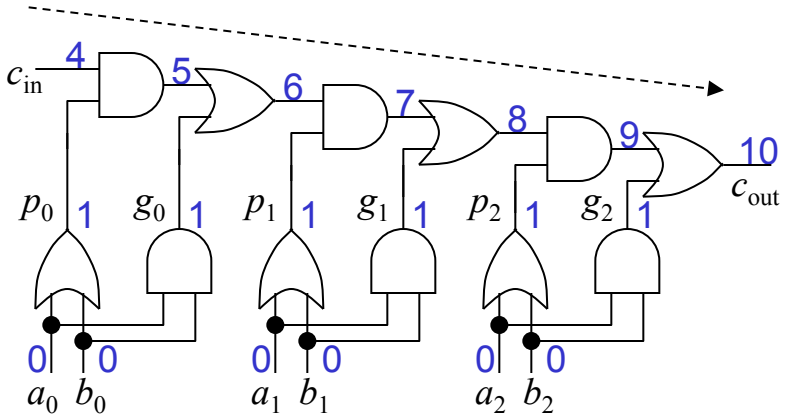
Primary inputs
(input pins, register outputs)

Boolean Network Restructuring

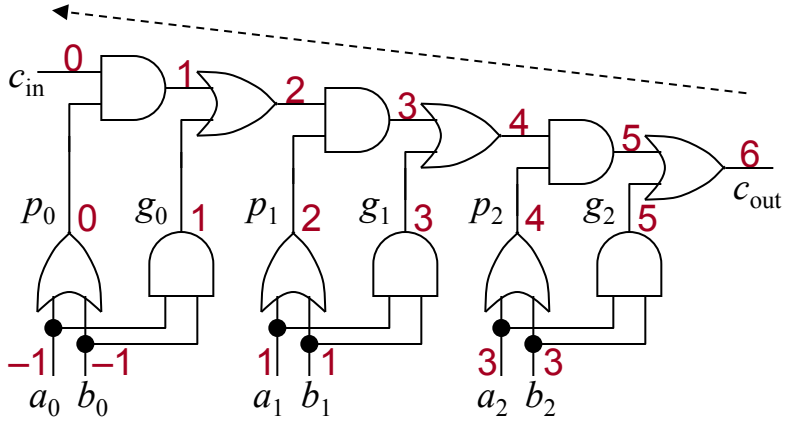
- There are some cases where delay-optimal technology mapping and fan-out optimization cannot satisfy the specified timing constraints
  → *need to change the circuit structure*
- On the paths with timing violations, collapse a part of the path into a large node (internally represented in sum-of-product form).
- Apply *timing decomposition* to the collapsed node to reduce the critical path delay.



Timing Analysis on Boolean Network

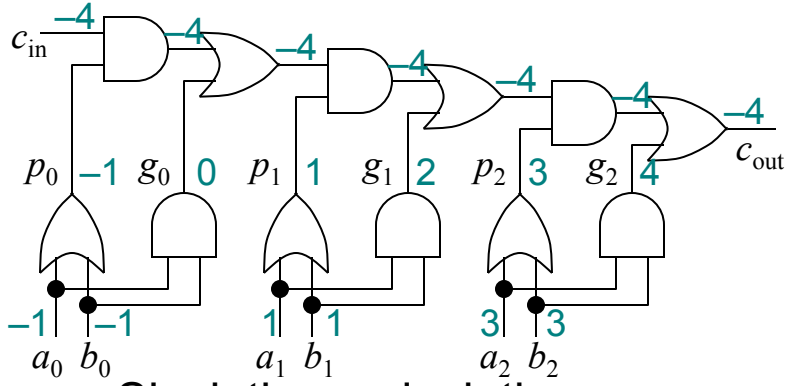


Arrival time calculation



Required time calculation

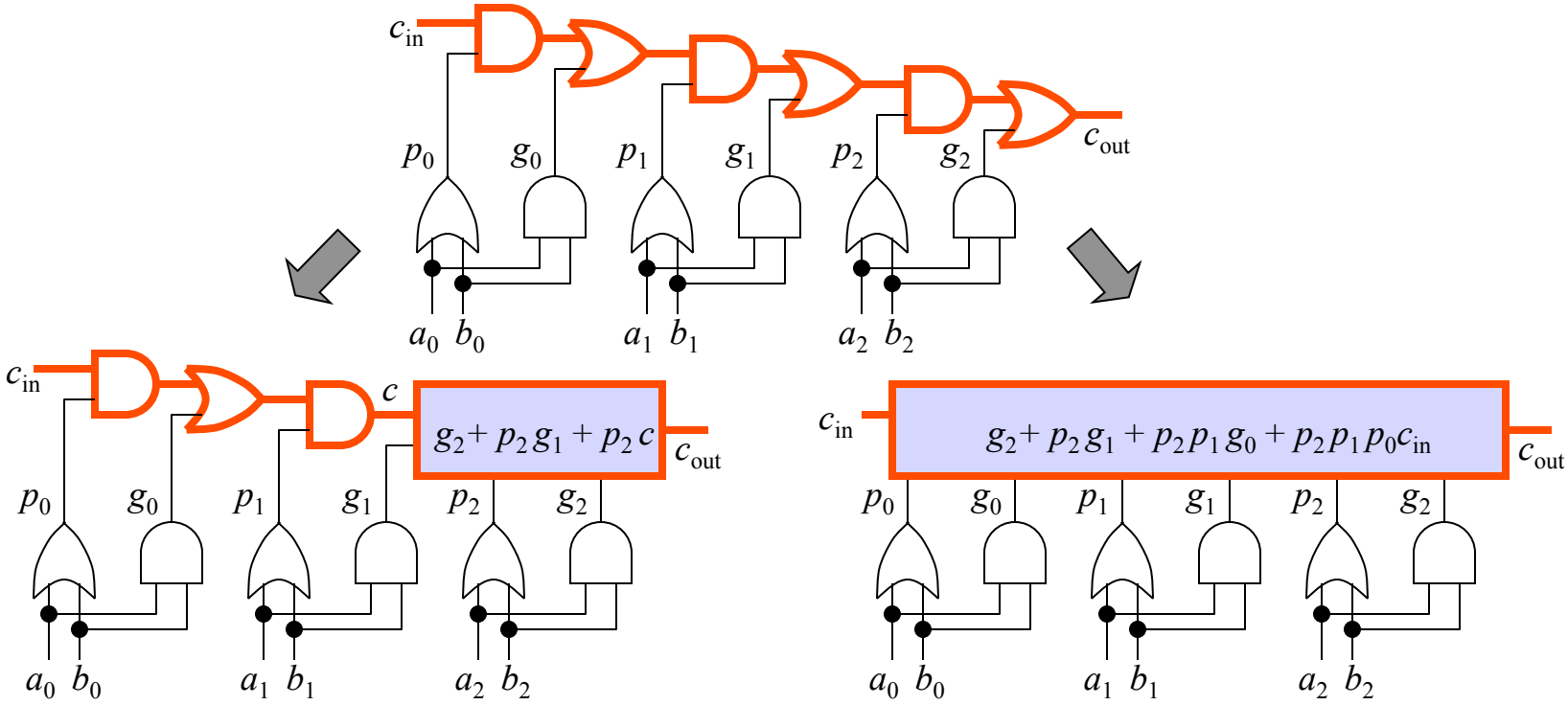
For simplicity, all gate delays are assumed to be 1 (output transition coefficient is assumed as 0)



Slack time calculation

Node Collapsing

- Select several nodes on the path with the least slack time and merge into one node.

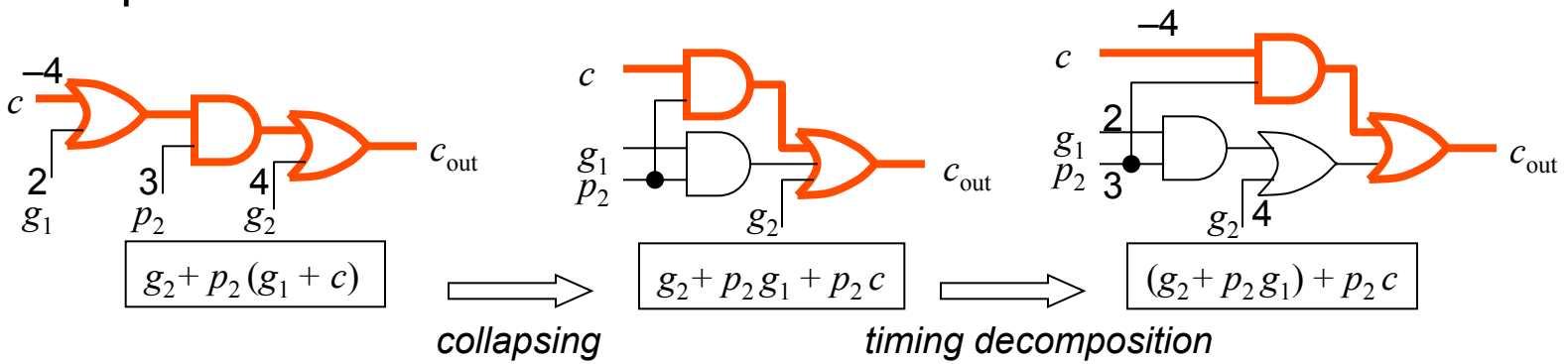


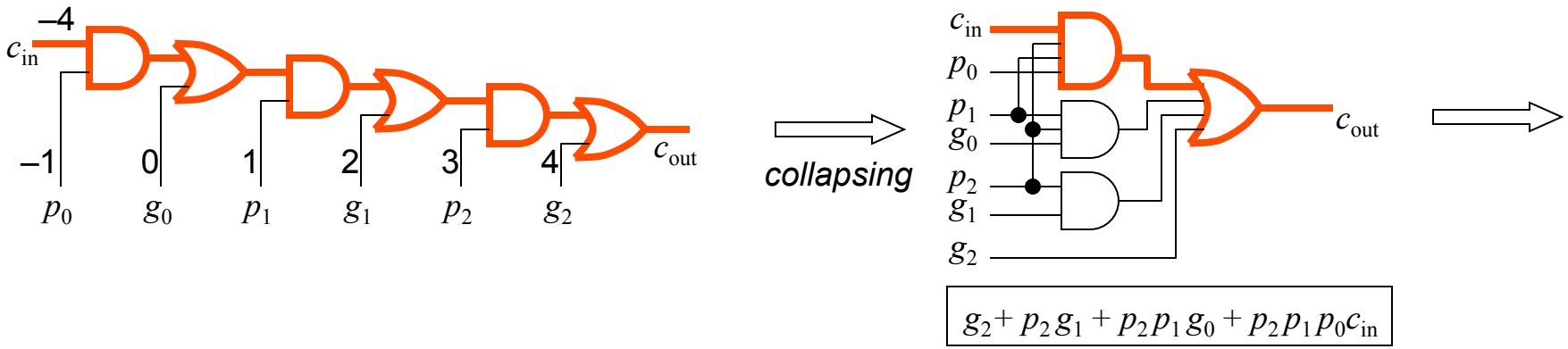
Collapse a part of the critical path

Collapse the entire critical path

Timing Decomposition (1)

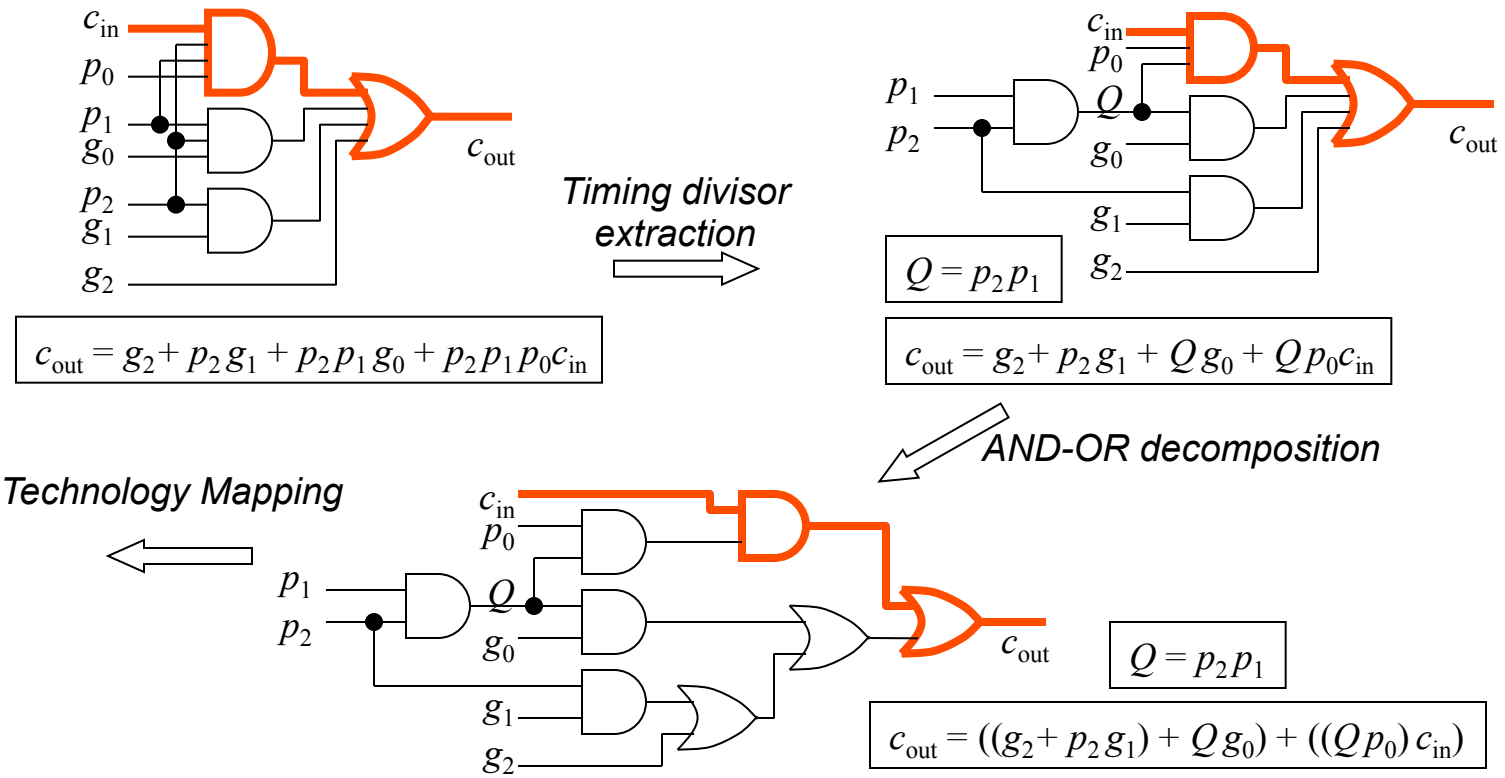
- Decompose the collapsed node into 2-input gates (AND, OR) so that the signals with smaller slack times becomes closer to the output.





Timing Decomposition (2)

- Bottom-up timing decomposition
  - Timing divisor extraction : Among the algebraic divisors which leads to circuit area reduction, extract the ones which is not on the critical path.
  - AND-OR decomposition : Decompose the division remainder into 2-input AND and OR gates. Put the signals on the critical path closer to the output.



Summary on Timing-Driven Technology Mapping and Boolean Network Restructuring

1. Each instance of delay-optimal technology mapping is dependent on other mapping results and fan-out tree optimizations. Therefore, several passes may be needed if timing violation occurs. Also, Boolean Network restructuring may be needed if the timing violation cannot be resolved in technology mapping phase (which will require another run of technology mapping).
2. There can be a number of possible strategies on the scheduling of each computation phase (technology mapping, fan-out optimization, area recovery, timing decomposition on Boolean Network) which will greatly affect the nature of the mapped circuits as well as the computation time.
3. In recent VLSI process technology, wiring delays are becoming a dominant factor in circuit speed compared to gate delays. Wiring delays cannot be accurately estimated until the physical layout of the circuit, and therefore very hard to anticipate during logic synthesis and technology mapping.