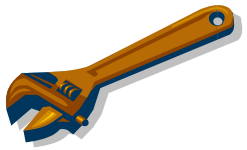
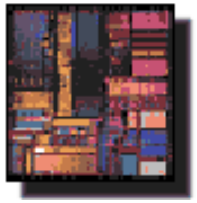






ECE 595Z
Digital VLSI Design Automation

Module 5 (Lectures 14-20): Multi-level Synthesis



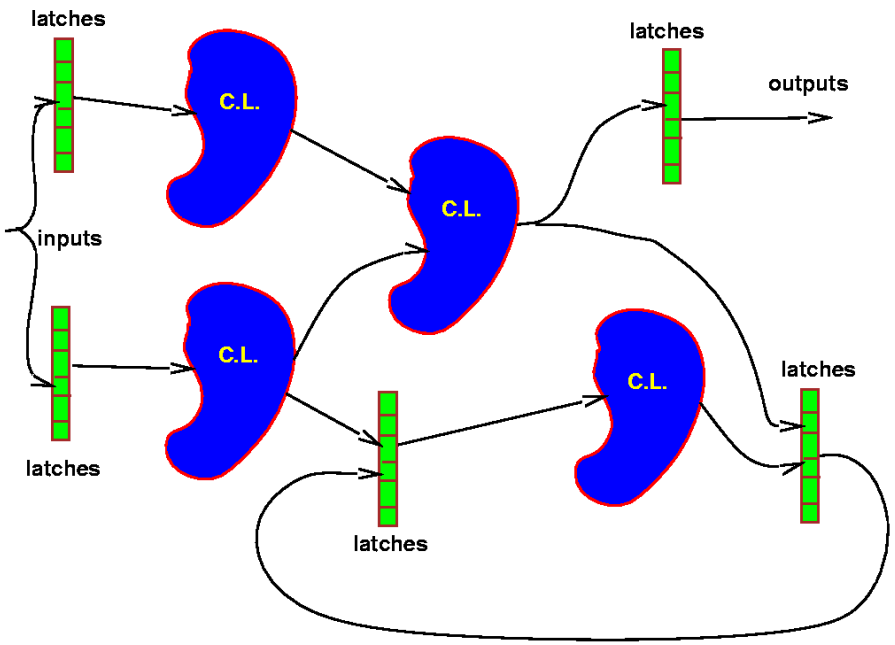
Anand Raghunathan

MSEE 348

raghunathan@purdue.edu

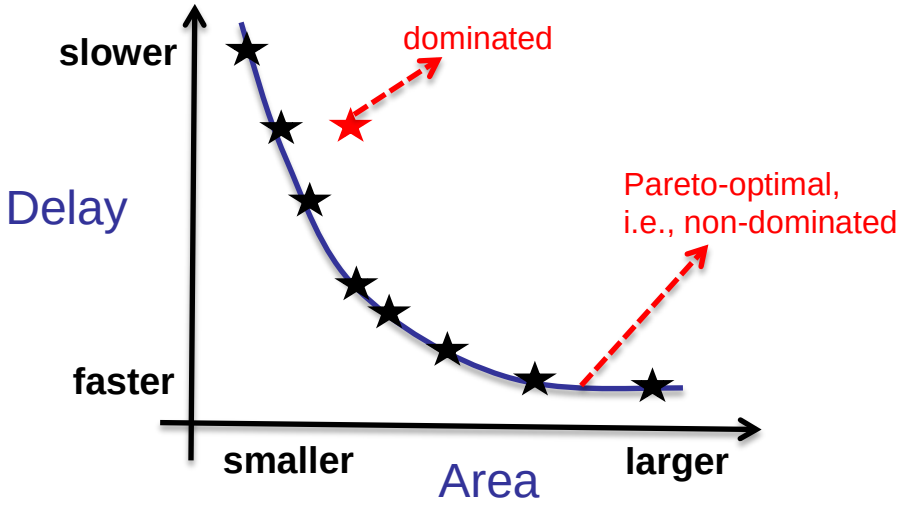
Structure of a Logic Circuit

- Combinational logic, memory elements (FFs, latches), I/O
- Combinational logic synthesis focuses on optimizing the combinational logic
  - Two-level synthesis is one form of combinational logic synthesis



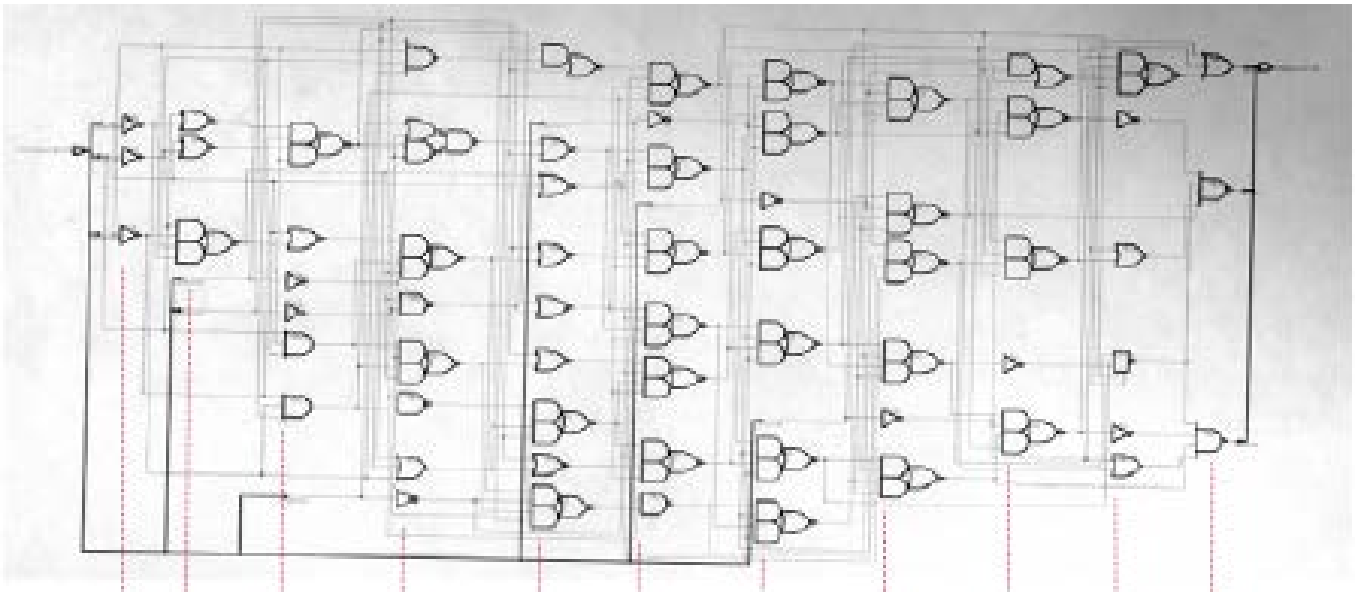
Why Multi-level?

- Two-level implementation is impractical (too large) for many functions
  - N-bit adder has $O(2^N)$ product terms in minimum cover
- Specification by designers (Verilog, VHDL) is often naturally in multi-level form
- Multiple metrics of interest (a specific point in the tradeoff space between area vs. delay, delay vs. power, *etc.* is not enough)
  - Two-level implementations represent only a small part of this space



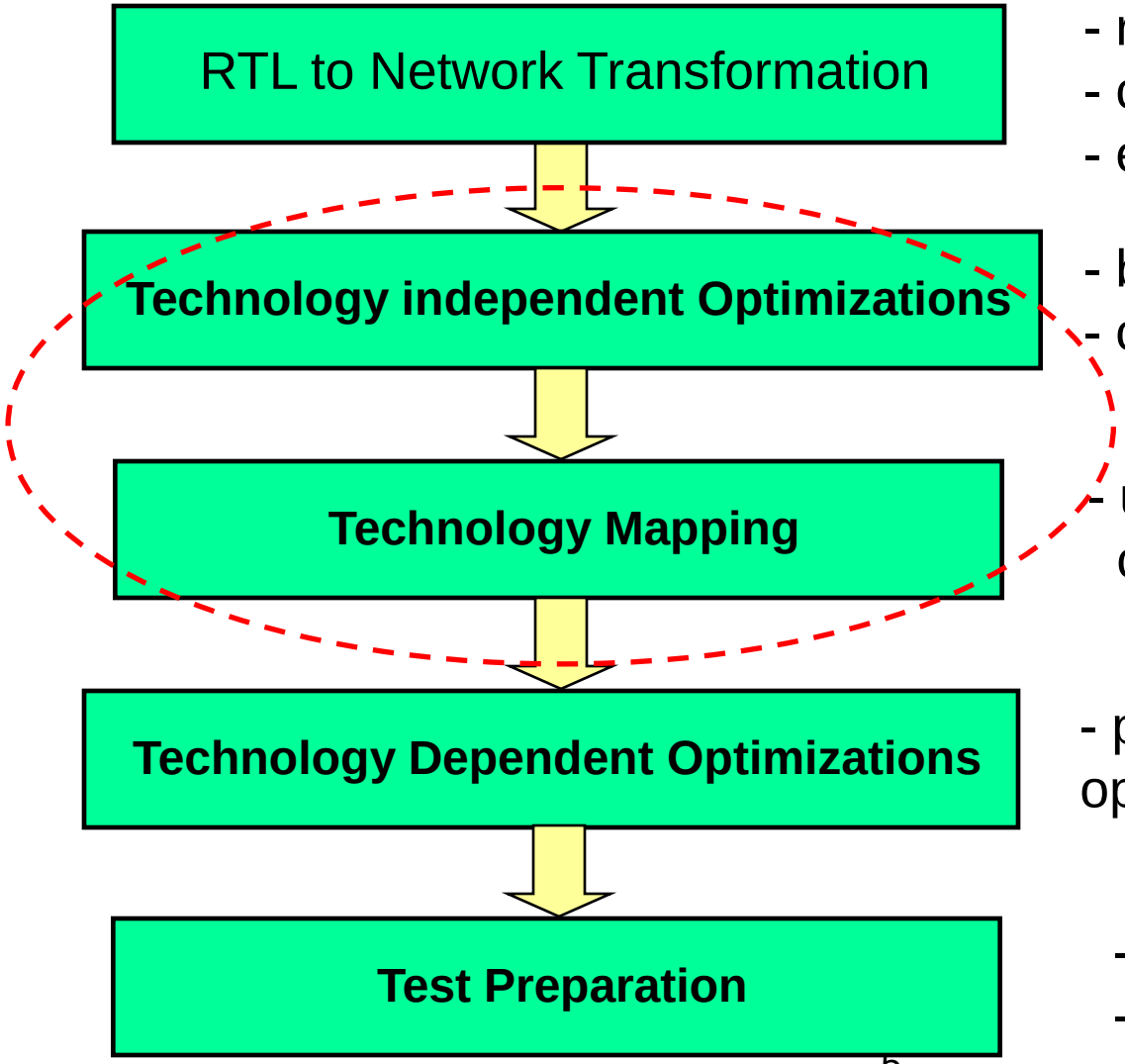
Multi-level Logic Implementation

- Logic depth in modern microprocessors: 10-25



11 “levels” of logic

Typical Synthesis Flow



- read HDL
- control/data flow analysis
- elaboration

- basic logic restructuring
- crude estimates

- use logic gates from target cell library

- physically driven optimizations

- improve testability
- test logic insertion

Steps in Multi-level Synthesis that we will Discuss

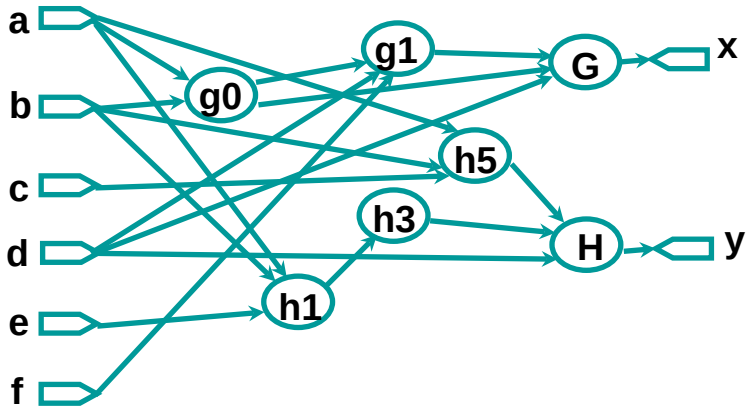
- Technology independent optimization
  - Optimize the network independent of a specific fabrication process (cell library)
  - Approximate metrics for area (literals), delay (levels in circuit), …
- Technology Mapping
  - Map the network to a specific cell library
  - More accurate estimation of metrics

Reading

- Introduction to multi-level synthesis
  - De Micheli, Chapters 8.1-8.2
  - Hachtel & Somenzi, Chapters 10.1-10.2

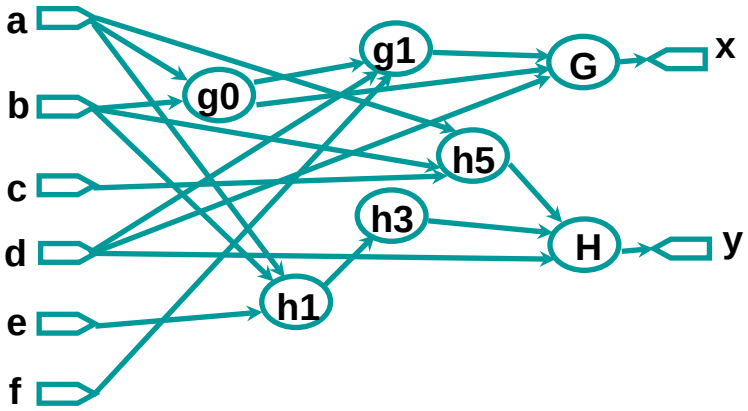
Representation : Boolean Network

- Abstraction of multi-level circuits
- Directed Acyclic Graph (DAG)
  - Nodes represent primary inputs, primary outputs, and internal functions
  - Edges : indicate input-output relationships between nodes



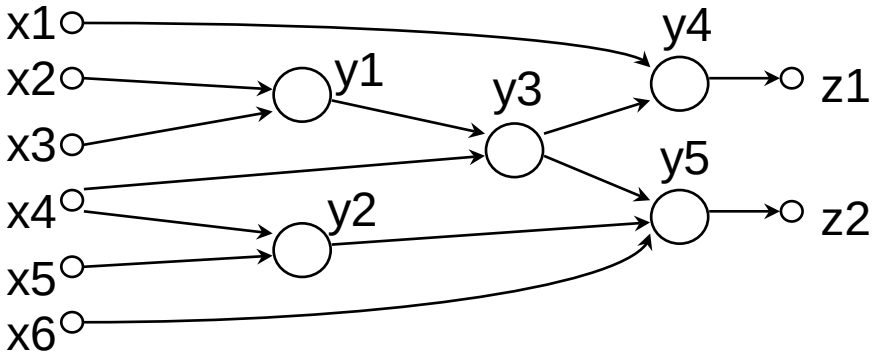
Representation : Boolean Network

- What is inside each node?
- A Boolean function!
  - Simple gate
  - SOP expression in terms of primary inputs and outputs of other nodes
  - Arbitrary logic expression



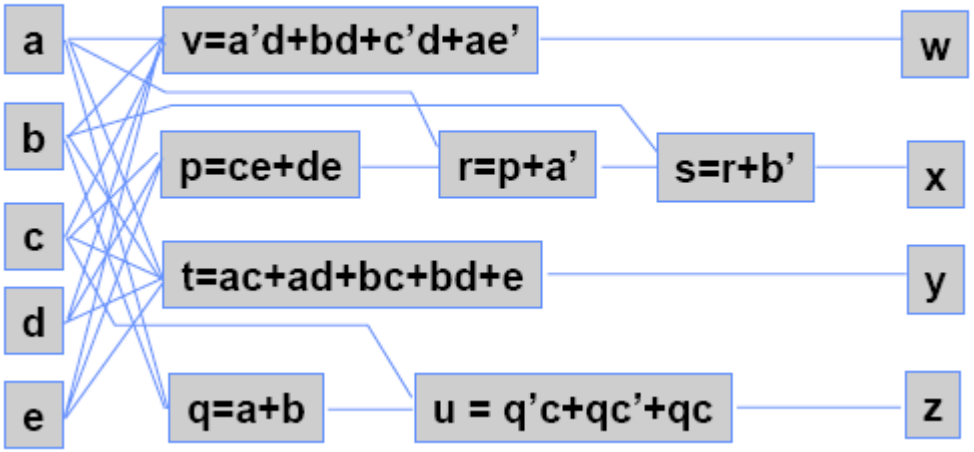
Boolean Network

- Directed Acyclic Graph
- Nodes
  - Primary Inputs
  - Primary Outputs
  - Intermediate nodes
- Edges : Connectivity between nodes
- Fan-in, transitive fan-in
- Fan-out, transitive fan-out



$$y_1 = f_1(x_2, x_3) \quad = x_2' + x_3'$$
$$y_2 = f_2(x_4, x_5) \quad = x_4' + x_5'$$
$$y_3 = f_3(x_4, y_1) \quad = x_4' y_1'$$
$$y_4 = f_4(x_1, y_3) \quad = x_1 + y_3'$$
$$y_5 = f_5(x_6, y_2, y_3) = x_6 y_2 + x_6' y_3'$$

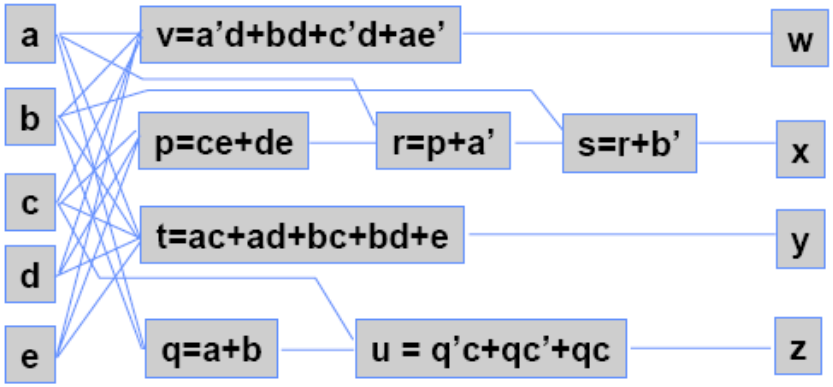
Boolean Network : Example



p = ce + de
q = a + b
r = p + a'
s = r + b'
t = ac + ad + bc + bd + e
u = q'c + qc' + qc
v = a'd + bd + c'd + ae'
w = v
x = s
y = t
z = u

Boolean Network : Example

- How do we measure the complexity of a network?
  - How many gates?
  - How many literals?



General Approach to Multi-level Synthesis

- **Much** more complex than two-level minimization
  - Give up any hope of optimality
- Iterative Improvement Approach
  - Similar to ESPRESSO
  - Apply transformations to improve the network as long as there is improvement
- User-driven
  - "Scripts" that specify sequences of optimization steps to perform

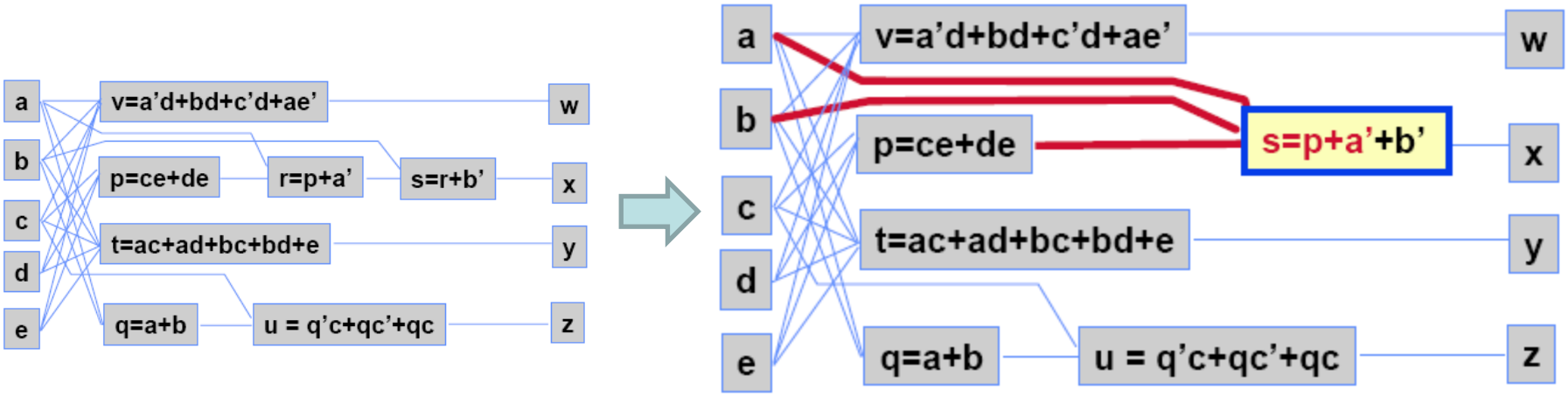
Transformations on Boolean Networks

- Scope
  - Global
    - Any optimization across nodes
    - Re-structuring of entire network
  - Local
    - Optimizations within a node
- Effect
  - Reducing # of nodes
    - Elimination, Substitution
  - Adding new nodes
    - Extraction, Factoring
  - Simplifying nodes
    - Directly apply what we learnt in two-level minimization

Transformation Example: Elimination

- Reducing the # of nodes in the network
  - Requires collapsing of the function performed by the node into the other nodes that it feeds

Example:

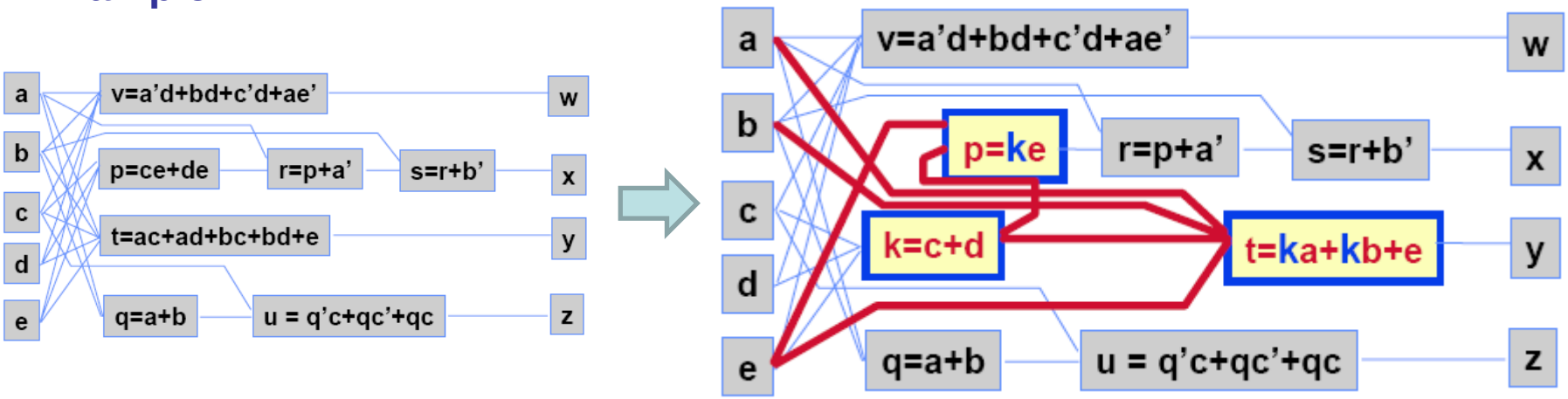


Question : What effect could this have on area (literals) and delay?

Transformation Example: Extraction

- Increases the # of nodes in the network
  - Creates a new node that implements a common sub-expression of two or more nodes in the original network
  - Aggressively substitute the output of the new node in other nodes

Example:

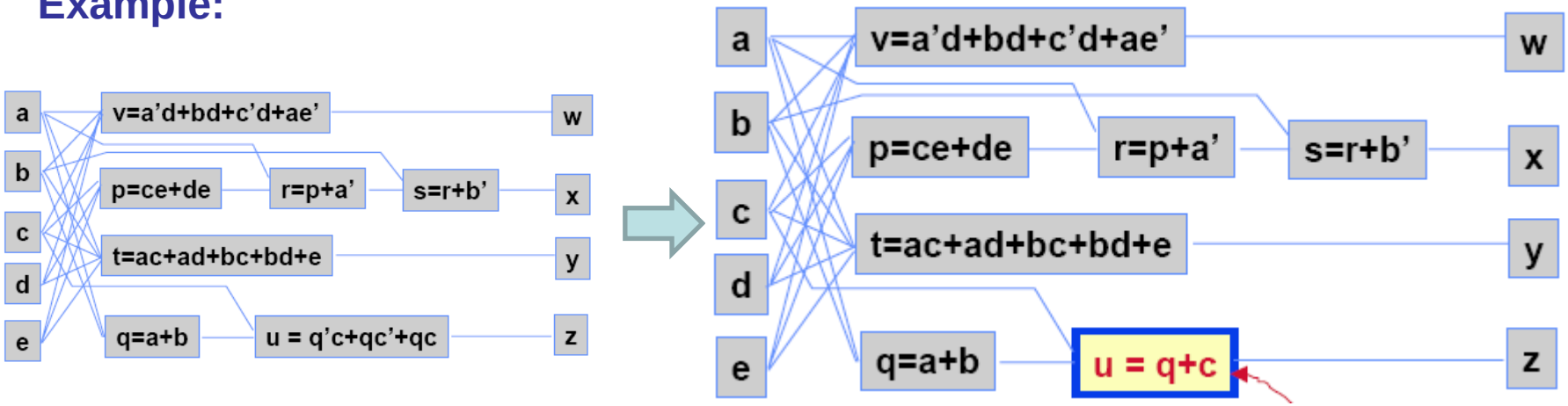


Question : What effect could this have on area (literals) and delay?

Transformation Example: Simplification

- Simplify the internal logic expression for a node
  - Direct application of ESPRESSO if the node functions are represented in SOP form!
  - Usually a local change (exception : eliminating a variable)

Example:



Question : What effect could this have on area (literals) and delay?

Multi-level synthesis operations: Summary

- Global
  - Extraction
    - Find common **factors** for two or more nodes
  - Elimination / Collapsing
    - Collapse a node into it's fan-outs
  - Re-substitution
    - Re-substitute node into other nodes in the network through **factoring**
- Local
  - Decomposition
    - Decompose a node through **factoring**
  - Simplification
    - Use ESPRESSO to simplify the expression inside a node