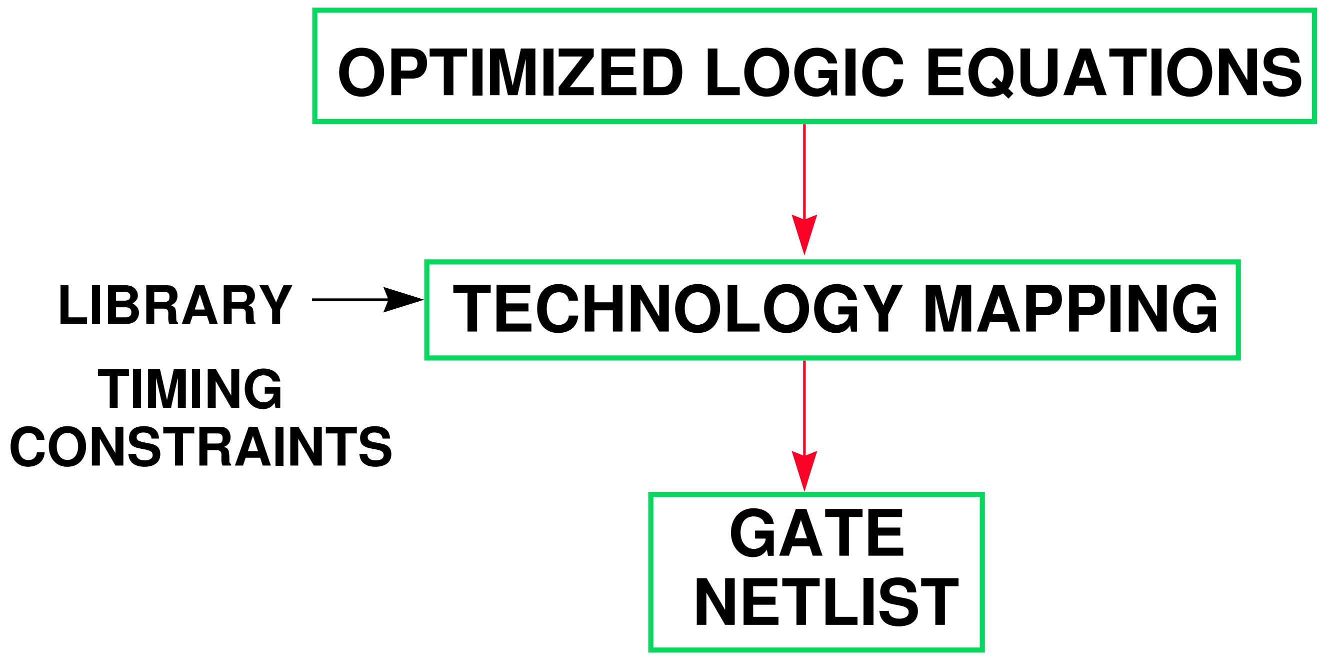
Tech.-Dependent Optimization

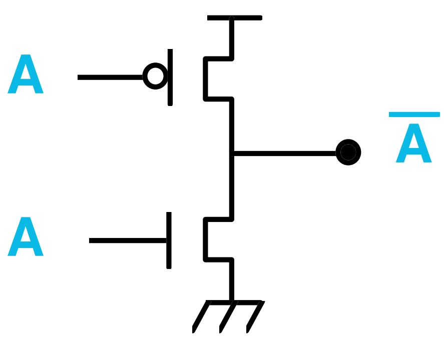


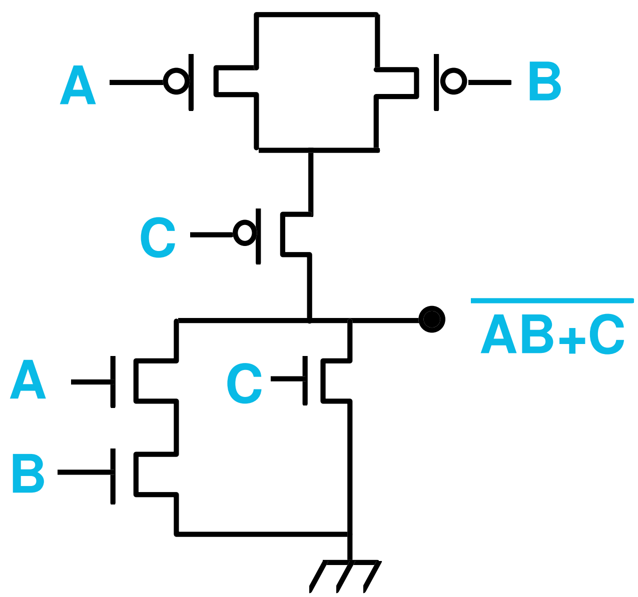
Area, delay and power dissipation cost functions

“Closed Book” Technologies

A standard cell technology or library is typically restricted to a few tens of gates e.g., MSU library: 31 cells

Gates may be NAND, NOR, NOT, AOIs.





Mapping via DAG Covering

Represent network in canonical form
$\Rightarrow$ subject DAG

Represent each library gate with canonical forms for the logic function
$\Rightarrow$ primitive DAGs

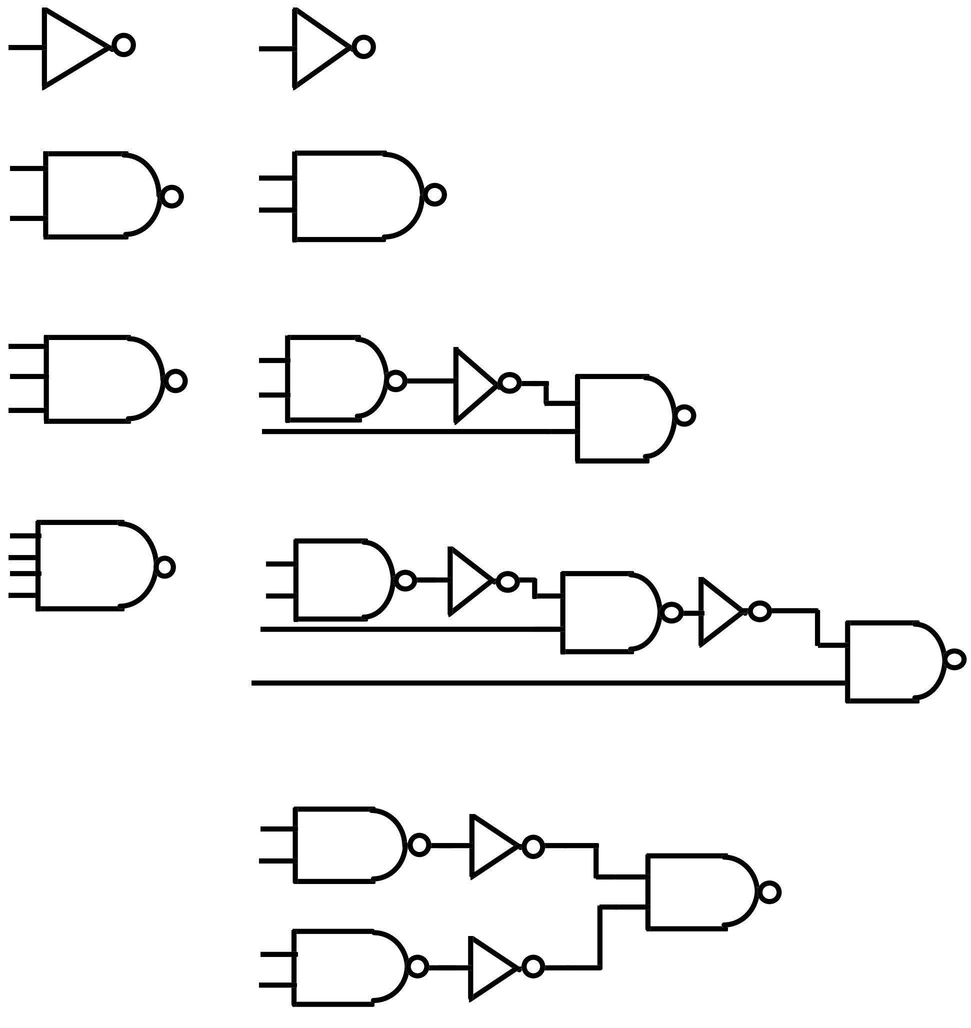
Each primitive DAG has a cost

Goal: Find a minimum cost covering of the subject DAG by the primitive DAGs

Canonical form: 2-input NAND gates and inverters

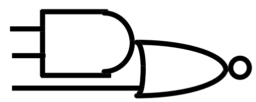
Sample Library

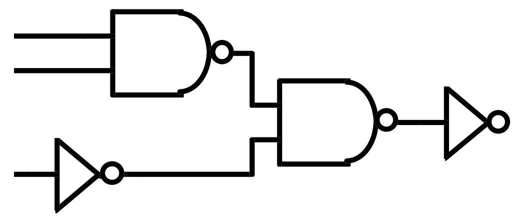
INVERTER	2
NAND2	3
NAND3	4
NAND4	5



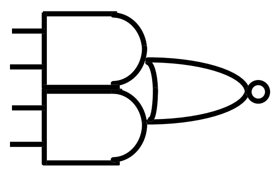
Sample Library - 2

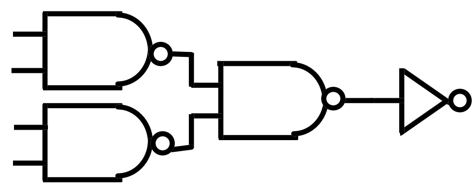
AOI21 4



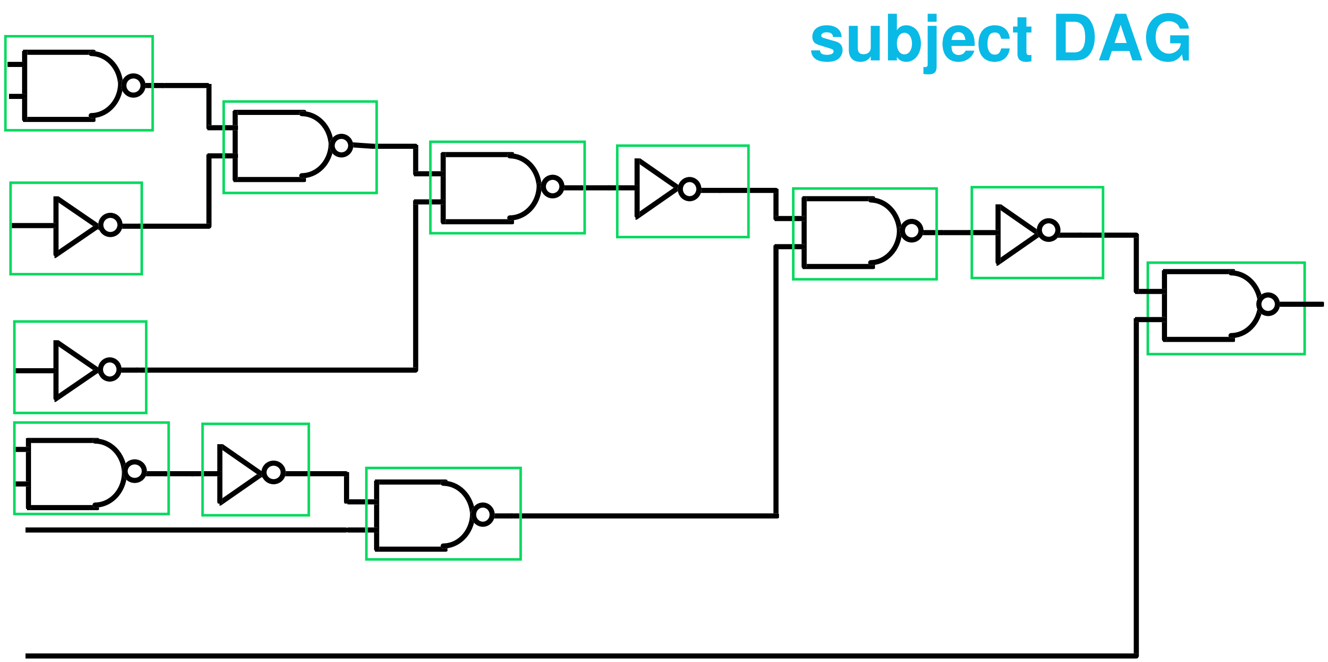


AOI22 5



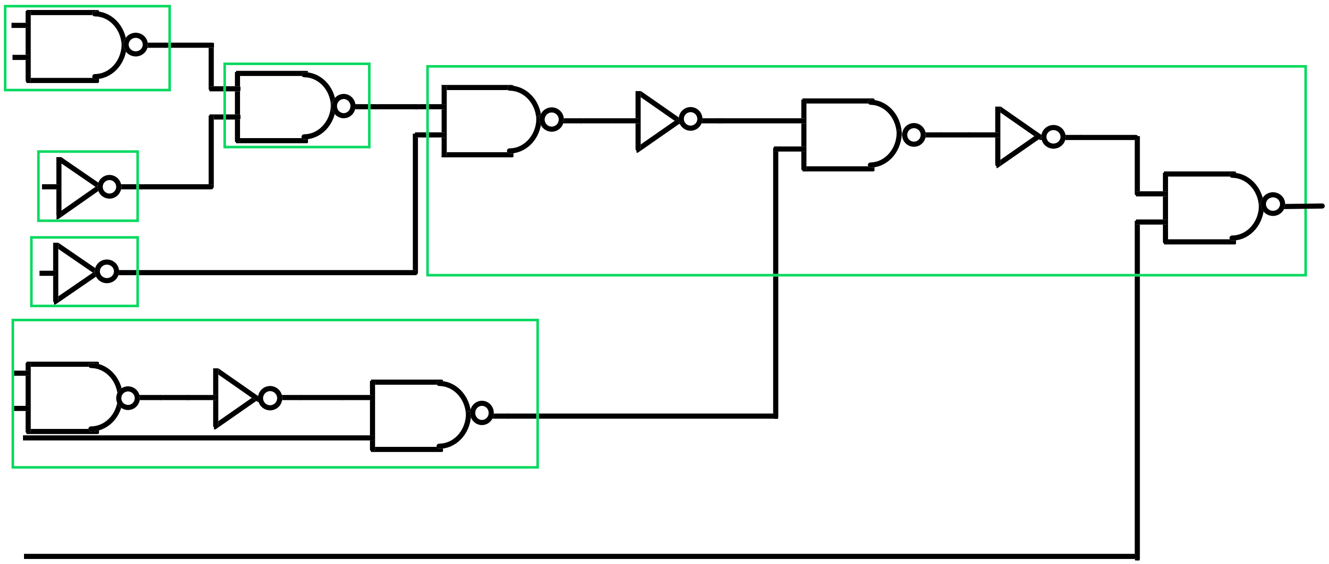


Trivial Covering



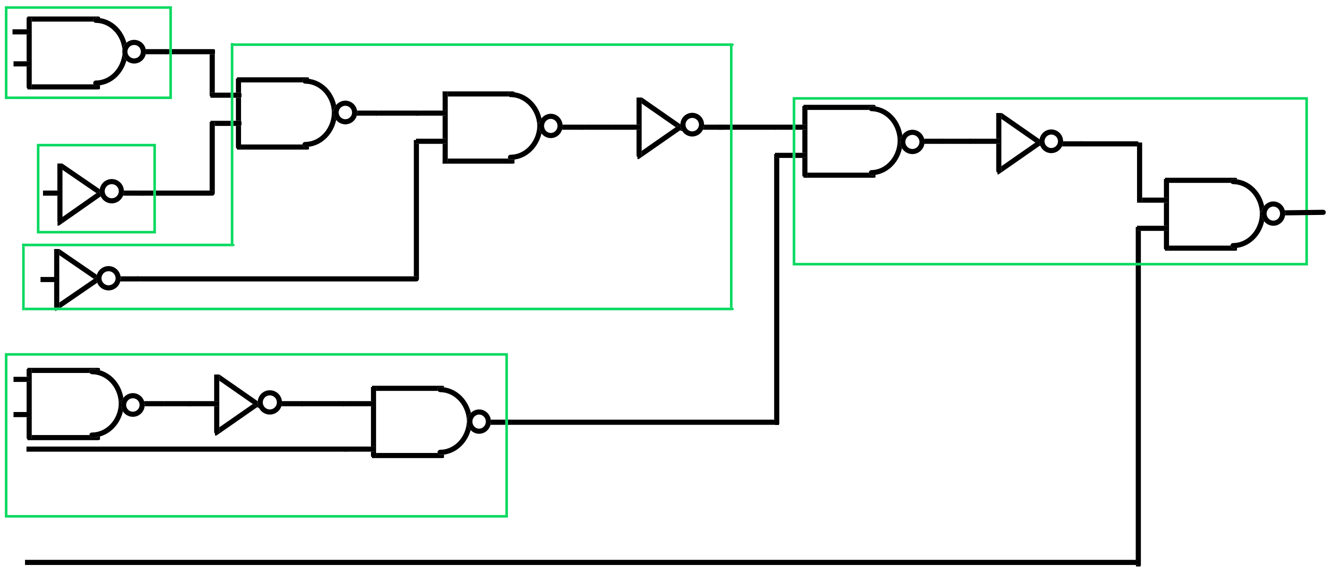
7	NAND2	=	21
5	INV	=	10
			31

Covering #1



2 INV	= 4
2 NAND2	= 6
1 NAND3	= 4
1 NAND4	= 5
	19

Covering #2

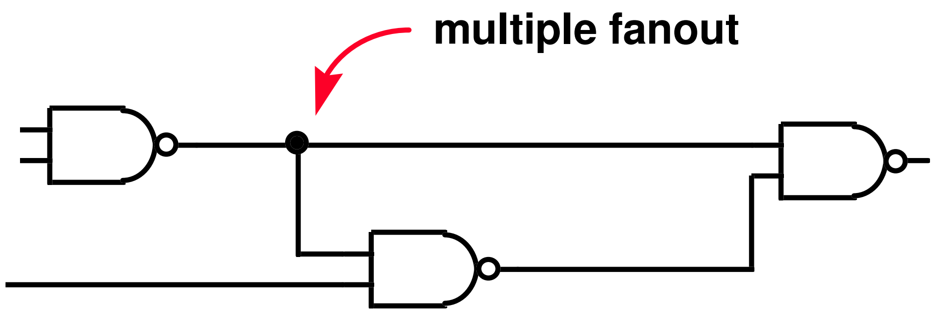


1 INV	= 2
1 NAND2	= 3
2 NAND3	= 8
1 AOI21	= 4
	17

DAG Covering

Sound Algorithmic approach

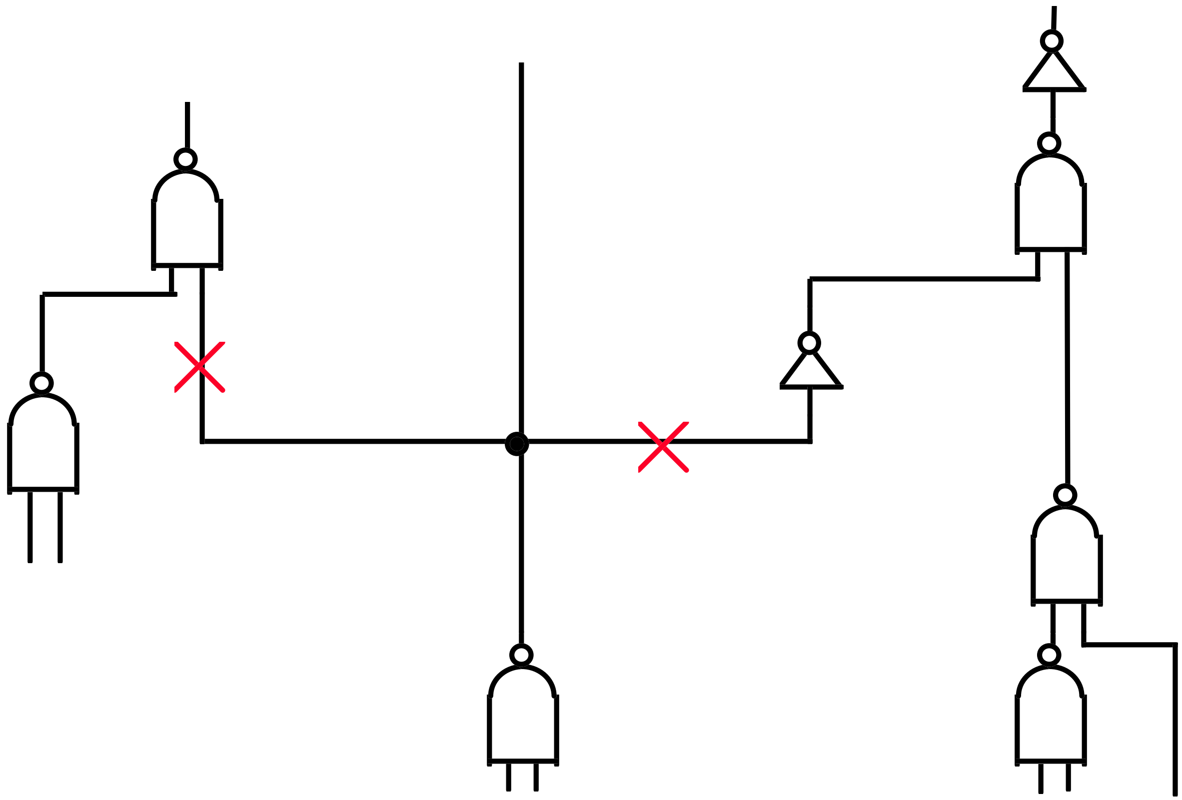
NP-hard optimization problem



Tree covering heuristic: If subject and primitive DAGs are trees, efficient algorithm can find optimum cover in linear time

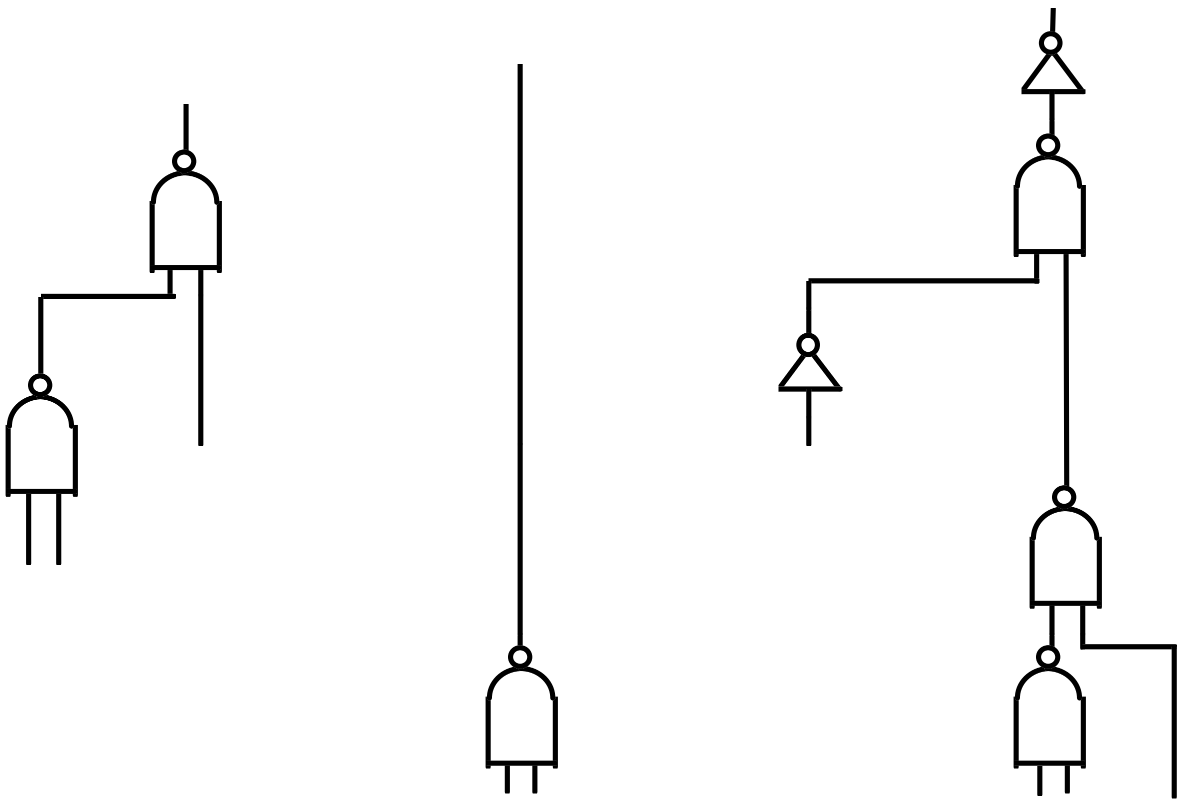
$\Rightarrow$ **dynamic programming formulation**

Partitioning a Graph



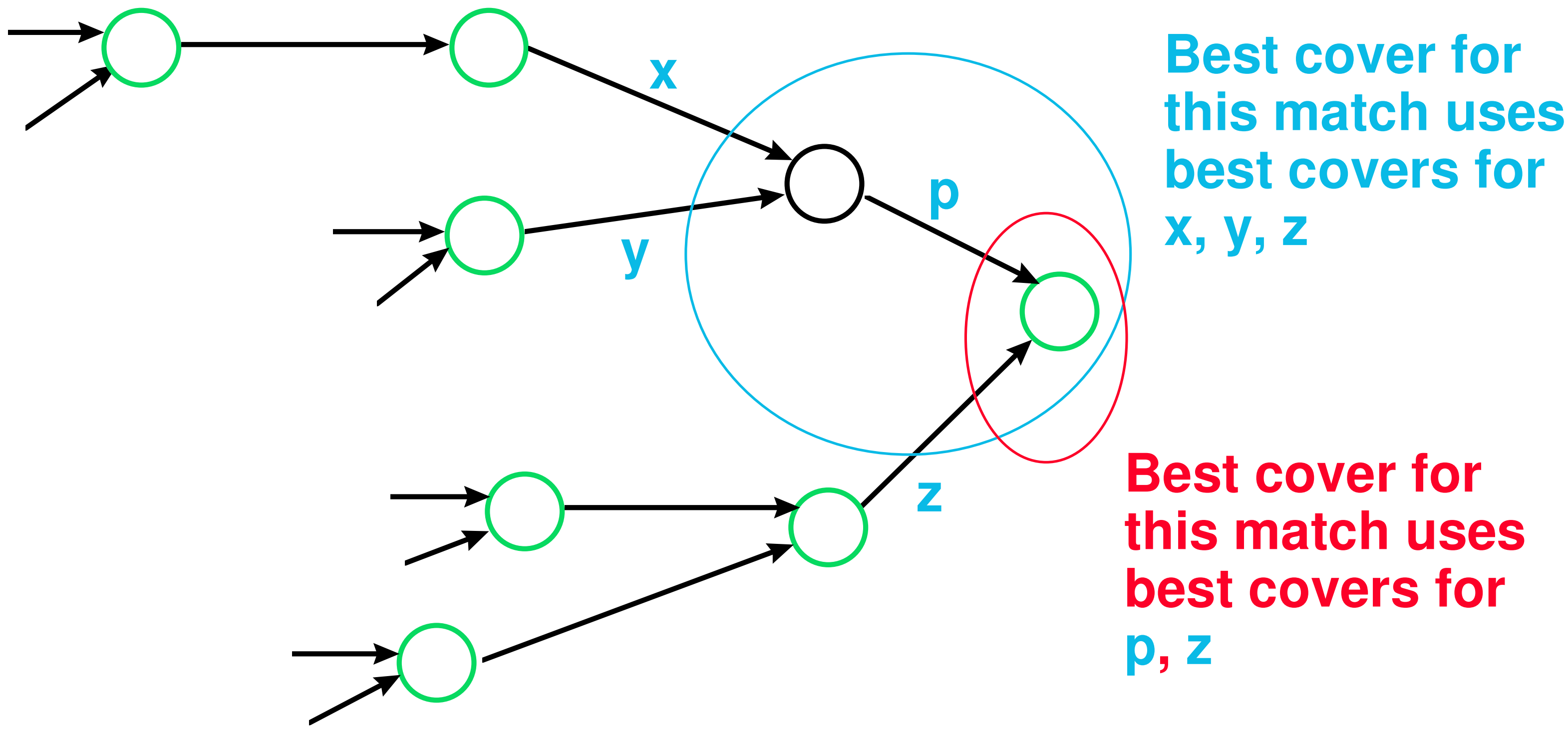
Resulting Trees

Break at multiple fanout points

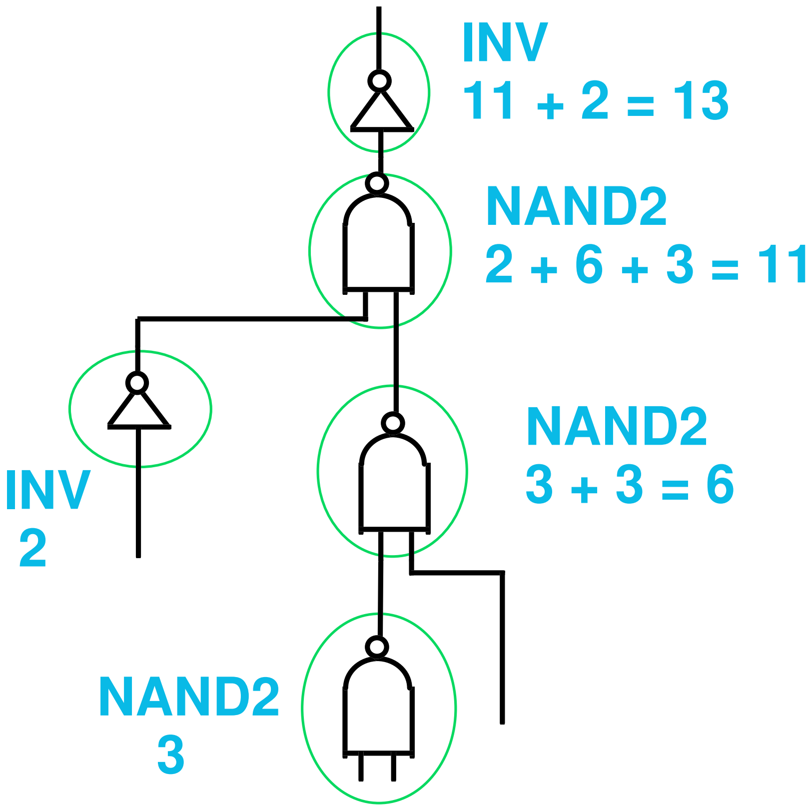


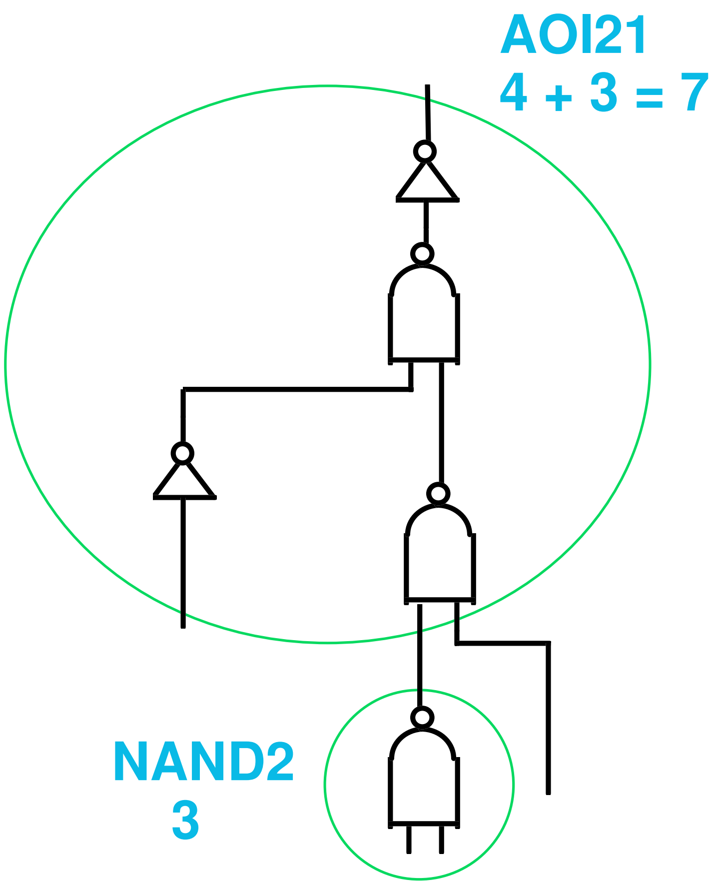
Dynamic Programming

Principle of optimality: Optimal cover for a tree consists of a match at the root of the tree plus the optimal cover for the sub-trees starting at each input of the match



Optimum Tree Covering





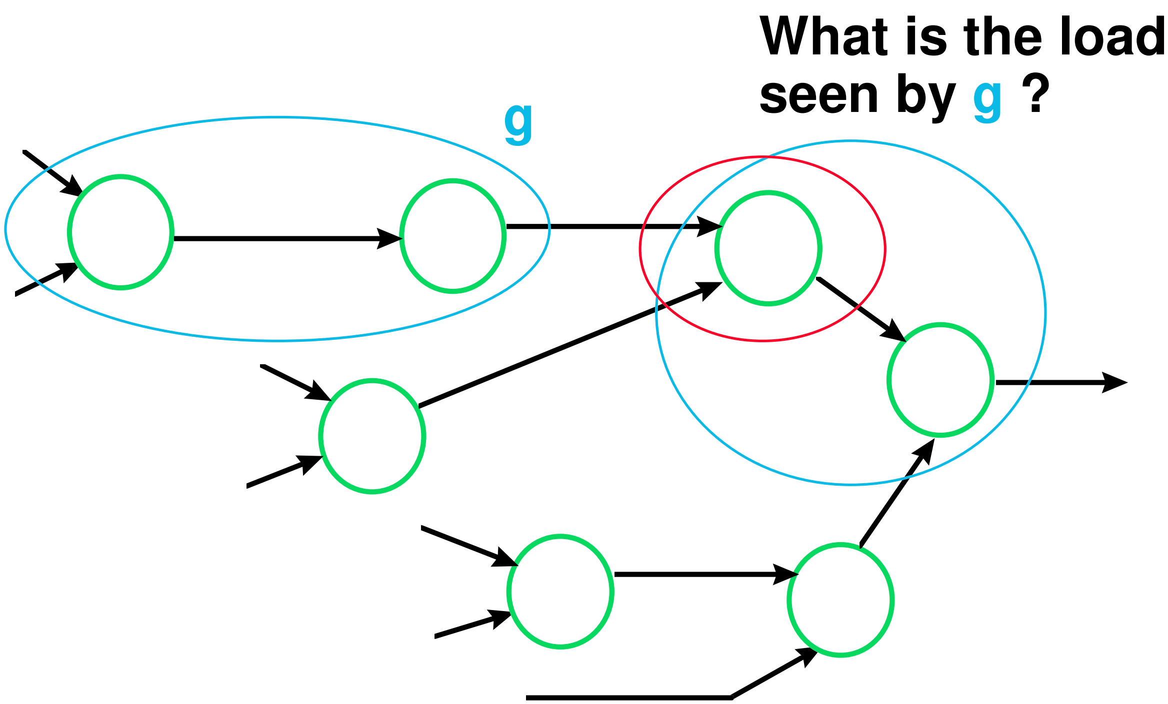
Delay Optimization

If delay is independent of the gate which is driven:

- Tree covering algorithm provides minimum arrival-time cover
  a) Compute and store arrival time at each node
  b) Choose minimum-arrival time match at each node

For more general delay model, principle of optimality does not apply

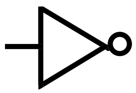
Delay Optimization - 2



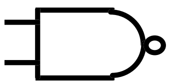
Optimum match depends on forward (unmapped) part of the tree

Library and Delay Information

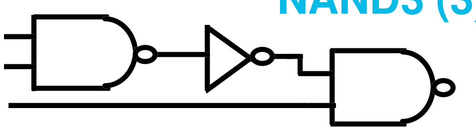
Area



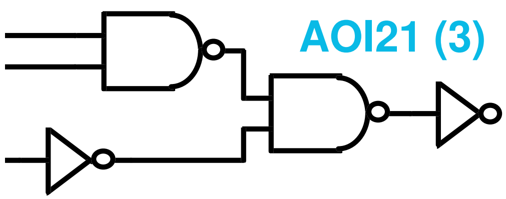
INV (1)



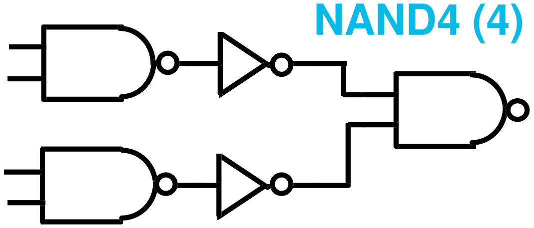
NAND2 (2)



NAND3 (3)



AOI21 (3)

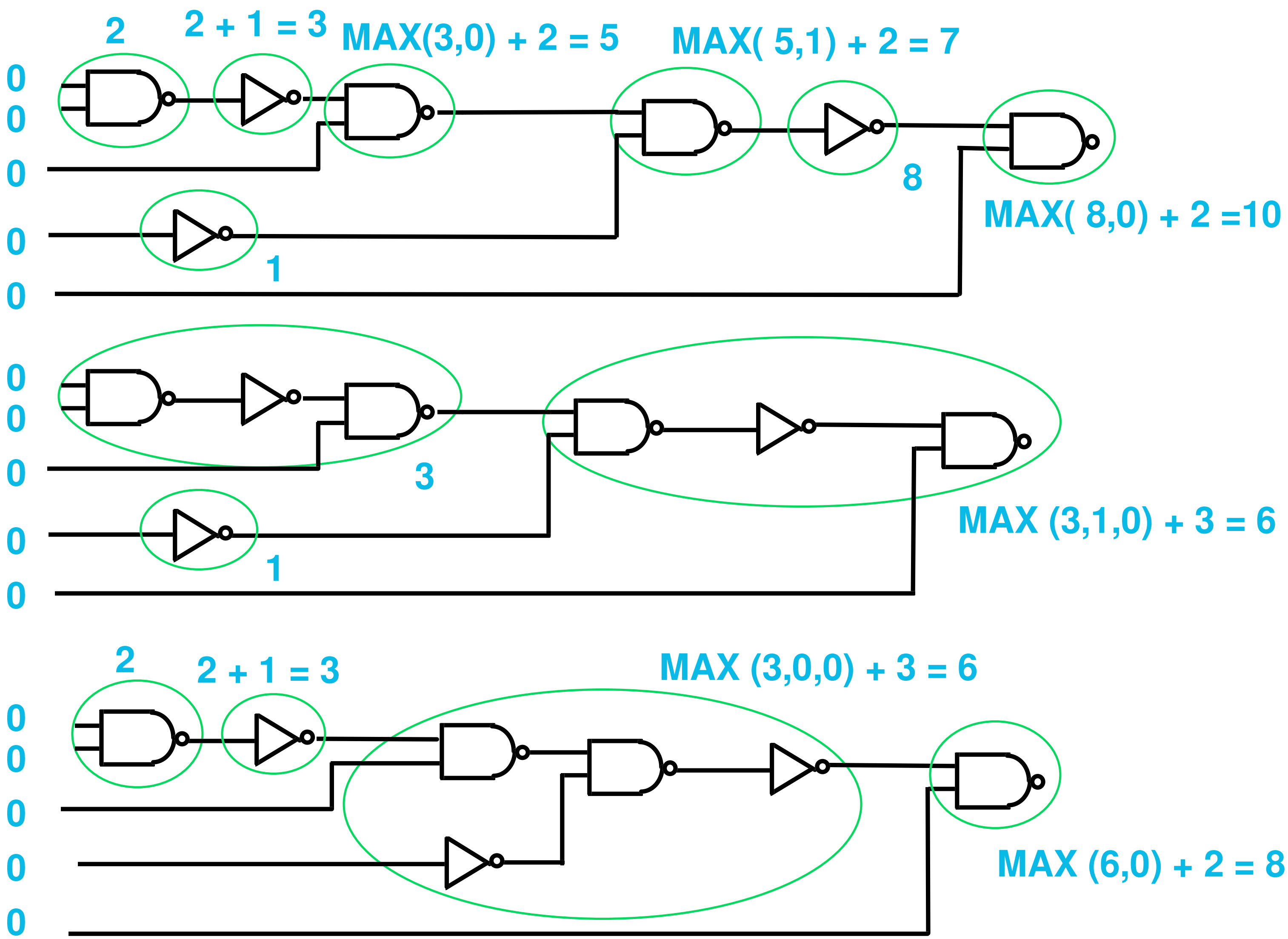


NAND4 (4)

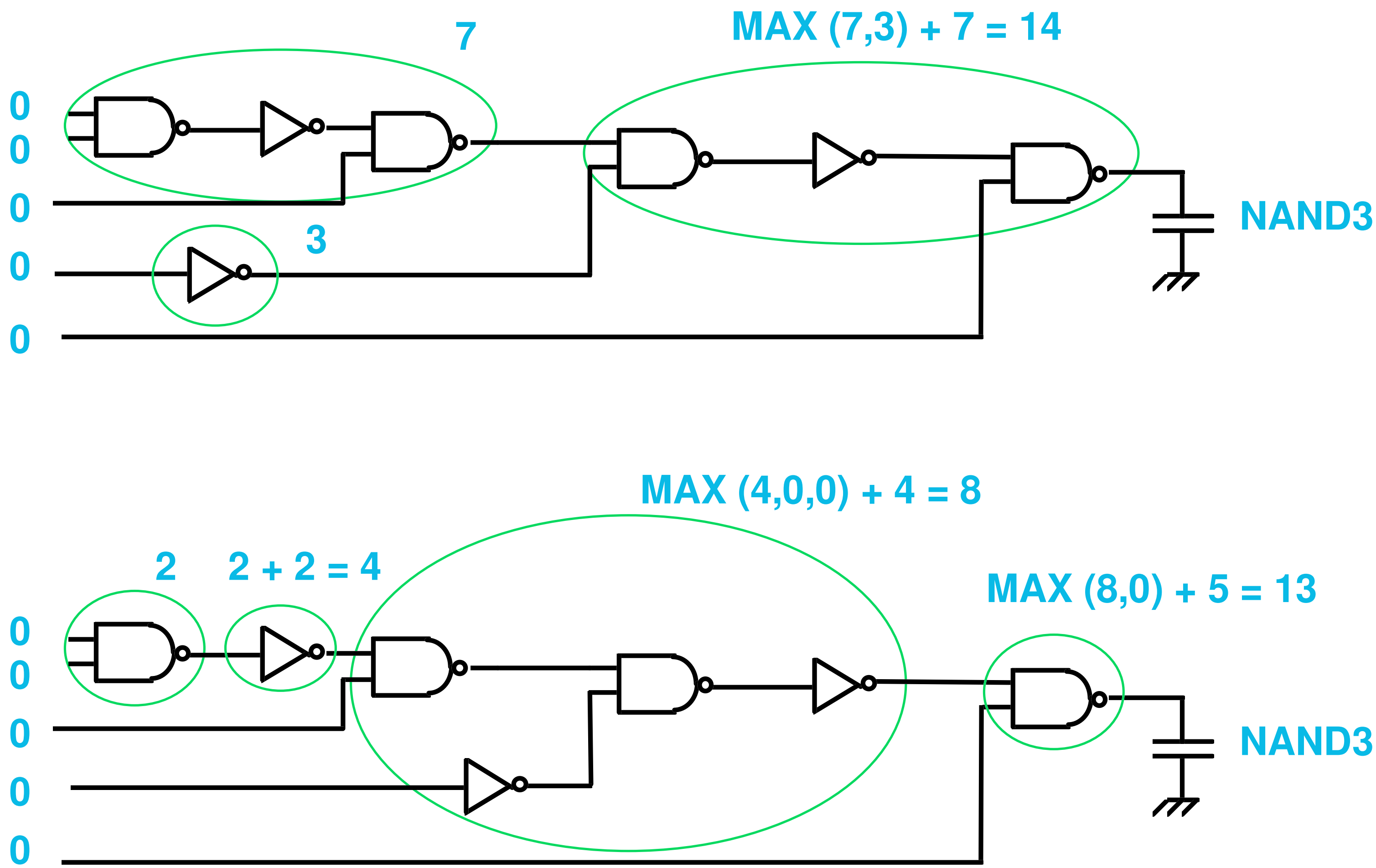
Load-Dependent Delay

INV (1)	NAND2 (2) AOI21	NAND3 (3) NAND4
INV (2)	NAND2 (4) AOI21	NAND3 (5) NAND4
INV (3)	NAND2 (5) AOI21	NAND3 (7) NAND4
INV (3)	NAND2 (4) AOI21	NAND3 (7) NAND4
INV (5)	NAND2 (9) AOI21	NAND3 (12) NAND4

Constant (Inverter) Load Covering

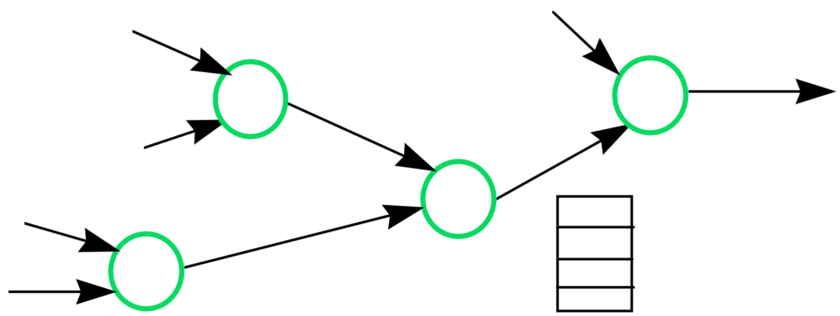


Variable Load



better for real loads!

Variable Load Delay Optimization



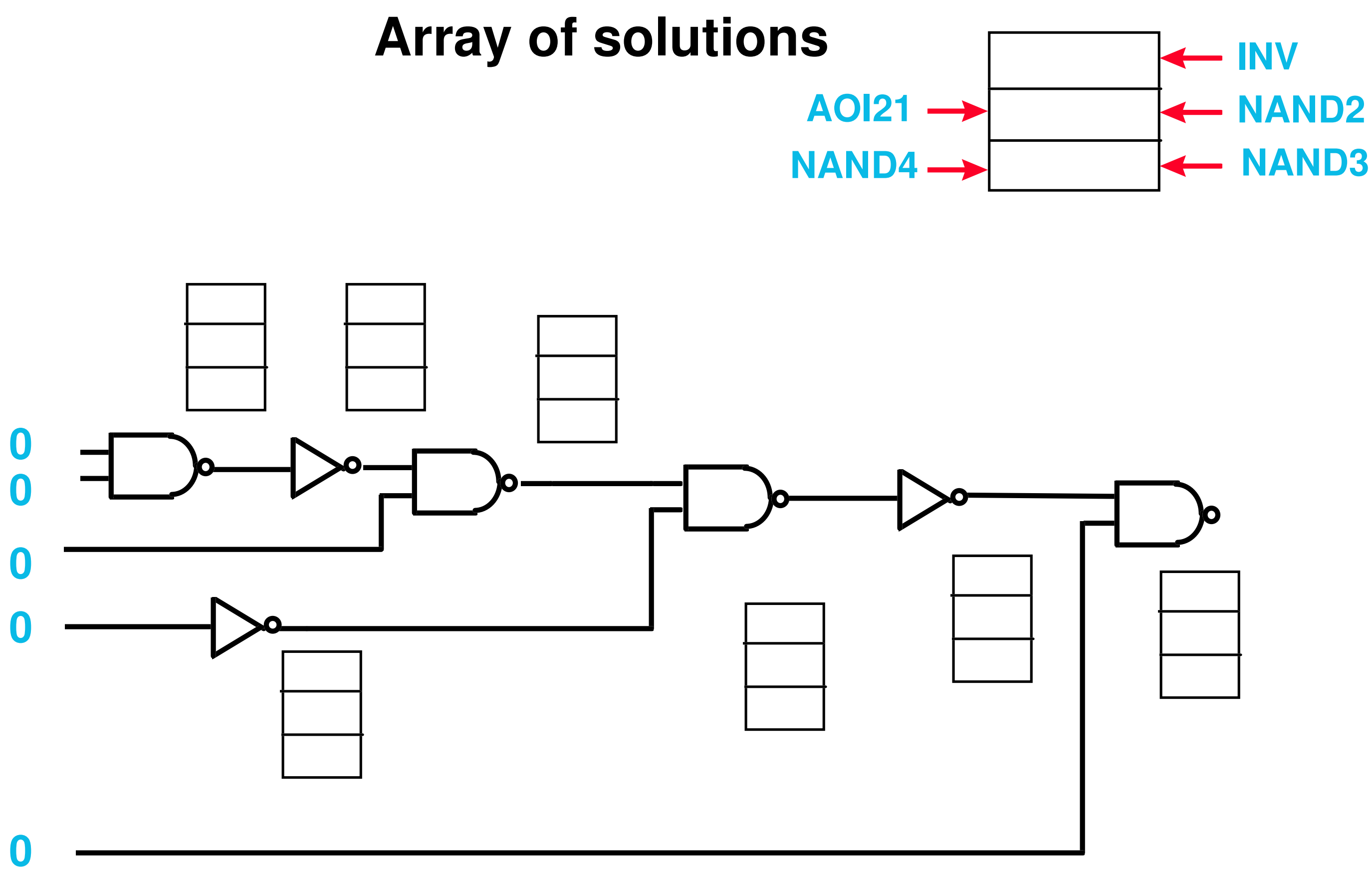
Create bin for each load value

Array of solutions at each node, one per load value

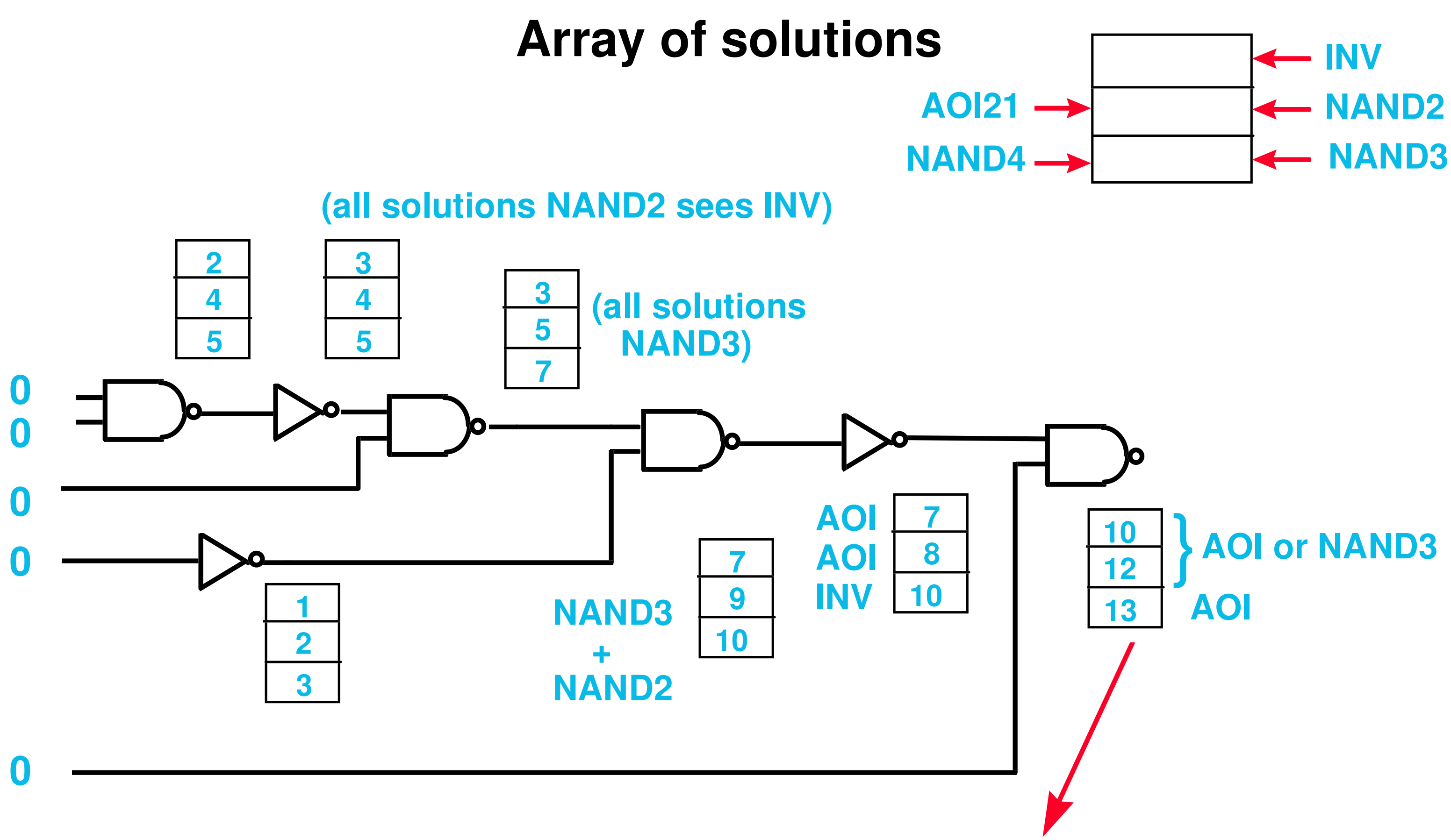
Compute arrival time for each match for <u>each load value</u>

When evaluating a match, use the optimal solution at the input node which is appropriate for the load presented by this match

Variable Load Covering



Variable Load Covering Result

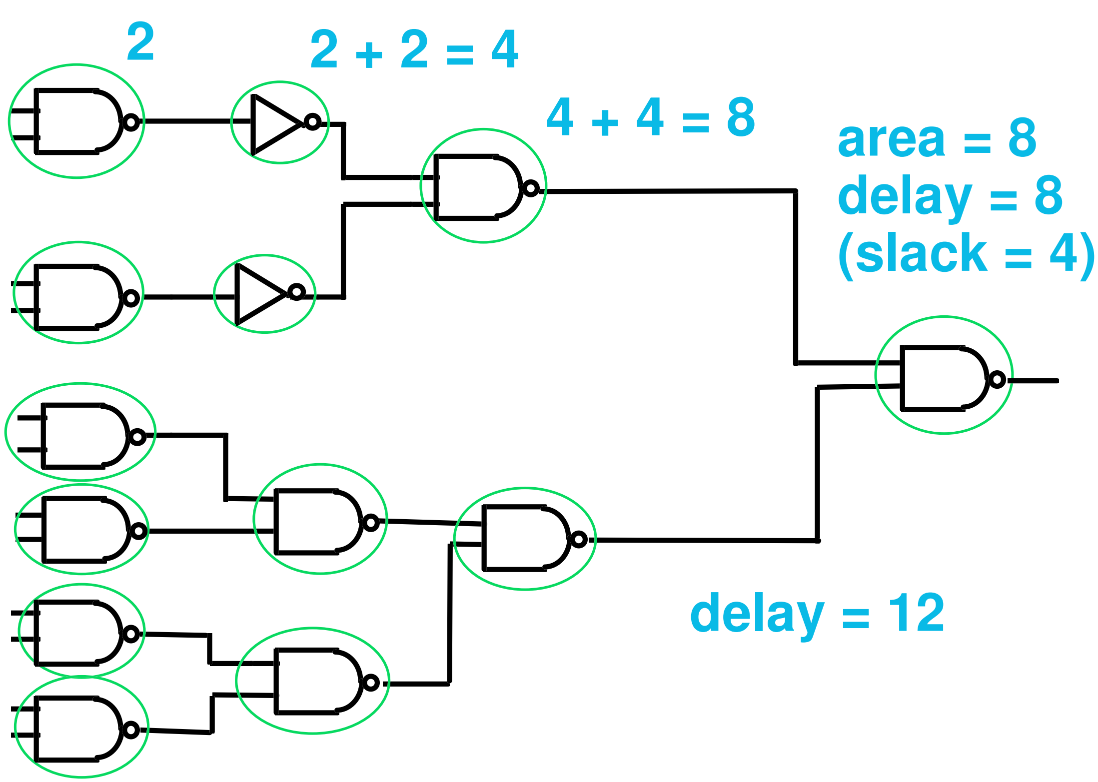


If driving NAND3 will get AOI21 solution with arrival time 13

If driving INV choose either AOI21 or NAND3 solution

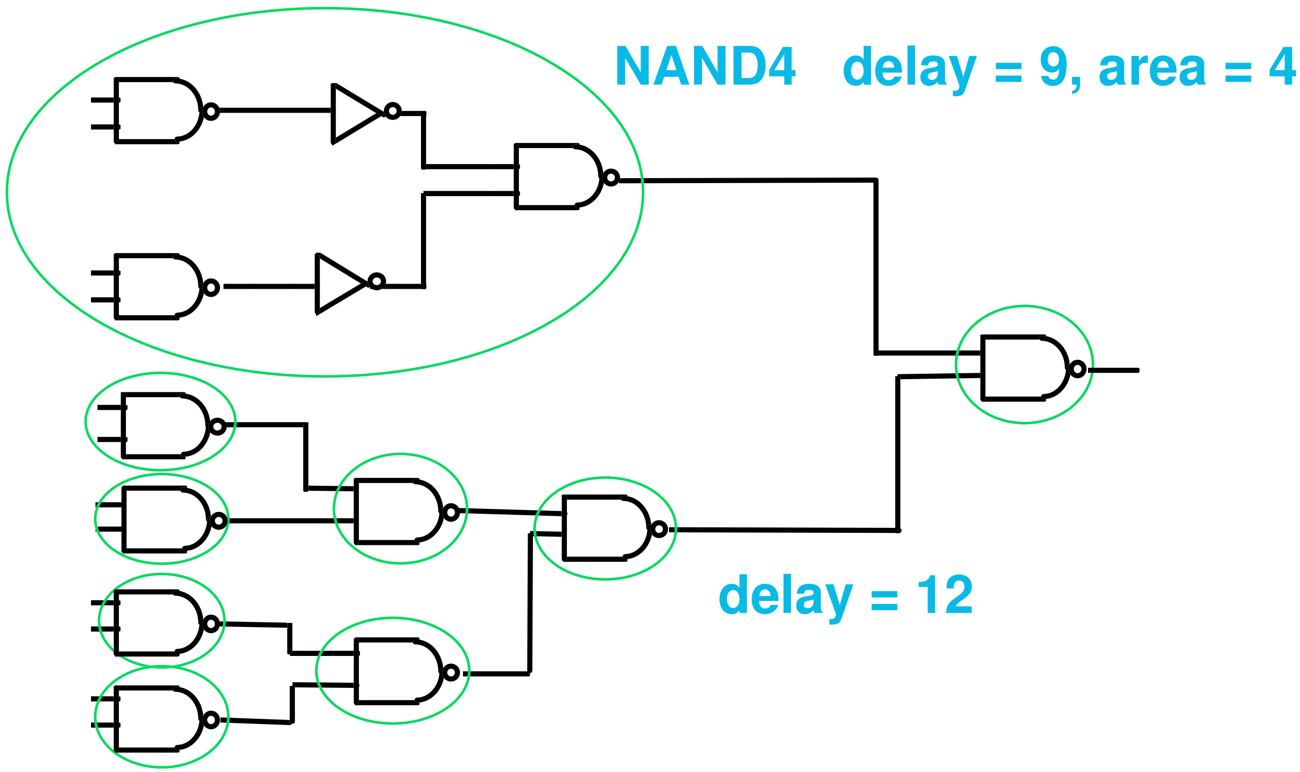
Minimum Arrival Time Cover

Makes all sub-trees maximally fast which wastes area



Reclaiming Area

Identify sub-trees with slack > 0 and check if lesser or minimum area result can be used instead.



Summary of Technology Mapping

Elegant DAG covering formulation

Heuristics based on tree covering for area and delay

Can we identify polynomial time heuristics for covering graphs that have special properties, but are not trees?

Integrate technology mapping with logic restructuring (e.g., algebraic factorization)