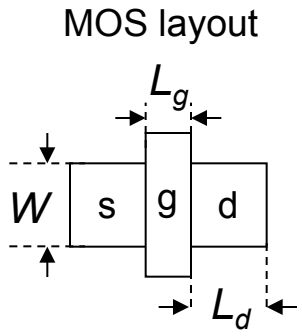
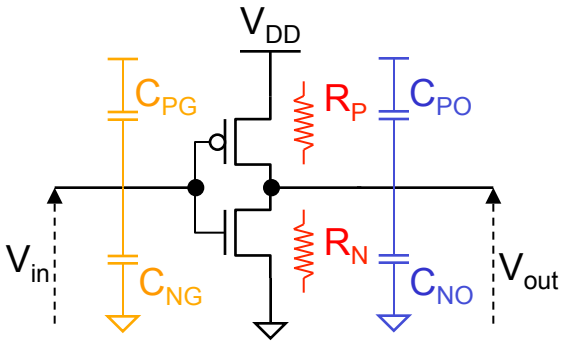
VLSI System Design
Part III : Technology Mapping (2)

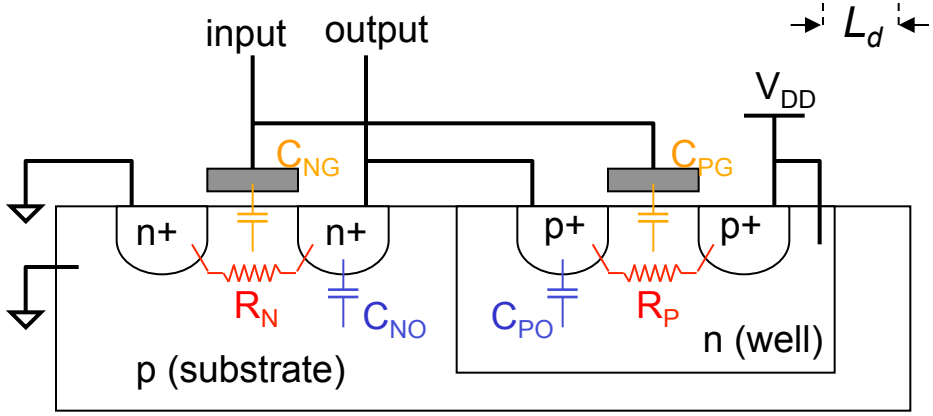
Lecturer : Tsuyoshi Isshiki
Dept. Communications and Computer Engineering,
Tokyo Institute of Technology
isshiki@vlsi.ce.titech.ac.jp

Simplified CMOS Delay Model (1)

1. INV device model
   - R_N, R_P : ON-resistance of nMOS / pMOS
     → Proportional to L_g / W
   - C_{NG}, C_{PG} : Gate capacitance of nMOS / pMOS
     → Proportional to $L_g \cdot W$
   - C_{NO}, C_{PO} : Drain capacitance of nMOS / pMOS
     → Proportional to $L_d \cdot W$ and $L_d + W$
   - ➢ *In reality, R_N, R_P, C_{NO}, C_{PO} are not constant, but dependent on the voltages at the gate and drain*







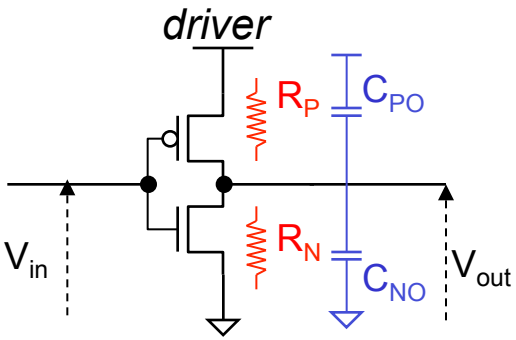
Simplified CMOS Delay Model (2)

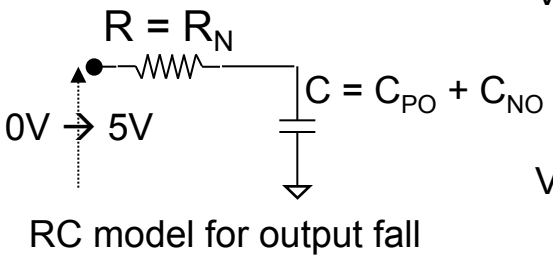
2. INV delay model (*no output load*)

→ ***internal delay, switching delay***

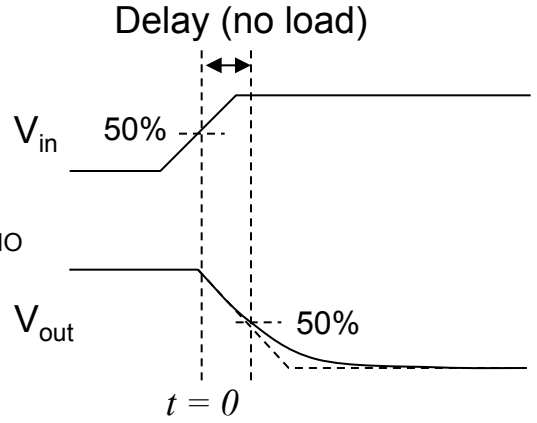
- When $V_{gs} > 0.5V_{DD}$: nMOS = ON, pMOS = OFF
- When $V_{gs} < 0.5V_{DD}$: nMOS = OFF, pMOS = ON
- Output rise time (input : 1 → 0) : $S_{rise} = kR_P (C_{PO} + C_{NO})$
- Output fall time (input : 0 → 1) : $S_{fall} = kR_N (C_{PO} + C_{NO})$

  (k : constant)





RC model for output fall



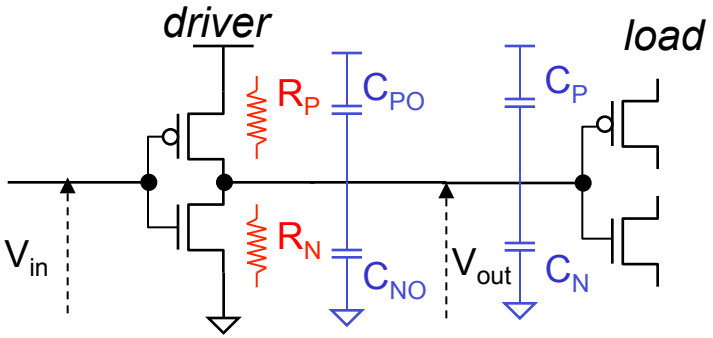
$$V_{out}(t) = V_{DD} e^{-t/RC}$$

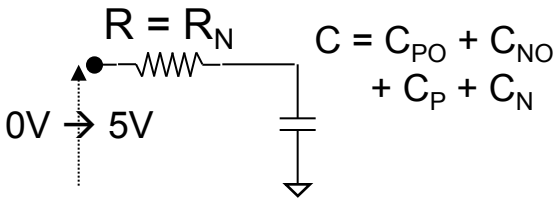
Simplified CMOS Delay Model (3)

3. INV delay model (*with output load*)
   - C_N, C_P : Gate capacitance of load nMOS / pMOS
   - Output rise time (input : 1 → 0) :

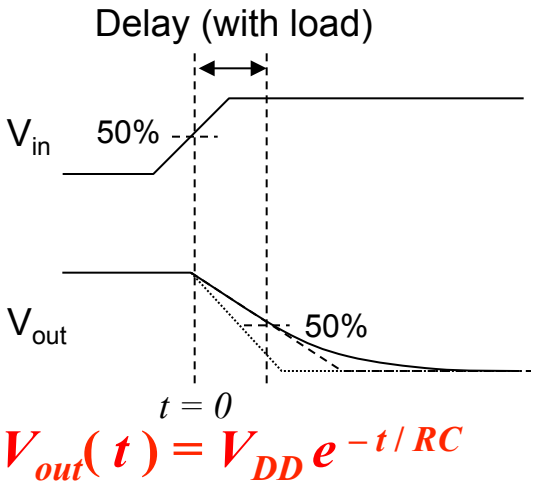
     $D_{rise} = kR_P (C_{PO} + C_{NO} + C_P + C_N) = S_{rise} + kR_P (C_P + C_N)$
   - Output fall time (input : 0 → 1) :

     $D_{fall} = kR_N (C_{PO} + C_{NO} + C_P + C_N) = S_{fall} + kR_N (C_P + C_N)$





RC model for output fall



Typical Delay Model for Standard Cells

$$\textit{Output rise delay} : D_R(i) = S_R(i) + T_R \cdot \textit{Load}$$

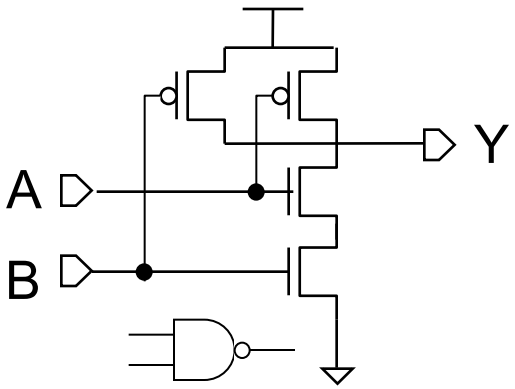
$$\textit{Output fall delay} : D_F(i) = S_F(i) + T_F \cdot \textit{Load}$$

$D_R(i)$, $D_F(i)$: total delay from input i to output

$S_R(i)$, $S_F(i)$: switching delay from input i to output

T_R, T_F : output transition coefficient (ns / pF)

Load : load connected to output (pF)

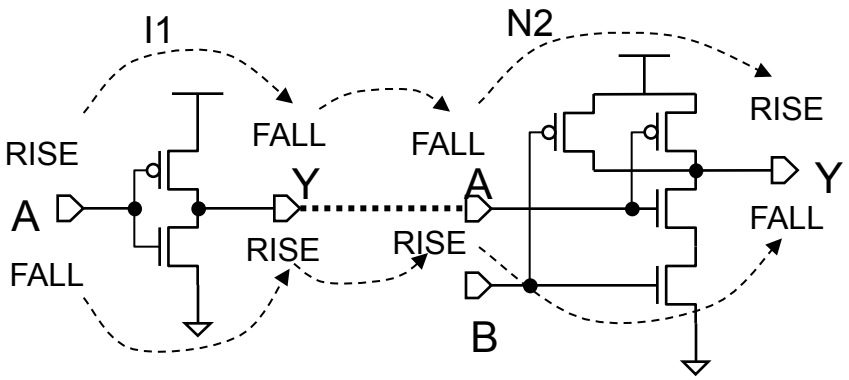


Load :
- A : 0.025pF
- B : 0.024pF

Internal delay = switching delay (S_R, S_F)
- A=>Y(rise) : 0.055 ns
- A=>Y(fall) : 0.051 ns
- B=>Y(rise) : 0.073 ns
- B=>Y(fall) : 0.060 ns

Output transition coefficient (T_R,T_F)
- Y(rise) : 1.532 ns/pF
- Y(fall) : 1.153 ns/pF

Path Delay Calculation



INV cell (I1)
Input Load :
L(A) = 0.025pF
Switching delay :
S_R(A) = 0.042 ns
S_F(A) = 0.039 ns
Output transition coef :
T_R = 1.534 ns/pF
T_F = 0.715 ns/pF

NAND2 cell (N2)
Input Load :
L(A) = 0.025pF
L(B) = 0.024pF
Switching delay :
S_R (A) = 0.055 ns
S_F (A) = 0.051 ns
S_R (B) = 0.073 ns
S_F (B) = 0.060 ns
Output transition coef :
T_R = 1.532 ns/pF
T_F = 1.153 ns/pF

Delay (I1.A→I1.Y(fall))
= I1.S_F(A) + I1.T_F * N2.L (A)
= 0.039 + 0.715 * 0.025 = 0.057
Delay (N2.A→N2.Y(rise))
= N2.S_R(A) + N2.T_R * N2.Load
= 0.055 + 1.532 * N2.Load
Delay (I1.A→N2.Y(rise))
= Delay(I1.A→I1.Y(fall))
+ Delay(N2.A→N2.Y(rise))
= 0.112 + 1.532 * N2.Load

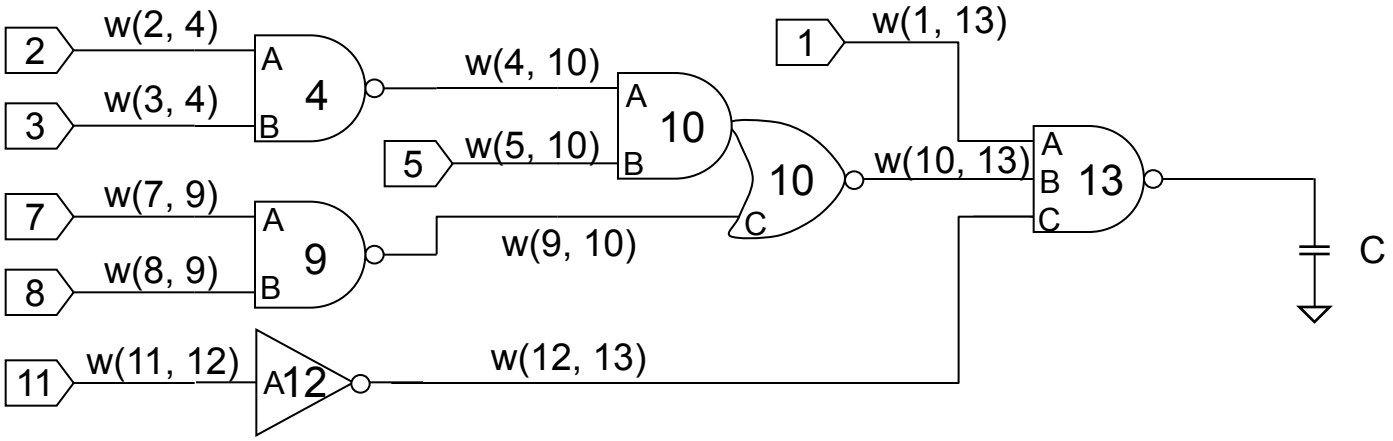
Delay (I1.A→I1.Y(rise))
= I1.S_R(A) + I1.T_R * N2.L (A)
= 0.042 + 1.534 * 0.025 = 0.080
Delay (N2.A→N2.Y(fall))
N2.S_F(A)+N2.T_F * N2.Load
= 0.051 + 1.153 * N2.Load
Delay (I1.A→N2.Y(fall))
= Delay(I1.A→I1.Y(rise))
+ Delay(N2.A→N2.Y(fall))
= 0.131 + 1.153 * N2.Load

Path Delays Graph Model (1)

- Total delay from output pin of node *i* to the output pin of node *j* $w(i, j) = j.S(pin_i) + j.T * j.Load$

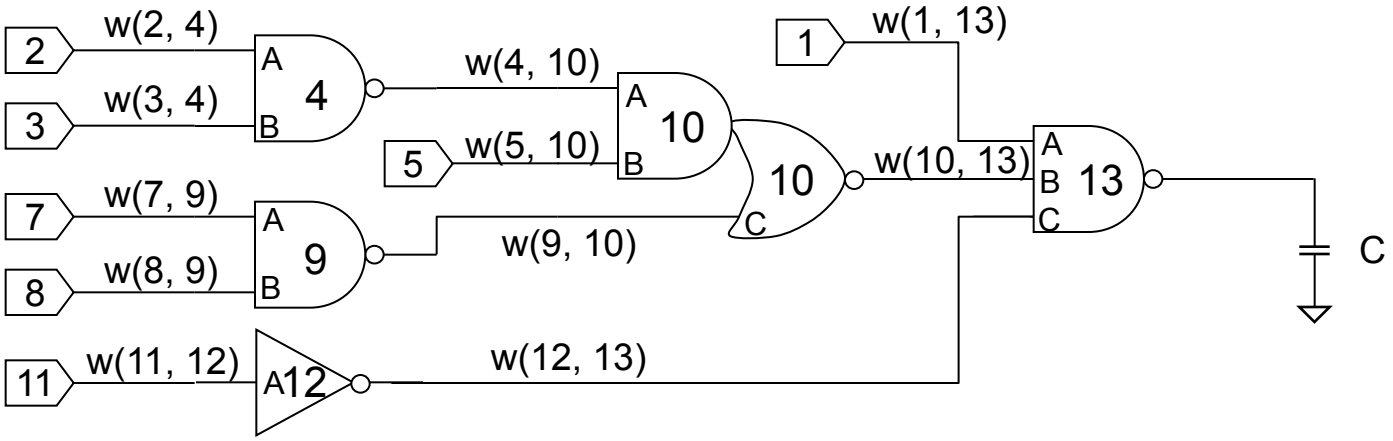
  (output of node *i* is connected to pin_i of node *j*)

- Describing separate delays for rise & fall transitions :
  - $w(i, j) = \{ w_R(i, j), w_F(i, j) \}$
    $= \{ j.S_R(pin_i) + j.T_R * j.Load , j.S_F(pin_i) + j.T_F * j.Load \}$



Path Delays Graph Model (2)

- Path delay from leaf node to root node
  - Delay(2→13) = Delay(2→10) + w(10, 13)
  - Delay(2→10) = Delay(2→4) + w(4, 10)
  - Delay(2→4) = w(2, 4)
- Path delay with separate rise / fall
  - $Delay_R(2 \rightarrow 13) = Delay_F(2 \rightarrow 10) + w_R(10, 13)$
  - $Delay_F(2 \rightarrow 10) = Delay_R(2 \rightarrow 4) + w_F(4, 10)$
  - $Delay_R(2 \rightarrow 4) = w_R(2, 4)$



Path Delays Graph Model (3)

- Path delay calculation (identical rise / fall)

$$\text{Delay}(i \rightarrow j) = \text{Delay}(i \rightarrow k) + w(k, j)$$

where k is the child of j which is on the path $i \rightarrow j$

- Path delay calculation (seperate rise / fall)
  - A *negative unate* input can only cause the opposite transition at the output (all inputs of INV, NAND, NOR, AOI are negative unate inputs)

$$\text{Delay}_R(i \rightarrow j) = \text{Delay}_F(i \rightarrow k) + w_R(k, j)$$

$$\text{Delay}_F(i \rightarrow j) = \text{Delay}_R(i \rightarrow k) + w_F(k, j)$$

  - A *positive unate* input can only cause the same transition at the output (all inputs of AND, OR, AO, 0-1 inputs of MUX are positive unate inputs)

$$\text{Delay}_R(i \rightarrow j) = \text{Delay}_R(i \rightarrow k) + w_R(k, j)$$

$$\text{Delay}_F(i \rightarrow j) = \text{Delay}_F(i \rightarrow k) + w_F(k, j)$$

  - A *non-unate* input can cause both same and opposite transition on the output (all inputs of XOR, selector input of MUX)

$$\text{Delay}_R(i \rightarrow j) = MAX\{ \text{Delay}_R(i \rightarrow k), \text{Delay}_F(i \rightarrow k) \} + w_R(k, j)$$

$$\text{Delay}_F(i \rightarrow j) = MAX\{ \text{Delay}_R(i \rightarrow k), \text{Delay}_F(i \rightarrow k) \} + w_F(k, j)$$

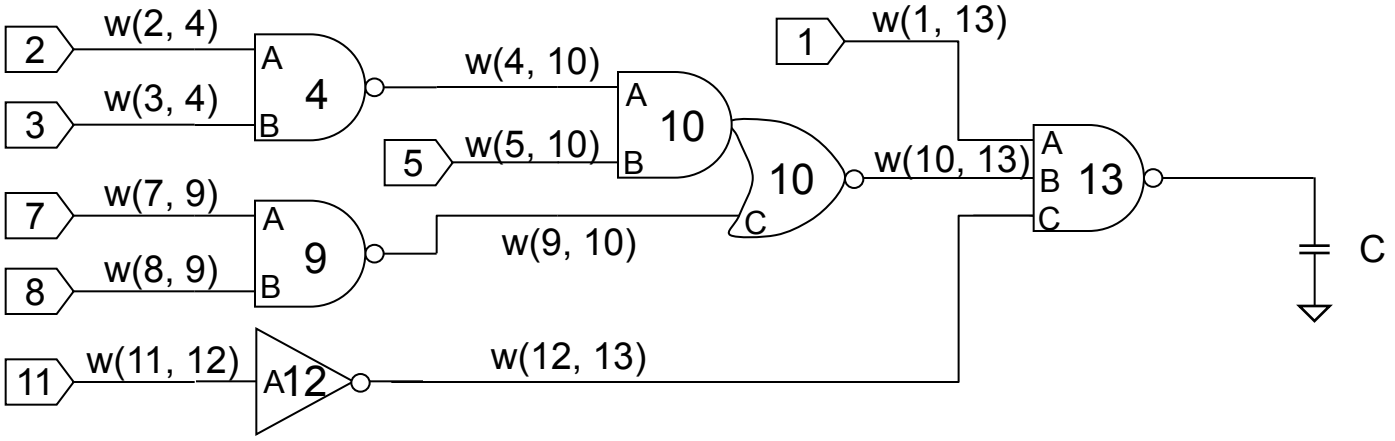
Arrival Time Calculation

- Arrival time : time when the last output transition occurs
  - Leaf inputs are given a specified arrival time
  - Arrival($i \rightarrow j$) : arrival time of leaf input i propagating to node j

    $$\text{Arrival}(i \rightarrow j) = \text{Delay}(i \rightarrow j) + \text{Arrival}(i)$$
  - Arrival(j) : latest arrival time of all leaf inputs propagating to node j

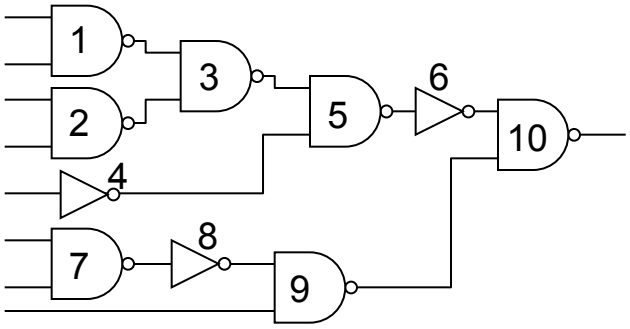
    $$\text{Arrival}(j) = MAX\{\text{Arrival}(i \rightarrow j) \mid i : \text{leaf input reachable to } j\}$$
    $$= MAX\{\text{Arrival}(k) + w(k, j) \mid k : \text{child of } j\}$$

EX : Arrival(13) = *MAX* { Arrival(1) + w(1, 13), Arrival(10) + w(10, 13), Arrival(12) + w(12, 13) }



Delay-Optimal Technology Mapping

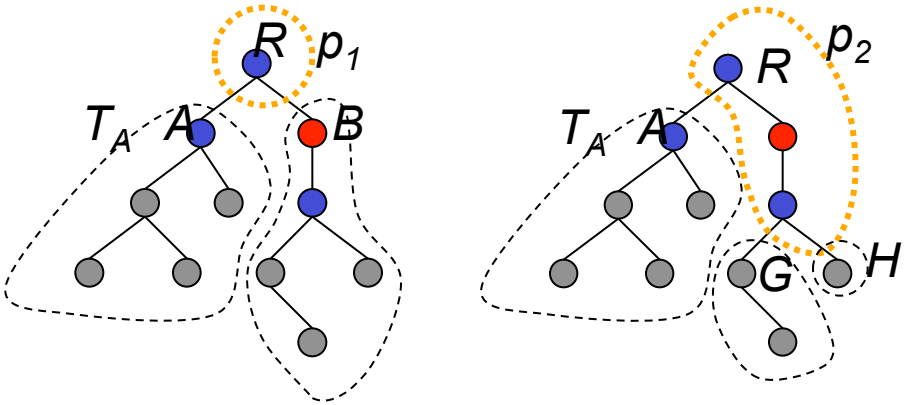
- Problem : Give a covering for the target NAND2-tree so that the latest arrival time at the root node is minimum.
- Can the dynamic programming technique used in area-optimal technology mapping be used to guarantee a delay-optimal solution??



Target NAND2-tree

Delay Cost Calculation (1)

- $C_{opt}(i)$: minimum arriving time at node i on the optimal covering of the subtree rooted at i
- $C_{map}(i, p_j)$: minimum arriving time at node i on the optimal covering of the subtree rooted at i when p_j is used to cover i
  → $C_{opt}(i) = MIN\{ C_{map}(i, p_j) \}$
- $C(k, p_j)$: delay of cell pattern p_j from the input connected to node k



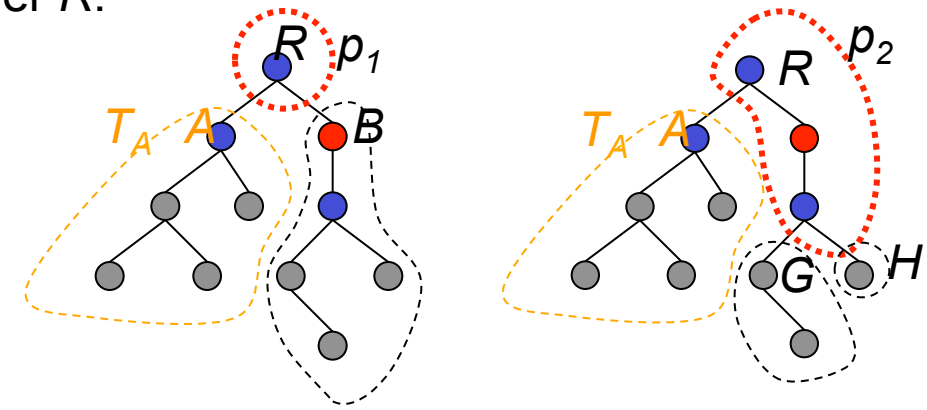
$C_{map}(R, p_1) = MAX\{ C_{opt}(A) + C(A, p_1), C_{opt}(B) + C(B, p_1) \}$
$C_{map}(R, p_2) = MAX\{ C_{opt}(A) + C(A, p_2), C_{opt}(G) + C(G, p_2), C_{opt}(H) + C(H, p_2) \}$
$C_{opt}(R) = MIN\{C_{map}(R, p_1) + C_{map}(R, p_2)\}$

Delay Cost Calculation (2)

$C(k, p_j)$: delay of cell pattern p_j from the input connected to node k

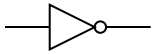
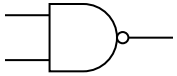
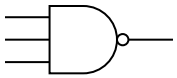
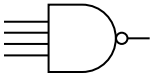
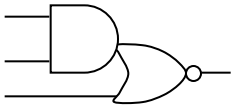
→ *THIS COST CANNOT BE DETERMINED UNLESS THE OUTPUT LOAD TO CELL p_j IS KNOWN*

- Below : optimal covering cost $C_{opt}(A)$ is dependent on the patterns p_1 and p_2 to cover R.



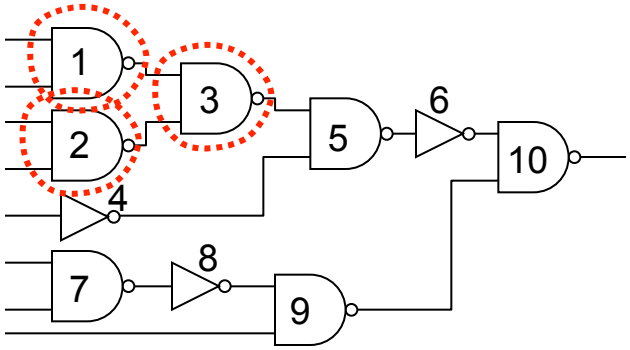
- But – <u>if all the loads (gate capacitances) are same</u>, then the cell delay can be determined. → *Can use dynamic programming for delay-optimal covering*
- *<u>Strategy : Compute the cell delays for several candidates of output loads</u>*

Cell Library Example

symbol	cell name	area	gate load	switching delay	output transition coef
	INV	2	3	12	4
	INVP	3	6	12	2
	NAND2	3	3	25	6
	NAND2P	5	6	25	3
	NAND3	4	2	40	8
	NAND3P	7	4	40	4
	NAND4	5	2	60	8
	NAND4P	9	4	60	4
	AOI21	4	3	60	8
	AOI21P	7	6	60	4

- For simplicity, all input pins have the same load and switch delay for each cell in this library
- In this library, candidates for output load values are 2, 3, 4, 6
- For more complex library, limit the number of candidates and quantizing each cell gate load to the candidate load values

Delay-Optimal Tree Covering (1)

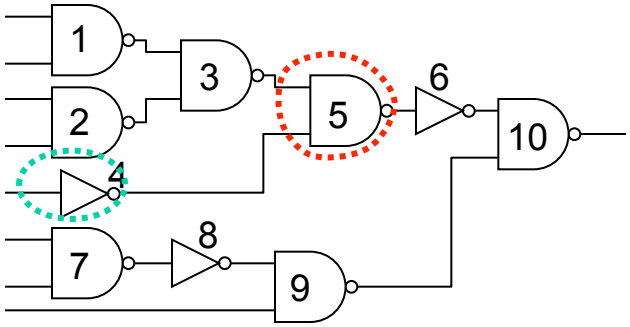


- *Here, assume arrival times are 0 at all leaf inputs*

Arrival Time for different output loads

sub-tree	cell	cell area	inputs	gate load	switch delay	output trans.	0-load	Output loads				total area
								2	3	4	6	
1	NAND2	3	-	3	25	6	25	37	43	49	61	3
	NAND2P	5	-	6	25	3	25	31	34	37	43	5
2	NAND2	3	-	3	25	6	25	37	43	49	61	3
	NAND2P	5	-	6	25	3	25	31	34	37	43	5
3	NAND2	3	1, 2	3	25	6	59	71	77	83	95	13
	NAND2P	5	1, 2	6	25	3	68	74	77	80	86	15

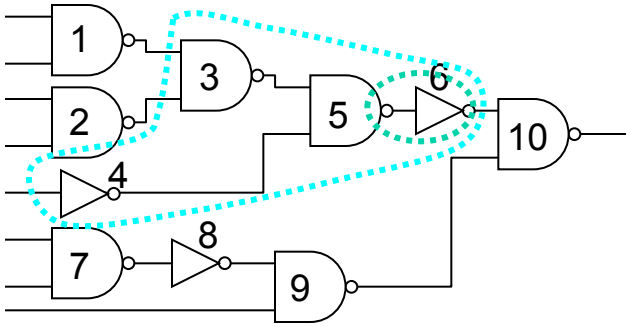
Delay-Optimal Tree Covering (2)



sub-tree	Output loads				total area
	2	3	4	6	
1	31	34	37	43	5
2	31	34	37	43	5
3	71	77	80	86	13 / 15

sub-tree	cell	cell area	inputs	gate load	switch delay	output trans.	0-load	Output loads				total area
								2	3	4	6	
4	INV	2	-	3	12	4	12	20	24	28	36	2
	INVP	3	-	6	12	2	12	16	18	20	24	3
5	NAND2	3	3, 4	3	25	6	102	114	120	126	138	19
	NAND2P	5	3, 4	6	25	3	111	117	120	123	129	23

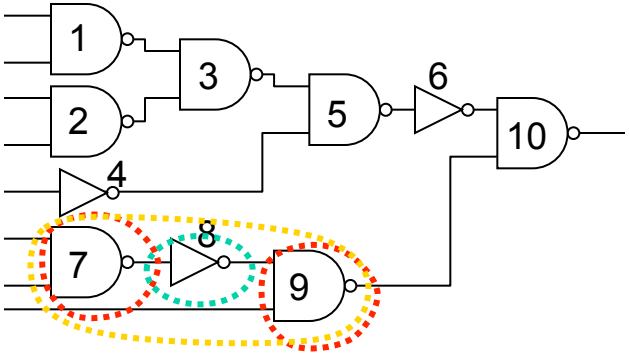
Delay-Optimal Tree Covering (3)



sub-tree	Output loads				total area
	2	3	4	6	
1	31	34	37	43	5
2	31	34	37	43	5
5	114	120	123	129	19 / 23

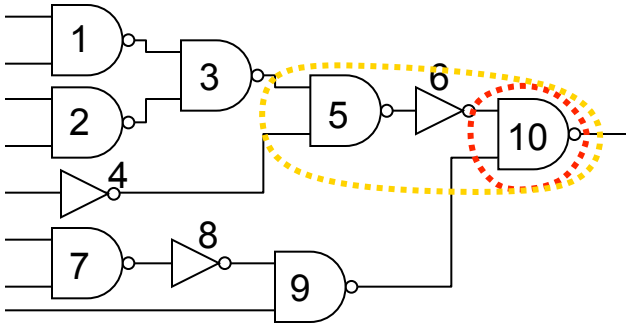
sub-tree	cell	cell area	inputs	gate load	switch delay	output trans.	0-load	Output loads				total area
								2	3	4	6	
6	INV	2	5	3	12	4	132	140	144	148	156	21
	INVP	3	5	6	12	2	141	145	147	149	153	26
	AOI21	4	1, 2	3	60	8	94	110	118	126	142	14
	AOI21P	7	1, 2	6	60	4	103	111	115	119	127	17

Delay-Optimal Tree Covering (4)



sub-tree	cell	cell area	inputs	gate load	switch delay	output trans.	0-load	Output loads				total area
								2	3	4	6	
7	NAND2	3	-	3	25	6	25	37	43	49	61	3
	NAND2P	5	-	6	25	3	25	31	34	37	43	5
8	INV	2	7	3	12	4	46	54	58	62	70	7
	INVP	3	7	6	12	2	55	59	61	63	67	8
9	NAND2	3	8	3	25	6	83	95	101	107	119	10
	NAND2P	5	8	6	25	3	92	98	101	104	110	13
	NAND3	4	-	2	40	8	40	56	64	72	88	4
	NAND3P	7	-	4	40	4	40	48	52	56	64	7

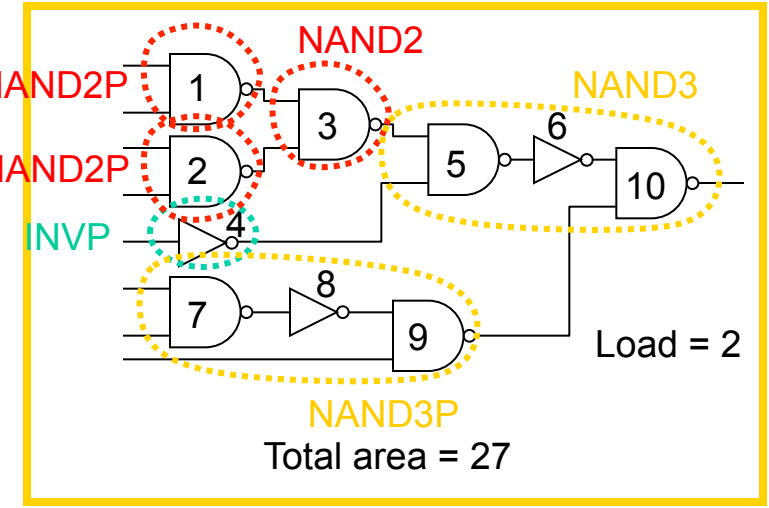
Delay-Optimal Tree Covering (5)

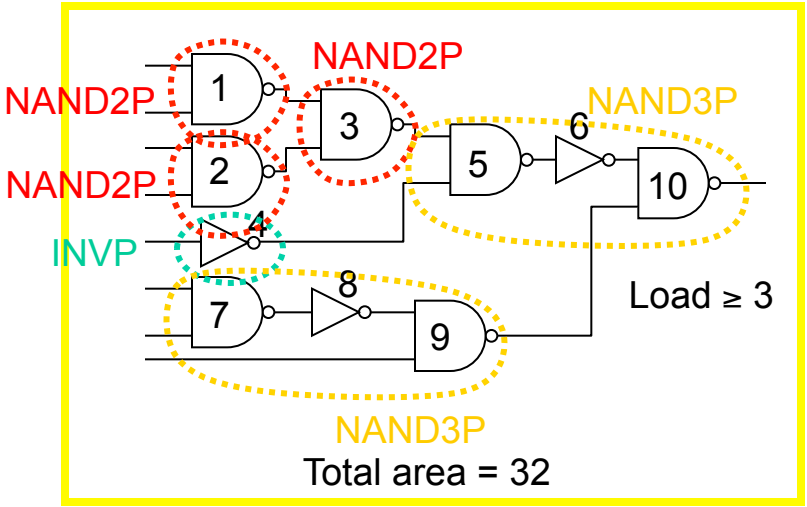


sub-tree	Output loads				total area
	2	3	4	6	
3	71	77	80	86	13 / 15
4	16	18	20	24	3
6	110	115	119	127	14 / 17
9	48	52	56	64	7

sub-tree	cell	cell area	inputs	gate load	switch delay	output trans.	0-load	Output loads				total area
								2	3	4	6	
10	NAND2	3	6, 9	3	25	6	140	152	158	164	176	27
	NAND2P	5	6, 9	6	25	3	152	158	161	164	170	29
	NAND3	4	3, 4, 9	2	40	8	111	127	135	143	159	27
	NAND3P	7	3, 4, 9	4	40	4	120	128	132	136	144	32

Retrieving the Delay-Optimal Covering Solutions





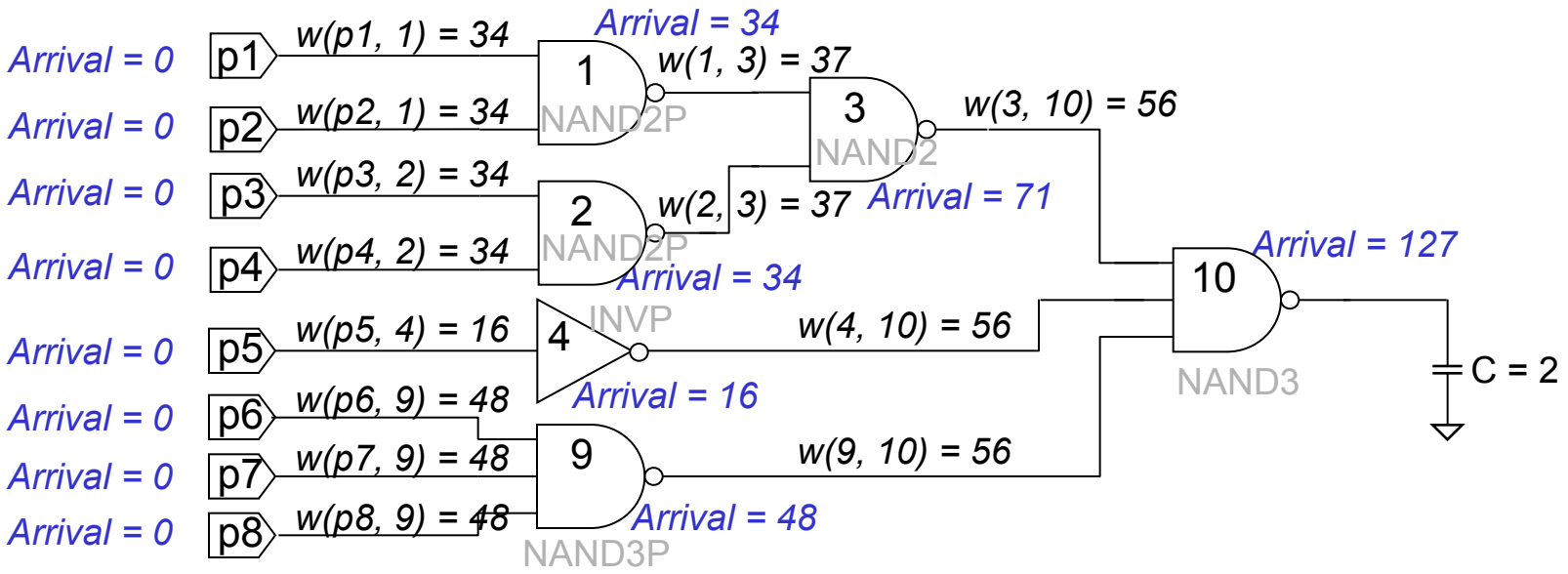
sub-tree	cell	inputs	gate load	Output loads				total area
				2	3	4	6	
1	NAND2	-	3	37	43	49	61	3
	NAND2P	-	6	31	34	37	43	5
2	NAND2	-	3	37	43	49	61	3
	NAND2P	-	6	31	34	37	43	5
3	NAND2	1, 2	3	71	77	83	95	13
	NAND2P	1, 2	6	74	77	80	86	15
4	INV	-	3	20	24	28	36	2
	INVP	-	6	16	18	20	24	3
9	NAND2	8	3	95	101	107	119	3
	NAND2P	8	6	98	101	104	110	5
	NAND3	-	2	56	64	72	88	4
	NAND3P	-	4	48	52	56	64	7
10	NAND2	6, 9	3	152	158	164	176	27
	NAND2P	6, 9	6	158	161	164	170	29
	NAND3	3, 4, 9	2	127	135	143	159	27
	NAND3P	3, 4, 9	4	128	132	136	144	32

Slack Time (1)

- Arrival time : time when the last output transition occurs
  - Leaf inputs are given a specified arrival time

    (Here, assume the arrival times are 0 for all leaf inputs)

    Arrival(j) = *MAX* { Arrival(k) + w(k, j) | k : child of j }



Slack Time (2)

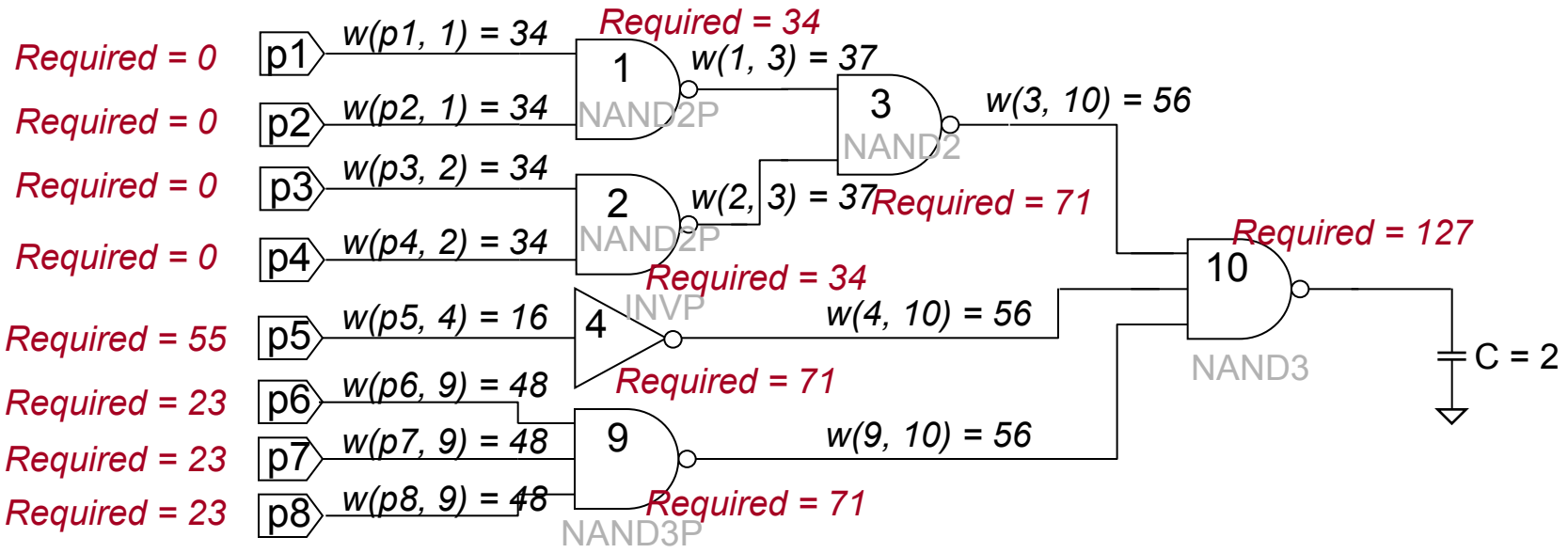
- Required time : time required to satisfy the given timing constraints

  Required(k) = Required(j) - w(k, j)

  (node j is the parent of node k)

  - Root output is given a specified required time and load value.
  - For non-tree graph (output can have more than 1 fan-out) :

    Required(k) = *MIN* {Required(j) - w(k, j) | j : destination of output of k }

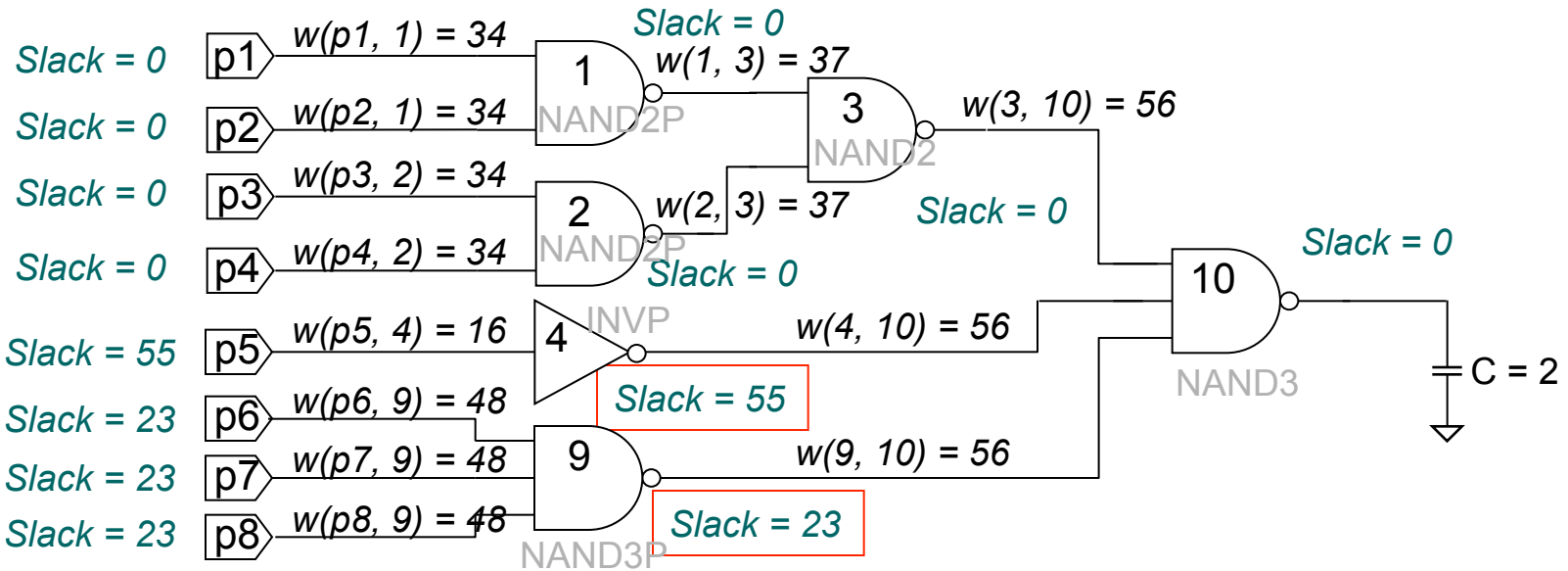


Slack Time (3)

- Slack time :

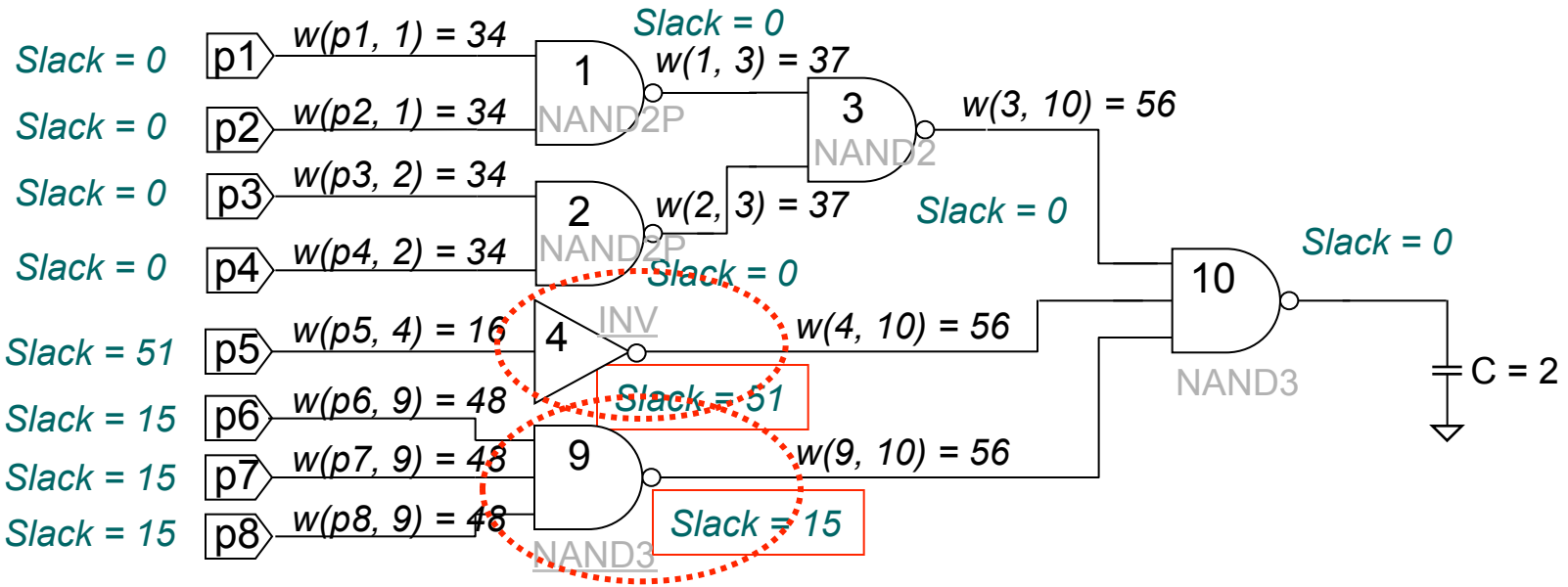
  Slack(*j*) = Required(*j*) – Arrival(*j*)

- Nodes with 0 slack times is said to be on a critical path (critical nodes)
- *Nodes with positive slack times (non-critical nodes) can be changed into slower cells in order to reduce circuit area*



Area Recovery

- *Change the covering such that the circuit area is reduced but the arrival time at the root is not changed (still delay optimal).*
- *Below :*
  - *Change INVP to INV at node 4 (save 1 unit area)*
  - *Change NAND3P to NANDP at node 9 (save 3 unit area)*



Summary of Delay-Optimal Tree Covering (1)

1. Cell delays are dependent on output loads, therefore the Principle of Optimality does not apply directly.
   - Compute the delay-optimal covering assuming a number of candidate load values (each solution itself retains the Principle of Optimality for that particular load value)
   - In general, there may be too many load value candidates. In such cases, limit the number of candidates, and for all cells in the library, quantize the gate loads to one of those candidate values.
2. Delay-optimal area-minimum tree covering can be obtained by applying area recovery on the non-critical nodes in the delay-optimal tree covering.

Summary of Delay-Optimal Tree Covering (2)

2. In real designs, there are a number of conditions and constraints which makes the delay-optimal tree covering problem more complex :
   - A) Arriving times at the leaf inputs are not necessarily 0, but may depend on the covering results on other trees. Or worse, they may depend on the load connected to the leaf inputs (which would be the general case)
   - B) Switching delays are generally different for rise and fall transitions → need to examine the transition polarity
   - C) Switching delays (internal delays) and gate load values are generally different among the input pins. Examining all valid permutations of input pins will lead to faster circuit but can become very time consuming.
   - D) Not all trees need to be delay-optimal, merely needs to satisfy some delay constraint.